Package 'rangeMapper'

February 26, 2021

Version 2.0.2

Title A Platform for the Study of Macro-Ecology of Life History Traits

Depends R (>= 3.5.0)

Imports graphics, methods, glue, future, future.apply, progressr, sf, RSQLite, DBI, magrittr, data.table, raster, exactextractr

Suggests testthat, knitr, ggplot2, viridis, rmarkdown, pkgdown, nlme, igraph, spdep

Description Tools for generation of (life-history) traits and diversity maps on hexagonal or square grids. Valcu et al.(2012) <doi:10.1111/j.1466-8238.2011.00739.x>.

LazyData true

License GPL (>= 2)

URL https://github.com/mpio-be/rangeMapper

BugReports https://github.com/mpio-be/rangeMapper/issues

RoxygenNote 7.1.1

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation no

Author Mihai Valcu [aut, cre] (<https://orcid.org/0000-0002-6907-7802>), James Dale [aut] (<https://orcid.org/0000-0001-5950-5829>), Joan Maspons [ctb]

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

Repository CRAN

Date/Publication 2021-02-26 21:40:07 UTC

R topics documented:

---

dem *dem*

Description

Digital elevation model of the Americas based on ETOPO1.

Usage

```
data(dem)
```

Format

A RasterLayer with 159 rows and 212 columns.

References

Amante, C. and B. W. Eakins, ETOPO1 1 Arc-Minute Global Relief Model: Procedures, Data Sources and Analysis. NOAA Technical Memorandum NESDIS NGDC-24, 19 pp, March 2009. Go to this web site: http://www.ngdc.noaa.gov/mgg/global/global.html.

Examples

```
require(rangeMapper)
data(dem)
raster::plot(dem)
```

rmap_add_bio *Add non-spatial tables to a rangeMapper project*

Description

Add any dataset to the project. The dataset is saved in a separate table inside the project and labelled as a `bio` table.

Usage

```
rmap_add_bio(con, x, ID, name)

## S4 method for signature 'rmapConnection,data.table,character,character'
rmap_add_bio(con, x, ID, name)

## S4 method for signature 'rmapConnection,ANY,character,missing'
rmap_add_bio(con, x, ID, name)

## S4 method for signature 'rmapConnection,data.frame,character,character'
rmap_add_bio(con, x, ID, name)

## S4 method for signature 'rmapConnection,sf,character,character'
rmap_add_bio(con, x, ID, name)
```

Arguments

`con`	a rangeMapper connection made with `rmap_connect()` .
`x`	an object inheriting from `base::data.frame()`.
`ID`	character string. name of the ID column, usually species name.
`name`	output table name. If name is missing then name is the same as `x`.

Details

The `bio` tables contain the data which is then mapped with `rmap_save_map()` at each canvas cell and/or data used to create `subsets` with `rmap_save_subset()`. If the `bio` table inherits from sf then the geometry is silently dropped and only the non-spatial data are imported.

Value

TRUE when the table is written to the project file, FALSE otherwise.

Examples

```
con = rmap_connect()
wrens = read_wrens()
rmap_add_ranges(con, wrens, 'sci_name')
```

```
rmap_add_bio(con, wrens, 'sci_name')
dbDisconnect(con)
```

`rmap_add_ranges` *Add polygons to a rangeMapper project*

Description

Add polygon ranges (usually species or populations distribution ranges) to a rangeMapper project

Usage

```
rmap_add_ranges(con, x, ID)

## S4 method for signature 'rmapConnection,sf,character'
rmap_add_ranges(con, x, ID)
```

Arguments

`con`	a rangeMapper connection made with `rmap_connect()`.
`x`	a spatial polygon object of class `sf` (see `sf::st_as_sf()`) .
`ID`	character string. name of the ID column, usually species name.

Details

Polygons are saved as WKB (see `sf::st_as_binary()`).

Value

TRUE when the table is written to the project file, FALSE otherwise.

Examples

```
con = rmap_connect()
wrens = read_wrens()
rmap_add_ranges(con, x = wrens, ID = 'sci_name')
dbDisconnect(con)
```

`rmap_connect` *rangeMapper connect*

Description

Connect to a new or an existing rangeMapper project.

