Package 'rangeMapper'

September 13, 2012

Version 0.2-0

Date 2012-Sep-12

Title A platform for the study of macroecology of life history traits

Author Mihai Valcu <valcu@orn.mpg.de>, James Dale <J.Dale@massey.ac.nz>

Maintainer Mihai Valcu <valcu@orn.mpg.de>

Depends R (>= 2.14.0), methods, utils, RSQLite,RSQLite.extfuns, sp,rgdal, maptools, lattice, RColorBrewer, tcltk, tcltk2, pixmap,classInt

LinkingTo rgdal

Suggests raster, rgeos

SystemRequirements Bwidget, Tktable

Description A package to manipulate species range (extent-of-occurrence) maps, mainly tools for easy generation of biodiversity (species richness) or life-history traits maps.

License GPL (>= 2)

URL http://cran.at.r-project.org/package=rangeMapper ,http://rangemapper.r-forge.r-project.org/

Repository CRAN

Date/Publication 2012-09-13 11:34:09

R topics documented:

`bio.save` *Import 'BIO' tables to a* `rangeMapper` *project.*

Description

Import tables (e.g. life history data) to an active `rangeMapper` project.

Usage

```
bio.save(con, loc, tableName, ...)
metadata2bio(con, ...)
bio.merge(con, tableName, ...)
```

Arguments

`con`	An sqlite connection pointing to a valid `rangeMapper` project.
`loc`	file location or `data.frame` name
`tableName`	if missing, the name of the file or data.frame is used
`...`	Arguments to pass to the corresponding methods: e.g. the ID, the column corresponding to the names of the range files

Value

A 'BIO' table is created in the corresponding `rangeMapper` project.

Author(s)

Mihai Valcu <valcu@orn.mpg.de>

References

Valcu, M., Dale, J. and Kempenaers, B. (2012) rangeMapper: A platform for the study of macroecology of life history traits. 21(9). (DOI: 10.1111/j.1466-8238.2011.00739.x)

See Also

rangeMap.save. wrens

Examples

```
require(rangeMapper)
wd = setwd(tempdir())
r = readOGR(system.file(package = "rangeMapper", "extdata", "wrens", "vector_combined"), "wrens", verbose = FA
dbcon = rangeMap.start(file = "wrens.sqlite", overwrite = TRUE, dir = tempdir() )
global.bbox.save(con = dbcon, bbox = r)
gridSize.save(dbcon)
canvas.save(dbcon)
processRanges(spdf = r, con =  dbcon, ID = "sci_name" )

# Upload BIO tables
data(wrens)
Troglodytes  = wrens[grep("Troglodytes", wrens$sci_name), c(2, 5)]
bio.save(con = dbcon, loc = Troglodytes,  ID = "sci_name")
#wrensPath = system.file(package = "rangeMapper", "data", "wrens.csv")
#bio.save(con = dbcon, loc = wrensPath,  ID = "sci_name")
#bio.merge(dbcon, "wrensNew")
#metadata2bio(dbcon)

summary(rangeMap("wrens.sqlite"))$BIO_tables
setwd(wd)
```

---

canvas.save *Project's canvas*

Description

The canvas is a regular grid of a given resolution. Each range map is overlayed onto the canvas and the results saved to project.

Usage

```
canvas.save(con)
canvas.fetch(con)
```

Arguments

con	An sqlite connection pointing to a valid rangeMapper project.

Value

canvas.fetch Returns a SpatialPixelsDataFrame object.

Note

The method canvasSave() fails if grid.size was not set and if the canvas was already constructed for the given project.

Author(s)

Mihai Valcu <valcu@orn.mpg.de>

References

Valcu, M., Dale, J. and Kempenaers, B. (2012) rangeMapper: A platform for the study of macroecology of life history traits. 21(9). (DOI: 10.1111/j.1466-8238.2011.00739.x)

See Also

rangeMap.save.
gridSize.save

Examples

```
require(rangeMapper)
wd = tempdir()
dbcon = rangeMap.start(file = "test.sqlite", overwrite = TRUE, dir = wd)
global.bbox.save(con = dbcon)
gridSize.save(dbcon, gridSize = 2)
canvas.save(dbcon)
cnv = canvas.fetch(dbcon)
summary(cnv)
plot(cnv, col = 'grey', axes = TRUE)
```

---

global.bbox *Global bounding box*

Description

Computes, sets or retrieves the global spatial bounding box.

