Package 'BIRDS'

March 20, 2020

Type Package

Title Biodiversity Information Review and Decision Support

Version 0.1

URL https://github.com/greensway/BIRDS

BugReports https://github.com/Greensway/BIRDS/issues

Description It helps making the evaluation and preparation of biodiversity data easy, systematic and reproducible. It also helps the users to overlay the point observations into a custom grid that is useful for further analysis. The review summarise statistics that helps evaluate whether a set of species observations is fit-for-use and take decisions upon its use of on further analyses. It does so by quantifying the sampling effort (amount of effort expended during an event) and data completeness (data gaps) to help judge whether the data is representative, valid and fit for any intended purpose. The 'BIRDS' package is most useful when working with heterogeneous data sets with variation in the sampling process, i.e. where data have been collected and reported in various ways and therefore varying in sampling effort and data completeness (i.e. how well the reported observations describe the true state). Primary biodiversity data (PBD) combining data from different data sets, like e.g. Global Biodiversity Information Facility (GBIF) mediated data, commonly vary in the ways data has been generated - containing opportunistically collected presence-only data together with and data from systematic monitoring programs. The set of tools provided is aimed at understanding the process that generated the data (i.e. observing, recording and reporting species into databases). There is a non-vital function on this package (makeDggrid()) that depends the package 'dggridR' that is no longer on CRAN. You can find it here <https://github.com/r-barnes/dggridR>. References: Ruete (2015) <doi:10.3897/BDJ.3.e5361>; Szabo, Vesk, Baxter & Possingham (2010) <doi:10.1890/09-0877.1>; Telfer, Preston 6 Rothery (2002) <doi:10.1016/S0006-3207(02)00050-2>.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0)

SystemRequirements GDAL (>= 2.0.1)

Imports data.table, dplyr, esquisse, geosphere (>= 1.5), leaflet (>= 2.0), lubridate (>= 1.7.4), magrittr, mapedit (>= 0.5), methods, nnet, rgeos (>= 0.4), rgdal (>= 1.4), rlang, sf (>= 0.7), shiny (>= 1.0), sp (>= 1.3), stringr (>= 1.4), shotGroups, xts

Suggests biogeo, CoordinateCleaner, covr, dggridR, knitr, KnowBR, leaflet.extras, leafpm, maps, rgbif, rmarkdown, taxize, testthat, utils, vegan

RoxygenNote 7.1.0

VignetteBuilder knitr

NeedsCompilation no

Author Debora Arlt [aut] (<https://orcid.org/0000-0003-0874-4250>), Alejandro Ruete [aut, cre] (<https://orcid.org/0000-0001-7681-2812>), Anton Hammarström [aut]

Maintainer Alejandro Ruete <`aleruete@gmail.com`>

Repository CRAN

Date/Publication 2020-03-20 14:00:02 UTC

R topics documented:

BIRDS 3
bombusObs 4
bombusObsShort 5
bryophytaObs 5
communityMatrix 6
createVisits 7
drawPolygon 8
exploreVisits 9
exportBirds 10
exposeIgnorance 11
focalSpReport 12
focalSpSummary 13
getGridIDs 14
getUTMproj 14
gotaland 15
gridAsString 15
listSpecies 16
makeCircle 16
makeDggrid 17
makeGrid 18
OB2Polygon 20
obsData 20
obsIndex 21
organizeBirds 22

overlayBirds 25
removeObs 26
SB 27
searchPolygon 28
simpleSB 28
simplifySpp 28
spatialVisits 29
speciesSummary 30
summarizeBirds 31
visits 32

Index **33**

BIRDS — *BIRDS: Biodiversity Information Review and Decision Support.*

Description

The primary aim of this package is to provide tools for Biodiversity Informatics in order to understand the specis information data and decide about what analyses we can perform and draw approriate conclusions. For this we need to understand the data generation process. With this R package we want to take the user a step closer to understanding the observers' behaviour. The 'BIRDS' packages provides a workflow for reproducable data review, involving three basic steps: organise data, summarise data, review data.

Examples

```
# Organise the data
OB <- organizeBirds(bombusObsShort, sppCol = "scientificName", simplifySppName = TRUE)

OB2 <- organizeBirds(bryophytaObs, sppCol = "species", simplifySppName = FALSE,
      taxonRankCol = "taxonRank", taxonRank = c("SPECIES", "SUBSPECIES","VARIETY"))

# Make a grid that can be used by summariseBirds()
# gotaland is a SpatialPolygonDataFrame provided as an example
grid <- makeGrid(gotaland, gridSize = 10)

# Summarise the data (using the grid to overlay with the organised data)
SB <- summariseBirds(OB, grid=grid)
EBnObs <- exportBirds(SB, dimension = "temporal", timeRes = "yearly",
variable = "nObs", method = "sum")
EBnVis <- exportBirds(SB, dimension = "temporal", timeRes = "yearly",
variable = "nVis", method = "sum")
EB<-exportBirds(SB, "Spatial", "Month", "nYears", "sum")
palBW <- leaflet::colorNumeric(c("white", "navyblue"),
         c(0, max(EB@data, na.rm = TRUE)), na.color = "transparent")
library(sp)
old.par <- par(no.readonly =TRUE)
par(mfrow=c(1,2), mar=c(1,1,1,1))
```

```
plot(EB, col=palBW(EB@data$Jul))
mtext("July", 3)
plot(EB, col=palBW(EB@data$Dec))
mtext("December", 3)
legend("bottomleft", legend=seq(0, max(EB@data, na.rm = TRUE),length.out = 5),
       col = palBW(seq(0, max(EB@data, na.rm = TRUE), length.out = 5)),
       title = "Number of years", pch = 15, bty="n")
par(old.par)
```

bombusObs — *Species observations for the genus* Bombus *spp. in Götaland, Sweden.*