Usage

```
rmap_connect(path = ":memory:", overwrite = FALSE)
```

Arguments

`path`	filepath . When not specified, an in-memory file is created.
`overwrite`	when TRUE, the file is removed and the project re-initiated.

Details

An empty `rangeMapper` file is an sqlite database with five system tables:

- **rmap_nfo** containing the package version, the crs string, the canvas type and the bounding box.
- **rmap_master** a table similar with the in-build `sqlite_master` table holding information about the tables created or importing while working on the project.
- **canvas_ranges** a table that makes the link between the `canvas` and any entities usually species mapped on the canvas.
- **wkt_canvas** a table containing the canvas polygons as wkt binary.
- **wkt_ranges** a table containing the range polygons (usually species distribution ranges) as wkt binary.

If any of system tables is changed or missing then the file is considered corrupted and cannot be open with `rmap_connect()`.

Value

an object of class rmapConnection

Examples

```
require(rangeMapper)
con = rmap_connect()
class(con)
dbDisconnect(con)
```

rmap_prepare *Define a canvas and process ranges*

Description

rmap_prepare updates a 'raw' unprepared project to a ready to use project. rmap_prepare creates the project's canvas and assign each range to its corresponding canvas cells by performing a spatial intersection between the ranges and the canvas. The canvas is a regular grid of squares or hexagons.

Usage

```
rmap_prepare(con, grid_type, cellsize, chunksize, ...)

## S4 method for signature 'rmapConnection,character,numeric,missing'
rmap_prepare(con, grid_type, cellsize, chunksize, ...)

## S4 method for signature 'rmapConnection,character,numeric,numeric'
rmap_prepare(con, grid_type = "hex", cellsize, chunksize = 1/10, ...)
```

Arguments

con	a rangeMapper connection made with rmap_connect().
grid_type	character "hex" (default) or "square", see sf::st_make_grid().
cellsize	target cellsize, see sf::st_make_grid().
chunksize	parallel processing chunk size expressed as proportion from the range size. Default to 1/10.
...	further arguments passed to sf::st_make_grid()

Details

Because rmap_prepare can be potentially time consuming it can be run in parallel using the support provided by the future package. future allows parallel processing on a variety of systems including high performance computing environments. For details see future::plan(). Additionally, you can keep track of of the computations using progressr::handlers().

Value

TRUE when the table is written to the project file, FALSE otherwise.

References

Birch, C. P., Oom, S. P., & Beecham, J. A. (2007). Rectangular and hexagonal grids used for observation, experiment and simulation in ecology. Ecological modelling, 206(3-4), 347-359.

See Also

rmap_add_ranges()

Examples

```
# IN-MEMORY PROJECT
require(rangeMapper)
wrens = read_wrens()
con = rmap_connect()
rmap_add_ranges(con, wrens, 'sci_name')
rmap_prepare(con, 'hex', cellsize=500)
dbDisconnect(con)

## Not run:

# PROJECT PREPARED IN PARALLEL

require(future)
require(progressr)
plan(multisession, workers = 2)
handlers(global = TRUE)

Path = tempfile()
con = rmap_connect(Path)
rmap_add_ranges(con, wrens, 'sci_name')

rmap_prepare(con, 'hex', cellsize=200, chunksize = 0.1)

dbDisconnect(con)
plan(sequential)

## End(Not run)
```

`rmap_save_map` *Save maps*

Description

Maps are aggregate summaries computed for each canvas cell.

Usage

```
rmap_save_map(con, fun, src, v, subset, dst, ...)

## S4 method for signature
## 'rmapConnection,missing,missing,missing,missing,missing'
rmap_save_map(con)

## S4 method for signature
## 'rmapConnection,missing,missing,missing,character,character'
rmap_save_map(con, subset, dst)
```

```
## S4 method for signature
## 'rmapConnection,character,character,character,missing,character'
rmap_save_map(con, fun, src, v, dst)

## S4 method for signature
## 'rmapConnection,character,character,character,character,character'
rmap_save_map(con, fun, src, v, subset, dst)

## S4 method for signature
## 'rmapConnection,`function`,character,character,missing,character'
rmap_save_map(con, fun, src, v, subset, dst, ...)

## S4 method for signature
## 'rmapConnection,`function`,character,character,character,character'
rmap_save_map(con, fun, src, v, subset, dst, ...)

## S4 method for signature
## 'rmapConnection,`function`,character,ANY,missing,character'
rmap_save_map(con, fun, src, v, subset, dst, ...)

## S4 method for signature
## 'rmapConnection,`function`,character,ANY,character,character'
rmap_save_map(con, fun, src, v, subset, dst, ...)

## S4 method for signature
## 'rmapConnection,character,Raster,missing,missing,character'
rmap_save_map(con, fun, src, v, subset, dst, ...)
```