Usage

```
global.bbox.save(con, ...)
global.bbox.fetch(con)
```

Arguments

con	An SQLiteConnection object pointing to a rangeMapper project
...	Arguments to pass to the corresponding methods: *bbox* can be a character vector; the path to the range files directory *bbox* can also be an object inheriting from Spatial *p4s* an object of class CRS

Details

global.bbox.save saves the *global bounding box* and the *proj4* string to the sqlite database.
global.bbox.fetch retrieves the *global bounding box* as a SpatialPolygonsDataFrame.

Note

If *bbox* is a character vector then the corresponding method calls rangeMapBbox with checkProj = TRUE which requires all ranges to have the same *proj4* argument.
If *p4s* is set then the *bbox* will be set with that *p4s* string else the *p4s* will be identical with the *proj4* string of the range files.
If *bbox* and *p4s* are missing then an unprojected global bounding box is set.

Author(s)

Mihai Valcu <valcu@orn.mpg.de>

References

Valcu, M., Dale, J. and Kempenaers, B. (2012) rangeMapper: A platform for the study of macroecology of life history traits. 21(9). (DOI: 10.1111/j.1466-8238.2011.00739.x)

See Also

rangeMapper
proj4string
bbox

Examples

```
require(rangeMapper)
wd = tempdir()

f= system.file(package = "rangeMapper", "extdata", "wrens", "vector")

# Using default values for both bbox and p4s
dbcon = rangeMap.start(file = "test.sqlite", overwrite = TRUE, dir = wd )
global.bbox.save(con = dbcon)
bbox0 = global.bbox.fetch(dbcon)

plot(bbox0, axes = TRUE)
```

```
# more examples at: http://rangemapper.r-forge.r-project.org
```

`gridSize.save` *Save or retrieve the grid size from an* `rangeMapper` *project.*

Description

Save or retrieve the grid size from the active sqlite database.

Usage

```
gridSize.save(con, ...)
gridSize.fetch(con)
```

Arguments

`con`	A connection pointing to a valid `rangeMapper` project.
`...`	`gridSize`: A numeric vector of one unit length. See notes.

Value

`gridSize.fetch`	Returns a numeric vector of one unit length containing the grid size previously saved by `gridSize.save`

Note

If `gridSize` is not given the default grid size is computed based on the bounding box as the range of the smallest axis /100.

Author(s)

Mihai Valcu <valcu@orn.mpg.de>

References

Valcu, M., Dale, J. and Kempenaers, B. (2012) rangeMapper: A platform for the study of macroecology of life history traits. 21(9). (DOI: 10.1111/j.1466-8238.2011.00739.x)

See Also

`rangeMap.save`. `global.bbox`

Examples

```
require(rangeMapper)
wd = tempdir()
dbcon = rangeMap.start(file = "test.sqlite", overwrite = TRUE, dir = wd )
global.bbox.save(con = dbcon)
gridSize.save(dbcon, gridSize = 2)

dbcon = rangeMap.start(file = "test.sqlite", overwrite = TRUE, dir = wd )
global.bbox.save(con = dbcon)
gridSize.save(dbcon)
gridSize.fetch(dbcon) #default grid size value
```

---

metadata.update *Updates metadata table*

Description

Updates `metadata_table` of a `rangeMapper` project *after* importing ranges with `processRanges`.

Usage

```
metadata.update(rangeMap, FUN, name, map, overwrite = FALSE, ...)
```

Arguments

`rangeMap`	A `rangeMap` object.
`FUN`	Function used to aggregate the map values corresponding to each range
`name`	The name of the new `metadata_table` field containing the variable computed by `FUN`
`map`	Single-band `SpatialGridDataFrame` object
`overwrite`	If set to `TRUE` the the values of the field are replaced
`...`	extra arguments (e.g. `na.rm = TRUE`) to be passed to FUN.

Value

Nothing is returned.

Note

In order to compute taxa-level metadata which are not dependent on the project's resolution use `processRanges` with a `metadata` argument. See `rangeTraits` for more details.
The method can be extended to work with raster or vector objects (e.g. lines, polygons, points) using overlaying functions in the package `raster` and `rgeos` respectively.

Author(s)

Mihai Valcu <valcu@orn.mpg.de>

References

Valcu, M., Dale, J. and Kempenaers, B. (2012) rangeMapper: A platform for the study of macroecology of life history traits. 21(9). (DOI: 10.1111/j.1466-8238.2011.00739.x)