Description

A random sample of 10,000 observations for the genus *Bombus* spp. from an original dataset of 25,848 observations. The observations were accessed via https://www.gbif.org/. Citation for the original dataset: GBIF.org (03 April 2019) GBIF Occurrence Download https://doi.org/10.15468/dl.jhthmb
Searching parameters
And: (All must apply)

Country or area Sweden

Publisher
- b8323864-602a-4a7d-9127-bb903054e97d
- 28eb1a3f-1c15-4a95-931a-4af90ecb574d
- 6ba9a8cc-513a-4a51-bf93-6f5de8040a96
- 92f51af1-e917-49bc-a8ed-014ed3a77bec
- f314b0b0-e3dc-11d9-8d81-b8a03c50a862

Year Between start of 2000 and end of 2018

Geometry POLYGON((10.55786 59.23218,10.55786 58.90465,11.15112 57.75108,11.8103 57.24339,11.94214 56.58369,12.68921 55.31664,14.35913 55.20395,15.08423 55.70236,16.73218 55.82597,17.45728 57.36209,18.11646 58.47072,18.84155 59.29955,10.55786 59.23218))

Has coordinate true

Scientific name Bombus Latreille, 1802

Has geospatial issue false

Usage

```
bombusObs
```

Format

A data frame with 10,000 rows and 45 variables following DarwinCore standard https://dwc.tdwg.org/

Source

https://www.gbif.org/occurrence/download/0007731-190320150433242

`bombusObsShort` *A short version of bombusObs dataset for faster examples.*

Description

A random sample of 1,000 observations for the genus *Bombus* spp.

Usage

```
bombusObsShort
```

Format

A data frame with 1,000 rows and 45 variables following DarwinCore standard `https://dwc.tdwg.org/`

Source

`https://www.gbif.org/occurrence/download/0007731-190320150433242`

See Also

`bombusObs`

`bryophytaObs` *Species observations of bryophytes in Götaland, Sweden*

Description

A random sample of 10,000 observations of bryophytes from an original dataset of 178,765 observations. The observations were accessed via `https://www.gbif.org/`. Citation for the original dataset: GBIF.org (03 April 2019) GBIF Occurrence Download `https://doi.org/10.15468/dl.ijr8gw`.
Searching parameters
And: (All must apply)

Country or area Sweden

Publisher
- b8323864-602a-4a7d-9127-bb903054e97d
- 28eb1a3f-1c15-4a95-931a-4af90ecb574d
- 6ba9a8cc-513a-4a51-bf93-6f5de8040a96
- 92f51af1-e917-49bc-a8ed-014ed3a77bec
- f314b0b0-e3dc-11d9-8d81-b8a03c50a862

Year Between start of 2000 and end of 2018

Geometry POLYGON((10.55786 59.23218,10.55786 58.90465,11.15112 57.75108,11.8103 57.24339,11.94214 56.58369,12.68921 55.31664,14.35913 55.20395,15.08423 55.70236,16.73218 55.82597,17.45728 57.36209,18.11646 58.47072,18.84155 59.29955,10.55786 59.23218))

Has coordinate true

Scientific name Bryophyta

Has geospatial issue false

Usage

```
bryophytaObs
```

Format

A data frame with 10,000 rows and 45 variables following DarwinCore standard `https://dwc.tdwg.org/`.

Source

`https://www.gbif.org/occurrence/download/0007732-190320150433242`

`communityMatrix` *Create a community matrix*

Description

A function that counts the number of observations or visits per grid cell for all species.

Usage

```
communityMatrix(x, sampleUnit = "observation")
```

Arguments

`x`	an object of class ‘SummarizeBirds’.
`sampleUnit`	an string specifying the sample unit within a grid cell. Options are “observation” (default) or “visit”. If spillOver=TRUE and visits are defined by locality, it may happen that some species observations are counted in more than one grid cell.

Value

a `matrix` with counts of observations or visits for each species on each non-empty grid cell.

See Also

`summarizeBirds`, `exportBirds`

Examples

```
grid <- makeGrid(searchPolygon, gridSize = 10)
SB <- summarizeBirds(organizeBirds(bombusObsShort), grid=grid)
CM <- communityMatrix(SB, sampleUnit="visit")
```

`createVisits` *Create unique visits IDs*

Description

Takes a dataframe and a vector of column names and classifies each row of the dataframe based on the combination of values in the specified columns.

Usage

```
createVisits(
  x,
  idCols = c("locality", "recordedBy"),
  timeCols = c("day", "month", "year"),
  grid = NULL
)
```

Arguments

`x`	An object of class `data.frame` or `SpatialPointsDataFrame` including at least the columns specified that are used to identify a visit.
`idCols`	A vector with the names of the columns other (than time columns) that are used to identify a visit. This variable cannot be empty. At least the recorders name or any other ID must be provided. Default is the Darwin Core variables `c("locality","recordedBy")`.
`timeCols`	A vector with the names of the time columns that are used to identify a visit. If timeCols=NULL then time is ignored to create a visit ID. Default is the Darwin Core variables `c("day","month","year")`.
`grid`	Either `NULL` to be ignored or an object of class `SpatialPolygons` or `SpatialPolygonsDataFrame` defining the maximum extent of visits effort. Then x must be an object of class SpatialPointsDataFrame

Details

What a visit should be is not always clearly defined and extractable in a dataset. A reasonable assumption is that a visit could be identified from the records made by one person on a certain day and at a specific location or site. The default value for the variable column is therefore that a visit is identified by the Darwin Core variables `c("locality","day","month","year","recordedBy")`.

Value

A vector of the same length as the number of rows as the dataframe with a unique number for each combination of the values in the specified columns.