Arguments

con	a rangeMapper connection made with rmap_connect().
fun	the name of the function to save, either an SQLite or an R function. see Details.
src	the name of the source table previously imported by rmap_add_bio().
v	the variable to map or a function taking several variables as arguments. and returning one or several values.
subset	the name of a subset table. see rmap_save_subset.
dst	the name of the new map table.
...	arguments passed to fun.

Details

rmap_save_map makes maps based on data within the project or based on external raster objects. Aggregate functions can be:

- internal SQL aggregate functions: 'avg', 'count', 'max', 'min', 'sum', 'stdev', 'variance', 'mode', 'median', 'lower_quartile', 'upper_quartile', 'group_concat'.

- R functions taking one argument and returning one value.
- arbitrary statistical models applied on bio tables.

Value

TRUE when a table or a database view is written to the project file, FALSE otherwise.

Examples

```
require(rangeMapper)
require(data.table)
con = rmap_connect()
wrens = read_wrens()
rmap_add_ranges(con, x = wrens, ID = 'sci_name')
rmap_prepare(con, 'hex', cellsize=500)
rmap_save_map(con) # default is a species_richness map.

rmap_add_bio(con, wrens, 'sci_name')
rmap_save_map(con, fun='avg', src='wrens',v='body_mass', dst='avg_bodymass')

rmap_save_subset(con,dst ='ss1', species_richness = 'species_richness > 10')
rmap_save_map(con,subset = 'ss1', dst ='sr2')
rmap_save_map(con, fun='avg', src='wrens',v='body_mass',
 subset='ss1', dst='avg_bodymass_high_SR')

rmap_save_map(con, fun= mean, na.rm = TRUE, src='wrens',
 v='body_mass', dst='mean_bodymass')

Median = function(x) median(x,na.rm = TRUE)

rmap_save_map(con, fun = Median, src='wrens',
 v='body_mass', dst='median_bodymass')

rmap_save_map(con, fun= mean, na.rm = TRUE, src='wrens',v='body_mass',
 subset='ss1', dst='mean_bodymass_high_SR')

linmod = function(x) {
  lm(clutch_size ~ log(female_tarsus), x) %>%
  summary %>% coefficients %>% data.table %>% .[-1] }
rmap_save_map(con, fun= linmod, src='wrens', dst='slope_clutch_size')

data(dem)
rmap_save_map(con, fun= 'mean', src= dem ,  dst='dem', progress = FALSE)

x = rmap_to_sf(con)

dbDisconnect(con)
```

`rmap_save_subset` *Save subsets*

Description

`rmap_save_subset` creates subsets based on the canvas properties and/or the properties of one or several `bio` tables.

Usage

```
rmap_save_subset(con, dst, ...)

## S4 method for signature 'rmapConnection,character'
rmap_save_subset(con, dst, ...)
```

Arguments

`con`	a `rangeMapper` connection made with `rmap_connect()`.
`dst`	the name of the new subset table.
`...`	SQL `WHERE` calls, see Details.

Details

Subsets are defined using `table_name = "CONDITION"` where `CONDITION` can be any `SQL WHERE` call defined for the given table. Here is a summary of the SQL operators relevant in this context:

Operator	Description
= or == or `IS` or ! = or <> or IS NOT	Equals or Non-equals.
> or < or >= or <=	Greater (Less) than (or equal).
`IN` or NOT IN	multiple given values e.g. a IN (a,b,c,x,y).
`BETWEEN`	Between a given range (given values included) e.g. BETWEEN 1 and 10.
`LIKE`	Pattern search e.g. LIKE "%keyword%". `LIKE` is **case insensitive**.
`GLOB`	Similar to `LIKE` but uses the Unix wildcards (*,?,[]). e.g. [a-zA-Z0-9] matches any sing

Value

TRUE when the database view is written to the project file, FALSE otherwise.