See Also

processRanges
rangeTraits

Examples

```
require(rangeMapper)

# data
spdf = readOGR(system.file(package = "rangeMapper", "extdata", "wrens", "vector_combined"), "wrens", verbose =
rloc = system.file(package = "rangeMapper", "extdata", "etopo1", "etopo1_Americas.tif")
r = readGDAL(rloc, output.dim = c(50, 50))
spdf = spTransform(spdf, CRS(proj4string(r)) )

# the project
dbcon = rangeMap.start(file = "wrens.sqlite", overwrite = TRUE, dir = tempdir() )
rmap = new("rangeMap", CON = dbcon)
global.bbox.save(con = dbcon, bbox = spdf )
gridSize.save(dbcon, gridSize = 300000)
canvas.save(dbcon)
processRanges(spdf = spdf, con =  dbcon, ID = "sci_name" )

# metadata.update
metadata.update (rmap,
FUN = function(x, ...) {
res = diff(range(x, ...))
if( !is.finite(res)) res = 0
res
},
name = 'AltitudeRange', map = r, na.rm = TRUE, overwrite = TRUE)

# plot
mr = RMQuery(dbcon, 'select * from metadata_ranges')
maxRangeSp = mr[mr$AltitudeRange== max(mr$AltitudeRange), 'bioid']
image(r)
plot(rangeFetch(rmap, maxRangeSp), add = TRUE, border = 4, lwd = 3)
title(main = maxRangeSp)
```

plot-methods *Plot a SpatialPixelsRangeMap*

Description

This is a wrapper around spplot

Arguments

palette	The name of a color palette. When NULL all the qualitative color palettes are returned.

Methods

signature(x = "SpatialPixelsRangeMap", y = "missing") **x=SpatialPixelsRangeMap**

```
plot(x, colorpalette = brewer.pal.get('Spectral')[11:1], ncols = 20, scales = FALSE, style = "e
```

x	a SpatialPixelsRangeMap object.
colorpalette	A color palette. See also tkColorPalette which allows for interactive choice of color palettes.
ncols	Number of color classes required, default to 20; argument to be passed to classIntervals
scales	If 'FALSE', default, axes scale are not drawn.
style	class interval style; see classIntervals for more details
...	Any argument that can be passed to spplot

Note

brewer.pal.get is a simple wrapper around brewer.pal.info

Author(s)

Mihai Valcu <valcu@orn.mpg.de>

See Also

classIntervals tkColorPalette brewer.pal

Examples

```
require(rangeMapper)
dbcon = rangeMap.start(file = "test.sqlite", overwrite = TRUE, dir = tempdir() )
f = system.file(package = "rangeMapper", "extdata", "wrens", "vector_combined")
global.bbox.save(con = dbcon, bbox = f,
p4s = CRS("+proj=cea +lon_0=0 +lat_ts=30 +x_0=0 +y_0=0 +ellps=WGS84 +units=km +no_defs") )
gridSize.save(dbcon, gridSize = 200)  # cell size 2 deg
canvas.save(dbcon)
processRanges(spdf = readOGR(f, "wrens", verbose = FALSE), con =  dbcon, ID = "sci_name")
rangeMap.save(dbcon) # species richness
```

```
# PLOTS
all = rangeMap.fetch(dbcon)
SR = rangeMap.fetch(dbcon, 'species_richness')

plot(all)
plot(SR, style = "fisher", sub = "Wrens species richness")

pal = brewer.pal.get('RdYlGn')[11:1]

plot(SR, style = "fisher", colorpalette = pal)

# if there is tcltk support choose a color palette interactively.
if(interactive()) {
tkColorPalette(pal = brewer.pal.get(),name  = "pal")
plot(SR, style = "fisher", colorpalette = pal)
}
```

`processRanges` *Process ranges*

Description

Each polygon range is overlayed on the canvas and the results are saved to the active project file.

Usage

```
processRanges(con,...)
```

Arguments

`con`	An sqlite connection pointing to a valid `rangeMapper` project.
`...`	Arguments to pass to the corresponding methods: *spdf* `SpatialPolygonsDataFrame` object containing all the ranges. *ID* when spdf is set this is a `character` vector given the name of the range. *dir* ranges file directory where the individual ranges shp files are located. In this case the range ID is the file name. *metadata* a named list of functions. See `rangeTraits` and `metadataUpdate` *parallel* for the moment no method defined for this signature.

Details

The overlay is performed using `overlay`. If the overlay returns no results (i.e. the polygon is smaller than a grid cell) then the centroid of the range will snap to the nearest point and only one grid cell will be returned for that range.

Note

If thousands of individual range map polygons are processed, their geometries are complex and/or the canvas resolution is relatively high this step can be time consuming.

Author(s)

Mihai Valcu <valcu@orn.mpg.de>

References

Valcu, M., Dale, J. and Kempenaers, B. (2012) rangeMapper: A platform for the study of macroecology of life history traits. 21(9). (DOI: 10.1111/j.1466-8238.2011.00739.x)

See Also

rangeMapper rangeTraits metadataUpdate.