Examples

```
OB <- organizeBirds(bombusObs)
tmp.vis <- createVisits(bombusObs,
                        idCols=c("locality", "recordedBy"),
                        timeCols=c("day", "month", "year"))
visits(OB, name = "visNoRecorder", useAsDefault = TRUE) <- tmp.vis
```

`drawPolygon` *Create the polygon for the study area by drawing into a world map*

Description

Create the polygon for the study area by drawing into a world map

Usage

```
drawPolygon(lat = 0, lng = 0, zoom = 1, editor = "leafpm")
```

Arguments

`lat`	initial geographical coordinate for latititude in decimal degrees (EPSG:4326) for the map to start at. Default = “0”.
`lng`	initial geographical coordinate for longitude in decimal degrees (EPSG:4326) for the map to start at. Default = “0”.
`zoom`	initial zoom level for the map. Range 1 - 19. Default = “1”.
`editor`	type of editor for the drawing tools. Options are “leafpm” (default) and “leaflet.extras”. Requires additional packages `leafpm` and `leaflet.extras`, respectively.

Value

an object of class ‘SpatialPolygon-class’ with a polygon (only the first one drawn) with geodesic coordinates in WGS84 (ESPG:4326).

Examples

```
if(interactive()){
 polygon <- drawPolygon()
}
```

`exploreVisits` *A function to explore the definition of field visits*

Description

A function to explore the definition of field visits. Visits are a central concept in the approach to species observation data used by the BIRDS package. In order to assess if your definition of visit aligns with your grid size, you must explore the spatial extent of visits.

Usage

```
exploreVisits(x, visitCol = NULL, sppCol = "scientificName")
```

Arguments

`x`	an object of class 'OrganizedBirds' (organised BIRDS Spatial Dataframe). See `organizeBirds`.
`visitCol`	name of the column for the visits UID.
`sppCol`	name of the column for species names.

Value

a `data.frame` with summarized data per visit:

- "day"
- "month"
- "year"
- "nObs": number of species observations
- "SLL": species list length (i.e. the number of observed species)
- "effortDiam": the 2 times the maximum of the distances between the centroid of all observation points and any individual observation.
- "medianDist": the median (Q2) of the distances between the centroid and the observations, in meters.
- "iqrDist": the interquartile range of the distances between the centroid the observations, in meters.
- "nOutliers": the number of observations whose distance to the centroid are considered an outlier. Ouliers are defined as distances grater than the Q3 * 1.5 (i.e. `length(boxplot.stats(distances)$out)` as all distances are positive).

See Also

`createVisits`, `organiseBirds`

Examples

```
if(interactive()){
# create a visit-based data object from the original observation-based data
OB<-organizeBirds(bombusObs)
visitStats<-exploreVisits(OB)
 esquisse::esquisser(visitStats)
# alternatively, plot the variable you want, e.g.:
# to see the distribution of distances covered on each visit
hist(visitStats$effortDiam)
# to see the distribution of species list lengths of each visit
hist(visitStats$SLL)
# to identify suspicious visits reported the first day of each month or year
hist(visitStats$day)
# to see correlations
plot(visitStats$nObs, visitStats$effortDiam)
plot(visitStats$SLL, visitStats$effortDiam)
# to see the ditributions of observations along the days of the month
plot(visitStats$day, visitStats$nObs)
}
```

`exportBirds` *Export single variables from SummarisedBirds objects*

Description

Export single variables from SummarisedBirds objects

Usage

```
exportBirds(x, dimension, timeRes, variable, method = "sum")
```

Arguments

`x`	a SummarizedBirds-object
`dimension`	a character string indicating if the export should be `"spatial"` or `"temporal"`
`timeRes`	A character string indicating what tempral resolution should be used. For Spatial export the function accepts `NULL`,`"yearly"`,`"monthly"` or `"month"`. For temporal export the function accepts `"yearly"`,`"monthly"`,`"daily"` or `"month"`.
`variable`	a character string indicating which variable should be exported, `"nObs"`,`"nVis"`,`"nSpp"`,`"nDays"`,`"nCel` or `"avgSll"` . For `timeRes` = `c(NULL,"month")` the function also accepts `"nYears"`.
`method`	Only applicable to `timeRes` = `"month"`. A variable specifying which statistical method should be applied. The function accepts `"sum"`,`"median"`,`"mean"`.

Value

an xts time series (if dimension = "temporal"), a named vector (if dimension = "temporal" and timeRes = "month"), or a SpatialPolygonsDataFrame (if dimension = "spatial")

Note

the difference between `timeRes = "monthly"` and `timeRes = "month"` is that the former returns "n.years x 12" values, while month summarize over years and returns only 12 values aplying the method among years. For more details over the possible combinations of dimensions and variables please refer to the vignette "Technical details".

Examples

```
grid <- makeGrid(searchPolygon, gridSize = 10)
SB <- summariseBirds(organizeBirds(bombusObsShort), grid=grid)
EB <- exportBirds(SB, "spatial", "month", "nDays", "median")
```

`exposeIgnorance` *Create ignorance scores*

Description

Ignorance scores are a proxy for the lack of sampling effort, computed by making the number of observations relative to a reference number of observations that is considered to be enough to reduce the ignorance score by half (henceforth the Half-ignorance approach). The algorithm behind the Ignorance Score is designed for comparison of bias and gaps in primary biodiversity data across taxonomy, time and space

Usage

```
exposeIgnorance(nObs, nSpp = NULL, h = 1)
```

Arguments

`nObs`	an object of any class (mainly resulting from `summariseBirds` or `exportBirds` with the number of observations, or visits in your desired analysis unit.
`nSpp`	the number of unique species observed
`h`	the half ignorance parameter value.

Value

a `data.frame` with ignorance scores

References

Ruete (2015) <doi:10.3897/BDJ.3.e5361>

See Also

`summarizeBirds`, `exportBirds`

Examples

```
OB <- organizeBirds(bombusObsShort, sppCol = "scientificName", simplifySppName = TRUE)
grid <- makeGrid(searchPolygon, gridSize = 10)
SB <- summariseBirds(OB, grid=grid)
ignorance <- exposeIgnorance(nObs=SB$spatial@data$nObs)
```

`focalSpReport` *Summarise all records for a species*

Description

This function will produce a simple visual report for the obsrevation pattern of the focal species. It shows grid cells with records on a map, and bar charts with number of records per year and month.