Examples

```
require(rangeMapper)
con = rmap_connect()
wrens = read_wrens()
rmap_add_ranges(con, x = wrens, ID = 'sci_name')
rmap_prepare(con, 'hex', cellsize = 500)
rmap_add_bio(con, wrens, 'sci_name')
```

```
rmap_save_map(con)
rmap_save_subset(con,'s1',
   species_richness = 'species_richness > 10',
   wrens = 'body_mass > 19 AND clutch_size > 3')

dbDisconnect(con)
```

---

rmap_to_sf *Get* sf *data.frame-s.*

Description

Convert rangeMapper to sf.

Usage

```
rmap_to_sf(con, src, pattern)

## S4 method for signature 'rmapConnection,missing,missing'
rmap_to_sf(con)

## S4 method for signature 'rmapConnection,character,missing'
rmap_to_sf(con, src)

## S4 method for signature 'rmapConnection,missing,character'
rmap_to_sf(con, pattern)
```

Arguments

con	a rangeMapper connection made with rmap_connect().
src	the name of the source table. If missing all maps are returned.
pattern	a string that identifies several map names. It can be a regular expression.

Details

rmap_to_sf() retrieves one of the project's system tables: wkt_canvas, wkt_ranges or bbox or one or several map-s tables.

Value

an sf data.frame.

Examples

```
con = rmap_connect()
wrens = read_wrens()
rmap_add_ranges(con, x = wrens, ID = 'sci_name')
rmap_prepare(con, 'hex', cellsize = 500)
rmap_save_map(con) # default is a species_richness map.
rmap_save_subset(con, dst = 'ss1', species_richness = 'species_richness > 5')
rmap_save_map(con, subset = 'ss1', dst = 'species_richness_min5')
x = rmap_to_sf(con)
x = rmap_to_sf(con, 'species_richness_min5')

dbDisconnect(con)
```

---

st_thin *Thinning of polygonal grids*

Description

Nearest neighbours spatial thinning of polygonal grids

Usage

```
st_thin(x, lag)
```

Arguments

x	an sf data.frame.
lag	lag order.

Value

a thinned sf::st_as_sf() object.

Note

This function is still under development.

References

Based on SO answer: https://stackoverflow.com/questions/65907022/

Examples

```
## Not run:
require(rangeMapper)
con = rmap_connect()
wrens = read_wrens()
rmap_add_ranges(con, x = wrens, ID = 'sci_name')
rmap_prepare(con, 'hex', cellsize=500)
rmap_save_map(con)
x = rmap_to_sf(con)[, 'cell_id']

plot( st_thin(x,2) )

x = x[ ! x$cell_id  %in%  c(282,265) , ]

plot( st_thin(x,3) )


## End(Not run)
```

`wrens` *Wrens Life history.*

Description

Life history data of 84 wren species.

Usage

```
read_wrens()
```

Format

A GeoJSON file with with 84 entries and 12 variables. The variables are as follows:

- ID. Entry order as in ref. 1
- sci_name. Scientific name, character vector
- com_mame. English name, character vector
- subspecies. How many subspecies a species has.
- clutch_size. Mean or modal clutch size
- male_wing. Male wing length (mm)
- female_wing. Female wing length (mm)
- male_tarsus. Male tarsus length (mm)
- female_tarsus. Female tarsus length (mm)
- body_mass. Body mass (grams)
- data_src. bibliographic source of each trait given in the order they appear (see references)
- geometry. sfc simple feature geometry.

Note

The function read_wrens() reads the 'wrens.GeoJSON' data as a projected sf object.

References

BREEDING RANGES 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.

1. Brewer, David. Wrens, dippers and thrashers. Bloomsbury Publishing, 2010.
2. Kroodsma, D. E., and D. Brewer. "Family Troglodytidae (Wrens)." Lynx Edicions, Barcelona (2005).
3. Dunning Jr, John B. CRC handbook of avian body masses. CRC press, 2007.

Examples

```
require(rangeMapper)
require(sf)
wrens = system.file('extdata','wrens.GeoJSON',package = 'rangeMapper') %>% st_read

# or simpler
wrens = read_wrens()

plot(male_wing ~ female_wing, wrens)
plot(sf::st_geometry(wrens))
```

Index