Examples

```
require(rangeMapper)
wd = tempdir()

## Not run:
# Multiple files (one file per range)
rdr= system.file(package = "rangeMapper", "extdata", "wrens", "vector")
dbcon = rangeMap.start(file = "wrens.sqlite", overwrite = TRUE, dir = wd)
global.bbox.save(con = dbcon, bbox = rdr)
gridSize.save(dbcon) ; canvas.save(dbcon)
system.time(processRanges(dir = rdr, con =  dbcon))

## End(Not run)

# One file containing all the ranges
r = readOGR(system.file(package = "rangeMapper", "extdata", "wrens", "vector_combined"), "wrens", verbose = FA

dbcon = rangeMap.start(file = "wrens.sqlite", overwrite = TRUE, dir = wd )
global.bbox.save(con = dbcon, bbox = r)
gridSize.save(dbcon)
canvas.save(dbcon)

system.time(processRanges(spdf = r, con =  dbcon, ID = "sci_name" ))
# ~ 12 times faster than processing individual ranges.
```

rangeFetch *Range extractor*

Description

Fetch an arbitrary range from a rangeMapper project.

Usage

```
rangeFetch(rangeMap, bioid)
```

Arguments

rangeMap	A rangeMap object.
bioid	A character vector, usually a taxon name, which identifies a range within a given rangeMapper project.

Value

A SpatialPolygons.

Author(s)

Mihai Valcu <valcu@orn.mpg.de>

References

Valcu, M., Dale, J. and Kempenaers, B. (2012) rangeMapper: A platform for the study of macroecology of life history traits. 21(9). (DOI: 10.1111/j.1466-8238.2011.00739.x)

See Also

rangeMapper. rangeMapFetch. rangeMapSave.

Examples

```
wd = setwd(tempdir())
require(rangeMapper)
spdf = readOGR(system.file(package = "rangeMapper", "extdata", "wrens", "vector_combined"), "wrens", verbose =
dbcon = rangeMap.start(file = "wrens.sqlite", overwrite = TRUE, dir = tempdir() )
rmo = rangeMap("wrens.sqlite")
global.bbox.save(con = dbcon, bbox = spdf)
gridSize.save(dbcon, gridSize = 3)
canvas.save(dbcon)
processRanges(spdf = spdf, con =  dbcon, ID = "sci_name" )
rangeMap.save(dbcon)

house_wren = rangeFetch(rmo, "Troglodytes_aedon")
image(rangeMap.fetch(dbcon))
```

```
plot(house_wren, add = TRUE, border = 'blue', lwd = 2)
setwd(wd)
```

---

rangeMap-class *Class "rangeMap" to formally describe a rangeMapper sqlite file.*

Description

This is the basic class of the package.

Objects from the Class

Objects can be created by calls of the form `new("rangeMap", ...)`.

Slots

`CON`: Object of class `"SQLiteConnection"` pointing to a `rangeMapper` project

`VERSION`: Object of class `"character"` corresponding to the `rangeMapper` version

`ID`: Object of class `"character"`; the ID column corresponding to the names of the imported range files.

`BIOID`: Object of class `"character"` the ID column corresponding to the names of imported 'BIO' tables.

`PROJ4STRING`: Object of class `"character"` the proj4 string

`GRIDSIZE`: Object of class `"character"` grid size (in map units)

`BBOX`: Object of class `"character"` bounding box

`METADATA_RANGES`: Object of class `"character"` The name of a predefined table optionally constructed at the range import stage

`CANVAS`: Object of class `"character"` The name of the table hosting the project's regular grid (the canvas)

`RANGES`: Object of class `"character"` The name of the table hosting the ranges

`BIO`: Object of class `"character"` The prefix for 'BIO' tables

`MAP`: Object of class `"character"` The prefix for 'MAP' tables

Methods

canvasFetch `signature(object = "rangeMap")`: ...

canvasSave `signature(object = "rangeMap")`: ...

gridSizeFetch `signature(object = "rangeMap")`: ...

rangeMapBboxFetch `signature(object = "rangeMap")`: ...

rangeMapBboxSave signature(object = "rangeMap", bbox = "character", p4s = "CRS"):
...
rangeMapBboxSave signature(object = "rangeMap", bbox = "character", p4s = "missing"):
...
rangeMapBboxSave signature(object = "rangeMap", bbox = "missing", p4s = "CRS"):
...
rangeMapBboxSave signature(object = "rangeMap", bbox = "missing", p4s = "missing"):
...
rangeMapBboxSave signature(object = "rangeMap", bbox = "Spatial", p4s = "missing"):
...
rangeMapExport signature(object = "rangeMap", dirName = "character"): ...
rangeMapProcess signature(object = "rangeMap", spdf = "missing", dir = "character", ID = "missing",
...
rangeMapProcess signature(object = "rangeMap", spdf = "missing", dir = "character", ID = "missing",
...
rangeMapProcess signature(object = "rangeMap", spdf = "SpatialPolygonsDataFrame", dir = "missing",
...
rangeMapProcess signature(object = "rangeMap", spdf = "SpatialPolygonsDataFrame", dir = "missing",
...
summary signature(object = "rangeMap"): ...