Usage

```
focalSpReport(
  x,
  focalSp = NULL,
  long = TRUE,
  colVis = "grey",
  colPres = "red",
  ...
)
```

Arguments

`x`	an object of class ‘SummarizeBirds’.
`focalSp`	the focal spp to look for.
`long`	whether the map should be long or wide.
`colVis`	color to plot visited gird cells
`colPres`	color to plot grid cells where species is present
`...`	further plot parameters

Value

a plot with a brief species summary

See Also

`summarizeBirds`, `exportBirds`

Examples

```
library(sp)
OB <- organizeBirds(bombusObsShort, sppCol = "scientificName", simplifySppName = TRUE)
grid <- makeGrid(searchPolygon, gridSize = 10)
SB <- summariseBirds(OB, grid=grid)
allSpp <- listSpecies(SB)
focal <- allSpp[2]
focalSpReport(SB, focalSp=focal)
```

`focalSpSummary` *Summarize records of a species among all visits*

Description

A function to summarise records of a species among all visits. Returns number of grid cells with occurrences, number of observations, number of visits, number of years and number of months with occurrences.

Usage

```
focalSpSummary(x, focalSp = NULL)
```

Arguments

`x`	an object of class 'SummarizeBirds'.
`focalSp`	the focal sp to look for.

Value

a `data.frame` with summary data for the focal species

See Also

`summarizeBirds`, `exportBirds`

Examples

```
OB <- organizeBirds(bombusObsShort, sppCol = "scientificName", simplifySppName = TRUE)
grid <- makeGrid(searchPolygon, gridSize = 10)
SB <- summariseBirds(OB, grid=grid)
allSpp <- listSpecies(SB)
focal<-"Zygodon viridissimus"
focalSpSummary(SB, focalSp=focal)
```

getGridIDs *Create unique IDs based on a grid*

Description

Takes a spatial points dataframe and a grid and gets the overlay IDs.

Usage

```
getGridIDs(x, grid)
```

Arguments

`x`	a SpatialPointsDataFrame .
`grid`	A a SpatialPolygon object (a grid is expected) defining the maximum extent of visits effort.

Value

A vector of the same length as the number of rows (observations) as x with a unique number corresponding to the grid's ID.

getUTMproj *A wrapper around getUTMzone and produce a proj4 string*

Description

A wrapper around getUTMzone and produce a proj4 string

Usage

```
getUTMproj(x)
```

Arguments

`x`	an object of class 'OrganizedBirds' or 'SpatialPointsDataFrame'

Value

a proj4 character string for an apropiate UTM zone

Examples

```
OB <- organizeBirds(bombusObs)
getUTMproj(OB)
```

`gotaland` *Götaland, Sweden.*

Description

A polygon describing the territorial contour the collective province of Götaland, Sweden. Includes holes for water bodies.

Usage

```
gotaland
```

Format

A SpatialPolygonsDataFrame with 1 polygon with CRS("+init=epsg:4326")

`gridAsString` *Convert a grid into a web query string.*

Description

Converts a grid (or any SpatialPolygon for that matter) into a web query string.

Usage

```
gridAsString(grid)
```

Arguments

`grid`	an object of class 'SpatialPolygon-class' or 'SpatialPolygonDataFrame-class'.

Value

a character string with coordinates separated by "%20" and pairs by ",".

listSpecies *Lists all species names from the data set*

Description

Lists all species names from the data set.

Usage

```
listSpecies(x)
```

Arguments

x	an object of class ‘OrganizedBirds’ or ‘SummarizeBirds’.

Value

a `vector` with all species names in the data set

See Also

`summarizeBirds`, `exportBirds`

Examples

```
OB <- organizeBirds(bryophytaObs, sppCol = "scientificName", simplifySppName = TRUE)
allSpp <- listSpecies(OB)
```

makeCircle *Create the minimum circle containing the points*

Description

This funciton is based on the function shotGroups::getMinCircle() that uses the Skyum algorithm based on the convex hull. http://www.cs.au.dk/~gerth/slides/sven14.pdf

Usage

```
makeCircle(spdf, projCRS = NULL)
```

Arguments

spdf	an object of class ‘SpatialPointsDataFrame’ with defined CRS.
projCRS	a proj4-type string defining a projected CRS. It is very important that the selected CRS is accurate in the study area. This is not the CRS for the argument ’spdf’ which should be defined internally. This is the CRS used to make a flat circle. Some UTM variant is recommended. See `getUTMproj`

Value

a polygon object of class 'SpatialPolygon' with geodesic coordinates in WGS84 (ESPG:4326).

See Also

`getUTMproj`

`makeDggrid` *Make a discrete global grid*

Description

Construct a discrete global grid system (dggs) object over a preferred polygon.

Usage

```
makeDggrid(
  polygon,
  gridSize,
  buffer = FALSE,
  topology = "hexagon",
  simplify = FALSE,
  tol = 0.01
)
```

Arguments

`polygon`	an object of class 'SpatialPolygon' or 'SpatialPolygonDataFrame'
`gridSize`	width of the cells in Km. It defines the central assumption of this package that is the maximum area a person can explore during a day. Be aware, that the spatial extent of a visit is dependent on the taxonomic group, and many other variables. Maximum recomended for this package 10 km if there is no reliable definition for the spatial extent for visits.
`buffer`	shall the grid cells include the polygon border? Then `TRUE` (default = `FALSE`).
`topology`	Shape of cell. shall the grid cells be hexagonal, diamonds or triangular? Options are: "hexagon", "diamond", "tirangle". Default: "hexagon".
`simplify`	simplifies the polygon geometry using the Douglas-Peuker algorithm (from rgeos package). Complicated polygons (those with much detail) make this function run slower.
`tol`	numerical tolerance value for the simplification algorith. Set to 0.01 as default.