Author(s)

Mihai Valcu, <valcu@orn.mpg.de>

See Also

rangeMapper, processRanges

---

rangeMap.export *Export 'MAP' tables*

Description

Export 'MAP' tables as single-band geotiff files

Usage

```
rangeMap.export(con, dirName, ...)
```

Arguments

con	An sqlite connection pointing to a valid rangeMapper project.
dirName	The directory name where the 'MAP's will be exported. If missing, the 'MAP's will be exported in project's directory
...	Further arguments to pass to writeGDAL

Author(s)

Mihai Valcu <valcu@orn.mpg.de>

References

Valcu, M., Dale, J. and Kempenaers, B. (2012) rangeMapper: A platform for the study of macroecology of life history traits. 21(9). (DOI: 10.1111/j.1466-8238.2011.00739.x)

---

rangeMap.save *Save, retrieve and export maps.*

Description

Apply a chosen SQL or R function at each grid cell, allowing for complex subsetting at both ID (e.g. species) and pixel (e..g assemblage) levels.

Usage

```
rangeMap.save(CON, tableName, FUN, biotab, biotrait, subset, path , overwrite, ...)
```

Arguments

CON	An sqlite connection pointing to a valid rangeMapper project.
tableName	Name of the table (quoted) to be added to the sqlite database. The prefix 'MAP' will be appended to tableName prior to saving.
FUN	the function to be applied to each pixel. If FUN is missing then species richness (species count) is computed.
biotab	character string identifying the 'BIO' table to use.
biotrait	character string identifying the ID of the 'BIO' table. see bio.save
subset	A named list. See details
path	Path to the raster file(quoted) to be imported to the existing project. raster package is required at this step.
overwrite	If TRUE then the table is removed
...	When FUN is an R function, ... denotes any extra arguments to be passed to it.

Details

The subset argument accepts a named list. Names refers to 'BIO', 'MAP' and 'metadata_rages' table names while the strings in the list are character strings containing the SQL WHERE clause. The subset can point to either one table type (e.g. list(MAP_species_richness = "species_richness > 500")) or can point to several table types (e.g. list(BIO_lifeHistory = "clutch_size > 4", MAP_meanAltitude = "meanAl))

Any valid SQL expression can be used to build up a subset. See http://www.sqlite.org/lang_expr.html

Value

TRUE when the MAP was created successfully. rangeMap.fetch returns a SpatialPixelsRangeMap-class.

Note

SQL aggregate functions are more eficient then their R counterparts. For simple aggregate functions like mean, median, sd, count it is advisable to use SQL functions rather then R functions.

Author(s)

Mihai Valcu <valcu@orn.mpg.de>

References

Valcu, M., Dale, J. and Kempenaers, B. (2012) rangeMapper: A platform for the study of macroecology of life history traits. 21(9). (DOI: 10.1111/j.1466-8238.2011.00739.x)

See Also

link{[RSQLite.extfuns-package] {RSQLite.extfuns}} for a list of additional SQL aggregate functions. metadataUpdate.

Examples

```
require(rangeMapper)
dbcon = rangeMap.start(file = "test.sqlite", overwrite = TRUE, dir = tempdir() )

# Breeding range vector files location
f = system.file(package = "rangeMapper", "extdata", "wrens", "vector_combined")

# Save the global bounding box,
global.bbox.save(con = dbcon, bbox = f,
p4s = CRS("+proj=cea +lon_0=0 +lat_ts=30 +x_0=0 +y_0=0 +ellps=WGS84 +units=m +no_defs") )

# upload grid size using the proposed default
gridSize.save(dbcon, gridSize = 100000)  # cell size ~ 100km

#  save canvas
canvas.save(dbcon)
summary(canvas.fetch(dbcon) )

# Upload BIO tables
data(wrens)
bio.save(con = dbcon, loc = wrens,  ID = "sci_name")

# Process species ranges
r = readOGR(f, "wrens", verbose = FALSE)

processRanges(spdf = r, con =  dbcon, ID = "sci_name" )
```

```
#Using sqlite aggregate functions
rangeMap.save(dbcon, FUN = "median" , biotab = "wrens",
biotrait = "body_size", tableName = "body_size")

# Fetch maps

summary(rangeMap.fetch(dbcon) )

# more examples at: http://rangemapper.r-forge.r-project.org
```

`rangeMap.start` *Initiate/open a new rangeMapper project*

Description

Initiate/open a new rangeMapper project using a `rangeMapStart-class` object

Usage

```
rangeMap.start(...)
rangeMap.open(path, verbose)
rangeMap(path)
```

Arguments

`path`	Character vector; a path to a valid rangeMapper project
`verbose`	Character vector; if `TRUE` the project's summary is printed
`...`	Arguments to be passed to `rangeMapStart-class`

Value

rangeMap.start() and rangeMap.open() returns an sqlite connection. rangeMap() returns a `rangeMap-class` object.

Author(s)

Mihai Valcu <valcu@orn.mpg.de>

See Also

`rangeMap.save`.
`rangeMapStart-class`

Examples

```
td = setwd(tempdir())

dbcon = rangeMap.start(file = "test.sqlite", overwrite = TRUE, dir = tempdir() )
summary(dbcon)

summary(rangeMap("test.sqlite"))

dbcon = rangeMap.open(path = "test.sqlite")
summary(dbcon)
setwd(td)
```

`rangeMapper` *rangeMapper: A platform for the study of macroecology of life history traits.*

Description

rangeMapper is a front end platform for the study of macroecology of life history traits at both inter-specific and assemblage levels.

Details

The package uses species range (extent-of-occurrence) vector polygons and life history traits datasets to build up maps (e.g. species richness, mean body mass, ...).
rangeMapper comes with an user-friendly platform-independent tcltk graphical user interface.

Getting Started

For a quick start open the graphical user interface (**gui**) by typing `rangeMapper()`. Mouse over the buttons to see further notes (tool-tips) regarding each step.
A tutorial can be accessed from Help/'Get started' under the **gui** or by browsing the 'doc' package directory.
See also the example below in the *examples* section on how to use **rangeMapper** from the command line.

The rangeMapper pipeline

- Initiate a new project (an empty sqlite database) using a pre-defined template.
- Extract/define the global bounding box of all selected range maps.
- Generate a regular grid (the canvas) using the previously constructed global bounding box.
- Perform polygon-grid overlay of all selected range maps.
- Optionally compute pre-defined or user-defined range structure indexes.
- Optionally import non-spatial data to be mapped at each grid cell (data are saved as a 'BIO' table to database).

- Optionally import georeferenced raster files.
- Compute a chosen statistical model at each grid cell optionally using complex subsets.
- Save, plot, export the MAP-s.

Author(s)

Mihai Valcu <valcu@orn.mpg.de>, http://orn.mpg.de/mitarbeiter/valcu.html
James Dale <J.Dale@massey.ac.nz>, http://quelea.net

References

Valcu, M., Dale, J. and Kempenaers, B. (2012) rangeMapper: A platform for the study of macroecology of life history traits. 21(9). (DOI: 10.1111/j.1466-8238.2011.00739.x)

See Also

rangeMap.save.

Examples

```
## Not run:
rangeMapper()

## End(Not run)
```

---

rangeMapStart-class *Class* "rangeMapStart"

Description

Methods of this class allows to initiate a new rangeMapper project.

Slots

dir: Object of class "character"; the directory of the sqlite database.

file: Object of class "character"; sqlite database filename, If missing a default name is assigned.

skeleton: Object of class "list" containing the sql code to initiate the sqlite database.

overwrite: Object of class "logical" when TRUE all the database tables are dropped.

Methods

rangeMapStart `signature(object = "rangeMapStart")`: ...

Author(s)

Mihai Valcu <valcu@orn.mpg.de>

See Also

rangeMapper
rangeMap.start

Examples

```
showClass("rangeMapStart")
str(new("rangeMapStart"))
```

---

rangeTraits *A container of functions to apply on a* SpatialPolygons object

---

Description

This is a convenience function returning a named list of functions.

Usage

```
rangeTraits(..., use.default)
```

Arguments

...	functions, given as myfun = FUN, to apply on a SpatialPolygons object
use.default	If TRUE, the default, the output list contains functions to extract Area, Median, Min and Max extent of the SpatialPolygons object. This option is ignored if no functions are given.

Details

The function returns a named list so any aditional functions should be given as rangeTraits(funName1 = FUN1, funName2 = FUN2) where FUN1, FUN2 are SpatialPolygons extractor functions.

Value

Returns a named list contaning extractor functions to apply on SpatialPolygons objects.

Author(s)

Mihai Valcu <valcu@orn.mpg.de>

See Also

processRanges rangeMapper.