Details

This function depends on a package that is no longer on CRAN. You can find it in its GitHub repository `https://github.com/r-barnes/dggridR`. Also, this may generate odd results for very large rectangles, because putting rectangles on spheres is weird... as you should know, if you're using this package. Use the function `exploreVisits()` to assess if your definition of visit aligns with your grid size.

Value

an object of class 'SpatialPolygon' with a set of polygons conforming to a grid of equal-area cells, with geodesic coordinates in WGS84 (ESPG:4326).

Note

Depending on the total number of grid cells the computations may take time. If there are more than 100 cells on any dimension a warning message will be displayed. Grid cells must be smaller than the sampling area. If the grid cell size is wider than the polygon on any dimension an error message will be displayed.

See Also

`drawPolygon`, `renameGrid`, `OB2Polygon`, `exploreVisits`

Examples

```
library(dggridR) ## Not anylonger on CRAN
grid <- makeDggrid(gotaland, gridSize = 10)
```

`makeGrid` *Make a grid*

Description

Makes a grid adapted to the purpose of this package and simplifing options from the `sp` package. The central concept of the BIRDS package is the definition of the field visit, and most likely, your grid size will define the maximum area a person can explore during a day. Use the function `exploreVisits()` to assess if your definition of visit aligns with your grid size.

Usage

```
makeGrid(
  polygon,
  gridSize,
  buffer = FALSE,
  hexGrid = TRUE,
  offset = NULL,
  simplify = FALSE,
  tol = 0.01
)
```

Arguments

`polygon`	an object of class 'SpatialPolygon' or 'SpatialPolygonDataFrame'
`gridSize`	width of the cells in Km. It defines the central assumption of this package that is the maximum area a person can explore during a day. Be aware, that the spatial extent of a visit is dependent on the taxonomic group, and many other variables. Maximum recomended for this package 10 km if there is no reliable definition for the spatial extent for visits.
`buffer`	shall the grid cells include the polygon border? Then `TRUE` (default = `FALSE`).
`hexGrid`	shall the grid cells be hexagonal? Then `TRUE` (default). Else squared grid cells.
`offset`	the offset (position) of the grid (from `spsample` methods). If it is left empty (`NULL`, default), then takes default values. For squared grid cells the default is set to `c(0.5,0.5)` ("centric systematic"). For hexagonal grid cells the default is set to `c(0,0)`.
`simplify`	simplifies the polygon geometry using the Douglas-Peuker algorithm (from rgeos package). Complicated polygons (those with much detail) make this function run slower.
`tol`	numerical tolerance value for the simplification algorith. Set to 0.01 as default.

Value

an object of class 'SpatialPolygon' with a set of polygons conforming to a grid of equal-area cells, with geodesic coordinates in WGS84 (ESPG:4326).

Note

Depending on the total number of grid cells the computations may take time. If there are more than 100 cells on any dimension a warning message will be displayed. Grid cells must be smaller than the sampling area. If the grid cell size is wider than the polygon on any dimension an error message will be displayed.

See Also

`drawPolygon`, `renameGrid`, `OB2Polygon`, `exploreVisits`

Examples

```
grid <- makeGrid(gotaland, gridSize = 10)
```

`OB2Polygon` *Create the polygon for the study area from a dataset of class ‘OrganizedBirds’*

Description

Create the polygon for the study area from a dataset of class ‘OrganizedBirds’

Usage

```
OB2Polygon(x, shape = "bBox")
```

Arguments

`x`	an object of class ‘OrganizedBirds’ or ‘SpatialPointsDataFrame’
`shape`	which type of polygon should be made from the data: • a bounding box (“bBox” or “bounding box”; i.e. the smallest bounding rectangle that contains all points), • a convex hull (“cHull” or “convex hull”; i.e. the smallest convex set that contains all the points). • the minimum circle (“minCircle” or “min circle”; i.e. the smallest circle that covers all the points).

Value

an object of class ‘SpatialPolygon’ with a polygon with geodesic coordinates in WGS84 (ESPG:4326).

Examples

```
orgDf <- organizeBirds(bombusObs)
polygon <- OB2Polygon(orgDf, shape = "cHull")
```

`obsData` *Extract observation data*

Description

Extract the observation data from a OrganizedBirds-object

Usage

```
obsData(x)

## S3 method for class 'OrganizedBirds'
obsData(x)
```

Arguments

`x`	An OrganizedBirds-object

Value

A dataframe

Examples

```
ob <- organizeBirds(bombusObs)
head(obsData(ob))
```

`obsIndex` *Observation Index*

Description

This function extracts the proportion of visits (or observations) detecting a focal species to all visits (or observations) over time or space.

Usage

```
obsIndex(
  x,
  dimension,
  timeRes = NULL,
  focalSp = NULL,
  visits = TRUE,
  fs.rm = TRUE,
  norm = TRUE
)
```

Arguments

`x`	an object of class ‘SummarizeBirds’.
`dimension`	a character string indicating if the export should be `"spatial"` or `"temporal"`
`timeRes`	the time resolution as a character string if `dimension = "temporal"`: `"Yearly"`,`"Monthly"` or `"Daily"`
`focalSp`	the focal species to look for
`visits`	if TRUE (default) the observation index is calculated over number of visits, else uses the number of observations
`fs.rm`	if TRUE, assumes that the observations for the focal species are included in ’group’ and will remove them
`norm`	if TRUE, the result is nomalized to a 0-1 range

Value

If `dimension = "spatial"` a 'SpatialPolygonsDataFrame' or a 'xts' timeseries if `dimension = "temporal"`.