Examples

```
summary(rangeTraits(use.default = F))

f = system.file(package = "rangeMapper", "extdata", "wrens", "vector")
troaed = selectShpFiles(f, ogr = TRUE, polygons.only = TRUE)[71, ] # path to Troglodytes_aedon

r = readOGR(troaed$dsn, troaed$layer)

# Beware of the value returned for Area!
sapply(rangeTraits(), function(x) x(r) )

# Define an extra function to compute correct Area
Area2 = function(x) {
 x = spTransform(x, CRS("+proj=cea +lon_0=0 +lat_ts=30 +x_0=0 +y_0=0 +ellps=WGS84 +units=m +no_defs") )
 sum(sapply(slot(x, "polygons"), function(x) slot(x, "area") ))
}

sapply(rangeTraits(Area_sqm = Area2), function(x) x(r) )
```

`rm.rangeMapper` *Remove tables from a give project*

Description

Remove tables given prefix attribute or by name

Usage

```
rm.rangeMapper(con, ...)
```

Arguments

con	A valid sqlite connection.
...	Arguments passed to the corresponding methods specifically 'tablePrefix' or 'tableName'

Note

The default 'rm.rangeMapper(con)' will remove all 'MAP' and 'BIO' tables.

Author(s)

Mihai Valcu <valcu@orn.mpg.de>

RMQuery *Query a* rangeMapper *project*

Description

For the moment a simple wrapper to sqliteQuickSQL.

Usage

```
RMQuery(con, statement)
```

Arguments

con	An sqlite connection
statement	An SQL string

Value

Returns either a data.frame if the statement is a select-like or NULL otherwise.

Author(s)

Mihai Valcu <valcu@orn.mpg.de>

See Also

sqliteQuickSQL.

selectShpFiles *Select (recursively) shape files*

Description

Returns the file path to all '.shp' polygons in a directory.

Usage

```
selectShpFiles(dir, ...)
```

Arguments

dir	character string specifying the directory containing .shp files.
...	currently ignored

Value

Either a data.frame or a character vector is returned.

Note

The function uses getinfo.shape to only select polygon files (aka type 5).

Author(s)

Mihai Valcu <valcu@orn.mpg.de>

References

Valcu, M., Dale, J. and Kempenaers, B. (2012) rangeMapper: A platform for the study of macroecology of life history traits. 21(9). (DOI: 10.1111/j.1466-8238.2011.00739.x)

See Also

rangeMap.save. getinfo.shape

Examples

```
f= system.file(package="rangeMapper", "extdata", "wrens", "vector")
res = selectShpFiles(f, ogr = TRUE, polygons.only = TRUE)
head(res)
```

SpatialPixelsRangeMap *Class "SpatialPixelsRangeMap"*

Description

A class extending "SpatialPixelsDataFrame" used for 'MAP's retrieval from a rangeMapper project

Slots

mapvar: Object of class "character"

Extends

Class "SpatialPixelsDataFrame", directly.

Methods

plot signature(x = "SpatialPixelsRangeMap", y = "missing"): ...

Note

For the moment mapvar is the only slot which makes this class distinct from the "SpatialPixelsDataFrame" class.

Author(s)

Mihai Valcu <valcu@orn.mpg.de>

See Also

rangeMap.fetch

---

tkColorPalette *Interactively choose a color palette.*

Description

Interactively choose a color palette. Palettes are created on the fly using tkbutton.

Usage

```
tkColorPalette(pal, name, palette.size = 45, envir = .GlobalEnv)
tkMakeColorPalette(n)
```

Arguments

pal	pal a named list as e.g. returned by brewer.pal.get.
name	A quoted name to be assigned to envir.
palette.size	palette.size is the size of the tiles (tkbutton-s) in pixels.
envir	the environment to use.
n	length of the user-defined palette returned by tkMakeColorPalette

Details

tkMakeColorPalette is called by tkColorPalette

Author(s)

Mihai Valcu <valcu@orn.mpg.de>

See Also

colorRampPalette brewer.pal.get

Examples

```
## Not run:
## These cannot be run by examples() but should be OK when pasted
## into an interactive R session
pals =  c(brewer.pal.get(), list(heatcol = heat.colors (9),terraincol = terrain.colors(9), topocol = topo.colo

tkColorPalette(pal = pals, name  = "pal1")
tkColorPalette(pal = pals, name  = "pal2", palette.size = 17)
pal1
pal2

## End(Not run)
```

---

tkdbBrowse *Browse an sqlite database.*

Description

Browse an sqlite database using a tcltk interface.