Note

It implements the following algorithm to calculate the observation index: OI = log ((At / (At + Rt)) / (A / (A + R))) where At is the sum of observations of a focal species during time t (or gridcell), Rt is sum of observations of all species in reference group during t (or gridcell), A and R are the total sums for observations. If the ratio to log = 0 it adds 0.1 to avoid -Inf results.

References

Telfer, Preston 6 Rothery (2002) <doi:10.1016/S0006-3207(02)00050-2>

Examples

```
grid <- makeGrid(gotaland, gridSize = 10)
PBD <- bombusObsShort
OB <- organizeBirds(PBD, sppCol = "scientificName", simplifySppName = TRUE)
SB <- summariseBirds(OB, grid=grid)
spp <- listSpecies(SB)
tempOI <- obsIndex(SB, "temporal", "yearly", focalSp=spp[3], fs.rm = FALSE)
plot(tempOI$relObs, main=spp[3])
spatOI <- obsIndex(SB, "spatial", focalSp=spp[3])
minOI <- min(spatOI$relObs, na.rm=TRUE)
maxOI <- max(spatOI$relObs, na.rm=TRUE)
palRW <- leaflet::colorNumeric(c("white", "red"), c(minOI, maxOI), na.color = "transparent")
sp::plot(spatOI, col=palRW(spatOI$relObs), border="grey", main=spp[3])
legend("bottomleft", legend=seq(minOI, maxOI, length.out = 5),
       col = palRW(seq(minOI, maxOI, length.out = 5)), pch = 15, bty="n")
```

`organizeBirds` *Organize a dataframe to a usable format*

Description

Takes a dataframe with reported species observations and reformats it, using visit identifiers, to an OrganizedBirds-class that can be used in further analyses with the BIRDS-package.

Usage

```
organizeBirds(
  x,
  sppCol = "scientificName",
  idCols = c("locality", "recordedBy"),
  timeCols = c("year", "month", "day"),
  timeInVisits = "day",
  grid = NULL,
  presenceCol = NULL,
  xyCols = c("decimalLongitude", "decimalLatitude"),
  dataCRS = "+init=epsg:4326",
  taxonRankCol = NULL,
  taxonRank = c("SPECIES", "SUBSPECIES", "VARIETY"),
  simplifySppName = FALSE
)

organiseBirds(
  x,
  sppCol = "scientificName",
  idCols = c("locality", "recordedBy"),
  timeCols = c("year", "month", "day"),
  timeInVisits = "day",
  grid = NULL,
  presenceCol = NULL,
  xyCols = c("decimalLongitude", "decimalLatitude"),
  dataCRS = "+init=epsg:4326",
  taxonRankCol = NULL,
  taxonRank = c("SPECIES", "SUBSPECIES", "VARIETY"),
  simplifySppName = FALSE
)
```

Arguments

`x`	A dataframe or a SpatialPointsDataFrame containing at least a column for species name, one or several columns for date of observation, one or several columns for identifying a visit and, if it is not spatial, coordinate columns.
`sppCol`	A character string with the column name for the column for the species names. Default is the Darwin Core standard name `"scientificName"`.
`idCols`	A character vector of the names for the columns that are holding the information that identifies a visit. Default is the Darwin Core standard column names `c("locality","day","month","year","recordedBy")`.
`timeCols`	A character vector with the names for the column(s) holding the observation dates. Default is the Darwin Core standard column names `c("year","month","day")`.
`timeInVisits`	A flag indicating whether visits are defined by the time definition or not, and to which resolution. Default is 'day'. Alternatives are `c("day","month","year",NULL)`. Time is anyhow organised into three columns year, month, day.
`grid`	Either `NULL` to be ignored or an object of class `SpatialPolygons` or `SpatialPolygonsDataFrame` as indetifier of the visits spatial extent.
`presenceCol`	A character string with the column name for the column for the precense status. Default is `NULL`.
`xyCols`	A character vector of the names for the columns that are holding the coordinates for the observations. The order should be longitude(x), latitude(y). Default is the

	Darwin Core standard column names `c("decimalLongitude","decimalLatitude")`. Only applicable to non- spatial dataframes.
`dataCRS`	A character string for the dataframe CRS (Coordinate Reference System). Default is `"+init=epsg:4326"`, which is WGS 84. This is only applicable to non-spatial dataframes, since a spatial dataframes already should have this information.
`taxonRankCol`	the name of the column containing the taxonomic rank for the observation. That is the minimum taxonomic identification level.
`taxonRank`	a string or vector of strings containing the taxonomic ranks to keep. Only evaluated if taxonRankCol is not `NULL`
`simplifySppName`	wheter to remove everything else that is not the species name (authors, years and intraspecific epithets). Default set to FALSE

Details

An OrganizedBirds-class is essentially a list containing one element, a SpatialPointsDataFrame. This SpatialPointsDataFrame has its data formatted in a way that the other functions in the BIRDS-package can use further on. It also has the attribute `"visitCol"`, which indicates which column in the dataframe holds the visit identifier. The visit identifier is created by the function `createVisits`, which creates a unique id for each combination of the values in the defined columns.

The variable `timeCol` can be formatted differently. If the variable is a named vector with the names "Year", "Month" and "Day" (letter capitalization does not matter) it will use the variable named year as the year column and so on. Otherwise it will use the first variable as year, the second as month and the third as day, if there is a vector of length three or more. If the vector is of only length one it will interpret the column as a date column formatted as "yyyy-mm-dd".

Value

a ‘SpatialPointsDataFrame‘ wrapped into an object of class OrganizedBirds, with additional attributes.

See Also

`createVisits` to create unique visits IDs, `visits` to get or set the visit IDs to this class, `simplifySpp` to simplify species names, `obsData` to retrieve the dataframe from this class.

Examples

```
OB <- organizeBirds(bombusObs)
```

overlayBirds *Overlay BIRDS object and grid*

Description

Make an overlay for an OrganizedBirds object and a grid to identify which observations that fall into which grid cell.