Usage

```
tkdbBrowse(con, prefix = NULL, tables.name.only = FALSE, info)
```

Arguments

con	A valid sqlite connection.
prefix	Shows only tables with the given prefix
tables.name.only	If 'FALSE', the default, shows only table names.
info	A character string containing a short text to be shown left to the OK button.

Author(s)

Mihai Valcu <valcu@orn.mpg.de>

Examples

```
## Not run:
## These cannot be run by examples() but should be OK when pasted
## into an interactive R session

# Create an empty rangeMapper project
dbcon = rangeMap.start(file = "wrens.sqlite",  overwrite = TRUE, dir = tempdir() )

# Select one table and then press OK
tkdbBrowse(dbcon, info = " select a field in one table and then push OK!")
```

```
tkdbBrowse(dbcon, tables.name.only = TRUE)


## End(Not run)
```

---

vertices-methods *Vertices of a SpatialPolygonsDataFrame*

Description

Extract vertices from a SpatialPolygonsDataFrame and optionally applies an aggregating function to each Polygon.

Value

A SpatialPointsDataFrame containing an id column corresponding to each extracted Polygon.

Methods

"SpatialPolygonsDataFrame", FUN = function Extract Polygon vertices and remove the last (repeated line) of each Polygon.

Author(s)

Mihai Valcu <valcu@orn.mpg.de>

References

Valcu, M., Dale, J. and Kempenaers, B. (2012) rangeMapper: A platform for the study of macroecology of life history traits. 21(9). (DOI: 10.1111/j.1466-8238.2011.00739.x)

See Also

coordinates.

Examples

```
require(rangeMapper)
f = system.file(package = "rangeMapper", "extdata", "wrens", "vector")
camgul = selectShpFiles(f, ogr = TRUE, polygons.only = TRUE)[6, ] # path to Campylorhynchus_gularis breeding r
r = readOGR(camgul$dsn, camgul$layer)
mp = vertices(r, mean)
v = vertices(r)

plot(r)
points(mp, col = 2, pch = 3, cex = 2)
points(v, pch = 3, cex = .5)
```

wrens *Life history data of the New World Wrens*

Description

Life history data (body size, body mass and clutch size) of 84 wren **(Troglodytidae)** species

Usage

```
data(wrens)
```

Format

A data frame with 84 observations on the following 7 variables.

`ID_HBW` Handbook of the birds of the world ID

`sci_name` scientific name; a factor with 84 levels

`com_name` English name; a factor with 84 levels

`genus` Genus name

`body_size` body size (cm)

`body_mass` body mass (grams)

`clutch_size` mean or modal clutch size

`source` bibliographic source of each trait (see references)

Details

Taxonomic nomenclature follows (Kroodsma & Brewer, 2005) with the exception of *Donacoblis atricapilla* which has been excluded due to its uncertain taxonomic position.

References

Auer, S.K., Logue, D.M., Bassar, R.D. & Gammon, D.E. (2007) Nesting biology of the Black-bellied Wren (Thryothorus fasciatoventris) in central Panama. Wilson Journal of Ornithology, 119, 71-76.

Dunning, J.B. (2008) CRC handbook of avian body masses, 2nd edn. CRC Press, Boca Raton.

Freeman, B.G. & Greeney, H.F. (2008) First description of the nest, eggs and cooperative breeding behavior in sharpe's wren (Cinnycerthia olivascens). Ornitologia Colombiana, 7, 88-92.

Hron, K., Templ, M. & Filzmoser, P. (2010) Imputation of missing values for compositional data using classical and robust methods. Computational Statistics & Data Analysis, 54, 3095-3107 (function impKNNa using default arguments).

Kroodsma, D.E. & Brewer, D. (2005) Family Troglodytidae (Wrens). Lynx Edicions, Barcelona, Spain.

Londono, G.A. (2009) Eggs, Nests, and Incubation Behavior of the Moustached Wren (Thryothorus genibarbis) in Manu National Park, Peru. Wilson Journal of Ornithology, 121, 623-627.

Ridgely, R.S., T. F. Allnutt, T. Brooks, D. K. McNicol, D. W. Mehlman, B. E. & Young, a.J.R.Z.

(2007) Digital Distribution Maps of the Birds of the Western Hemisphere, version 3.0. NatureServe, Arlington, Virginia, USA.
Vargas-Soriano, J., Ortiz, J.S. & Segura, G.E. (2010) Breeding Phenology and Nesting Success of the Yucatan Wren in the Yucatan Peninsula, Mexico. Wilson Journal of Ornithology, 122, 439-446.

See Also

rangeMap.save.

Examples

```
data(wrens)
plot(body_size ~ body_mass, wrens)
plot(clutch_size ~ log(body_mass), wrens)
```

Index