Usage

```
overlayBirds(x, grid, spillOver = NULL)

## S3 method for class 'OrganizedBirds'
overlayBirds(x, grid, spillOver = NULL)
```

Arguments

`x`	An OrganizedBirds object
`grid`	A SpatialPolygonsDataFrame object of the grid over the study area
`spillOver`	Specifies if the function should search for observations with the same visitUID over several grid cell, and what to do with them. Default is `NULL`. It also accepts `c("unique","duplicate")`.

Details

This function takes an OrganizedBirds object, created by `organizeBirds`, and a polygon grid-layer, which could be created by `makeGrid` and splits the visits in the OrganizedBirds (i.e. data belonging to identified visits) to each grid cell.

If `spillOver = NULL` the splitting is done spatially according to the overlay of observations and grid cells, without further consideration of coherence for visits (visit UID). If `spillOver = c("unique","duplicate")` the splitting will be done spatially in a first step, and then: if `(spillOver = "unique")` assigns (and moves) all observations with same visitUID to the grid cell with most observations (or picks one grid cell at random if there is a tie); or if `(spillOver = "duplicate")` duplicates all observations with same visitUID across all grid cells containing at least one observation with that visitUID.

The later approach is usefull when the amount of observations spilled over neighbouring cells is minimal and information over the spatial extent of the sampling effort is more important than sample independence.

Value

The output is a OverlaidBirds-class object, which is a list containing three objects;

`observationsInGrid` Is basically the data in the OrganizedBirds object split by each grid cell (*n.b.* the use of `spillOver = TRUE` discussed under "Usage")

`grid` The SpatialPolygonsDataFrame from the input, but cleared of data to not waste unnecessary memory

`nonEmptyGridCells` An integer vector of which grid cells that have observations

Examples

```
ob <- organizeBirds(bombusObs)
grid <- makeGrid(gotaland, gridSize = 10)
ovB <- overlayBirds(ob, grid)
```

---

removeObs *Remove observations belonging to the shortest ("worst") visits*

Description

This function removes observations based on the visists effort or quality. Visit effort or quality could be given most often by species list length (that is, the number of species observed during the visit, SLL). However, in some cases there could be only one or few species observed but in great numbers each and spred across a big surveyed area. The effort then may not be small. If the user may find it necesary to remove those observations belonging to visits with an effort lower than a threshold, or a certain percentage of the "worst" observations, then this function will help.

Usage

```
removeObs(
  x,
  ev,
  criteria = "SLL",
  percent = 75,
  minCrit = NULL,
  stepChunk = 0.05
)
```

Arguments

`x`	an object of class 'OrganizedBirds' (organised BIRDS Spatial Dataframe). See `organizeBirds`
`ev`	an object of class 'data.frame' from exploreVistis.
`criteria`	the criteria to rank "good visits". Accepts c("SLL", "nObs", "effortDiam", "medianDist")
`percent`	the percentage (i.e. 0 - 100) of observation to keep, or NULL. (default = 75)
`minCrit`	the minimum accepted of a given criteria in the data set (default = NULL).
`stepChunk`	if the search for observations includes too many in a given quality stage, the search takes progressively smaller fractions of the dataset in steps. This argumente controls for how small fractions are discarded on each step. If stepChunk = 0.05 (default) means that in the first step 95 the observations will be tested, then 95 adequate number of observations are obtained. Increase this argument if you see the function takes too long.

Details

Please note: this function removes all observations belonging to visits that fullfil the criteria. Also, the percentage of "lower quality" visits in the sample is not necessarily the same as the the percentage of "lower quality" observations. The removal of observations is done stepwise by quantile therefore you may get a lower percentage than the aimed given than all remaining visists are too large to be included completely. This may happen particularly with samller datasets.

Value

An updated OrganisedBirds dataset

Note

If both 'percent' and 'minCrit' are defined then 'percent' prevails.

Examples

```
OB <- organizeBirds(bombusObs, sppCol = "scientificName", simplifySppName = TRUE)
EV <- exploreVisits(OB)
OBshorter <- removeObs(OB, EV, percent = 75)
```

SB *A simple summarisedBirds object.*

Description

A summarisedBirds object based on bombusObsShort for faster examples grid <- makeGrid(searchPolygon, gridSize = 10) SB <- summarizeBirds(organizeBirds(bombusObsShort), grid=grid)

Usage

```
SB
```

Format

An object of class summarisedBirds

`searchPolygon` *Searching polygon covering Götaland, Sweden.*

Description

An arbitrary polygon covering a portion of Sweden, used as search parameter.

Usage

```
searchPolygon
```

Format

A SpatialPolygonsDataFrame with 1 polygon with CRS("+init=epsg:4326")

`simpleSB` *A simple empty summarisedBirds object.*

Description

An empty summarisedBirds object used to dinamically test for validity of export parameter combinations `exportBirds()` in the sister package shinyBirds

Usage

```
simpleSB
```

Format

An object of class summarisedBirds

`simplifySpp` *Simplify species names*

Description

Removes infraspecific epithets, authors and years from scientific names

Usage

```
simplifySpp(df, sppCol)
```

Arguments

`df`	A dataframe with at least the column specified in sppCol
`sppCol`	A character vector with the column names for the species names.

Value

A vector vith data.frame with a scientific names up to specific names

---

`spatialVisits` *A function to make the exploreVisits Spatial*

Description

A function to

Usage

```
spatialVisits(
  x,
  xyCols = c("centroidX", "centroidY"),
  dataCRS = "+init=epsg:4326",
  radius = "medianDist"
)
```

Arguments

`x`	an object of class 'data.frame' from exploreVistis.
`xyCols`	a character vector with the column names for the coordinates. Default to `c("centroidX","centroidY")`
`dataCRS`	a character string with the proj4 description of the original coordinate projeciton system (CRS). Default to `"+init=epsg:4326"`
`radius`	either a character string with the name of the column containing the radius of the visit circle, or a numeric vector with its value in meters. Default to `"medianDist"`

Value

a list with a `SpatialPointsDataFrame` (the centroids) and a `"SpatialPolygonsDataFrame"` (the effort circles). Note that when plotted directly effort circles may not look like circles in the returned (Pseudo-Mercator) projection.

See Also

`exploreVisits`, `organiseBirds`

Examples

```
# create a visit-based data object from the original observation-based data
library(sp)
OB<-organizeBirds(bombusObsShort)
visitStats<-exploreVisits(OB)
spV<-spatialVisits(visitStats)
plot(spV$effort)
```

`speciesSummary` *Summarize all records for a species*

Description

A function that counts the number of observations, number of visits and number of grid cells with occurrences for all species.

Usage

```
speciesSummary(x)
```

Arguments

`x`	an object of class 'SummarizeBirds'.

Value

a `data.frame` with summary data for each species

See Also

`summarizeBirds`, `exportBirds`

Examples

```
grid <- makeGrid(searchPolygon, gridSize = 10)
SB <- summarizeBirds(organizeBirds(bombusObsShort), grid=grid)
summSB <- speciesSummary(SB)
```

`summarizeBirds` *Summarize the OrganizedBirds*

Description

Takes a OrganizedBirds-object and a SpatialPolygons*-grid and summarizes it in spatial and temporal dimensions.

Usage

```
summarizeBirds(x, grid, spillOver = NULL)

## S3 method for class 'OrganizedBirds'
summarizeBirds(x, grid, spillOver = NULL)

summariseBirds(x, grid, spillOver = NULL)
```

Arguments

`x`	An OrgnanizedBirds-object created by `organizeBirds`
`grid`	A SpatialPolygons or SpatialPolygonsDataFrame-object whith grid cells for the study area. This variable is optional and can be set to NULL, which will treat the area for all observations as one single grid cell.
`spillOver`	Specifies if the function should search for observations done during the same visit (identified by visit UID) but that fall outside the grid cell. Default is `NULL`. It also accepts `c("unique","duplicate")`. See Details for more information on how to use this.

Details

If `spillOver = NULL` the splitting is done spatially according to the overlay of observations and grid cells, without further consideration of coherence for visits (visit UID). If `spillOver = c("unique","duplicate")` the splitting will be done spatially in a first step, and then: if `(spillOver = "unique")` assigns (and moves) all observations with same visitUID to the grid cell with most observations (or picks one grid cell at random if there is a tie); or if `(spillOver = "duplicate")` duplicates all observations with same visitUID across all grid cells containing at least one observation with that visitUID.

The later approach is usefull when the amount of observations spilled over neighbouring cells is minimal and information over the spatial extent of the sampling effort is more important than sample independence.

Value

A SummarizedBirds-object

Examples

```
ob <- organizeBirds(bombusObsShort)
grid <- makeGrid(gotaland, gridSize = 10)
SB <- summarizeBirds(ob, grid)
nObsG <- rowSums(SB$spatioTemporal[,,13,"nObs"], na.rm = FALSE)
nObsG2 <- SB$spatial@data$nObs
any(nObsG != nObsG2, na.rm = TRUE) ## Check, two ways to obtain the same
```

---

visits *Get/set the visits*

Description

Gets or sets the visits identifier for a OrganizedBirds-class.

Usage

```
visits(x, name=NULL)

visits(x, name = NULL, useAsDefault = TRUE) <- value
```

Arguments

x	An OrganizedBirds-object
name	The name of the visit column. Default is NULL, which will get/write to the predefined visit column (visitUID).
useAsDefault	Specifies if the defined column in name should be used as the default column for the visits in further analysis. If name is NULL and useAsDefault = TRUE, value will be written to column (visitUID) and that column will be set to default.
value	the value to assign

Examples

```
ob <- organizeBirds(bombusObs)
attr(ob, "visitCol")
vis <- visits(ob)
tmp.vis <- createVisits(bombusObs, idCols=c("locality"), timeCols = c("day", "month", "year"))
visits(ob, name = "visNoRecorder", useAsDefault = TRUE) <- tmp.vis
vis2 <- visits(ob)
attr(ob, "visitCol")
```

Index

∗Topic **datasets**
    bombusObs, 4
    bombusObsShort, 5
    bryophytaObs, 5
    gotaland, 15
    SB, 27
    searchPolygon, 28
    simpleSB, 28

BIRDS, 3
bombusObs, 4, *5*
bombusObsShort, 5
bryophytaObs, 5

communityMatrix, 6
createVisits, 7, *9*, *24*

drawPolygon, 8, *18*, *19*

exploreVisits, 9, *18*, *19*, *29*
exportBirds, *6*, 10, *11–13*, *16*, *30*
exposeIgnorance, 11

focalSpReport, 12
focalSpSummary, 13

getGridIDs, 14
getUTMproj, 14, *16*, *17*
gotaland, 15
gridAsString, 15

listSpecies, 16

makeCircle, 16
makeDggrid, 17
makeGrid, 18, *25*

OB2Polygon, *18*, *19*, 20
obsData, 20, *24*
obsIndex, 21
organiseBirds, *9*, *29*
organiseBirds *(*organizeBirds*)*, 22
organizeBirds, *9*, 22, *25*, *26*, *31*
overlayBirds, 25

removeObs, 26
renameGrid, *18*, *19*

SB, 27
searchPolygon, 28
simpleSB, 28
simplifySpp, *24*, 28
spatialVisits, 29
speciesSummary, 30
summariseBirds *(*summarizeBirds*)*, 31
summarizeBirds, *6*, *11–13*, *16*, *30*, 31

visits, *24*, 32
visits<- *(*visits*)*, 32