Package: RWmisc (via r-universe)

November 4, 2024

Type Package

Title Miscellaneous Spatial Functions

Date 2022-02-14

Version 0.1.2

Description Contains convenience functions for working with spatial data across multiple UTM zones, raster-vector operations common in the analysis of conflict data, and converting degrees, minutes, and seconds latitude and longitude coordinates to decimal degrees.

Depends R (>= 3.4.0)

Imports sf, sp, raster, units

Suggests ggplot2, geosphere, lwgeom, microbenchmark, knitr, rmarkdown, testthat (>= 2.1.0), covr

License GPL (>=3)

Encoding UTF-8

URL https://github.com/jayrobwilliams/RWmisc

BugReports https://github.com/jayrobwilliams/RWmisc/issues

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

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

RemoteUrl https://github.com/jayrobwilliams/rwmisc

RemoteRef HEAD

RemoteSha 075ea87101ae617e557161ef64bd317ad75d8ab0

Contents

`dms2dd` *Convert from degrees, minutes, and seconds to decimal degrees*

Description

Convert latitudes and longitudes from degrees, minutes, and seconds to decimal degrees for conversion to spatial objects.

Usage

```
dms2dd(lon, lat)
```

Arguments

`lon`	a character vector of longitude coordinates in degrees, minutes, and seconds; see `details`
`lat`	a character vector of latitude coordinates in degrees, minutes, and seconds; see `details`

Details

`lon` and `lat` are expected to be in the format `"degrees° minutes' seconds" (direction)"` where `direction` is optional. If `direction` is not present, `dms2dd` will use negative signs (`-`) to determine positioning of coordinates.

Value

An $n * 2$ matrix where n is the length of `lon` and `lat`.

Examples

```
ll <- data.frame(lon = c("-122° 19' 55\"",
                         "71° 3' 32\" W"),
                 lat = c("47° 36' 22\"",
                         "42° 21' 36\" N"),
                 stringsAsFactors = FALSE)
dms2dd(ll[, 'lon'], ll[, 'lat'])
```

gadm.extract *Extract layers by country from GADM GeoPackage file*

Description

Extract one or more levels of administrative unit geometries from the GADM database in GeoPackage format

Usage

```
gadm.extract(input, output, countries = NULL, level = 0:5, ...)
```

Arguments

input	GeoPackage file to read from
output	name of file to save output to
countries	country or countries to limit results to, if NULL returns all countries
level	level(s) of administrative units 0:5 to extract; note not all levels are defined for all countries
...	additional arguments passed to sf::st_write()

Details

This function is designed to extract subsets of the Database of Global Administrative Areas (GADM). It uses the version of the database in GeoPackage format that provides one layer for each level of administrative division, available at https://gadm.org/download_world.html. The current version of this file is gadm36_levels.gpkg. It is intended for programmatic and reproducible subsetting of the database without requiring the user to individually download specific country data files.

Examples

```
## Not run:
## extract
gadm.extract("gadm36_levels.gpkg", "Nordics.gpkg",
             c("Denmark", "Finland", "Iceland", "Norway", "Sweden"),
             level = 0:2)

## add layers 3 and 4, use delete_layer = TRUE to rewrite existing layers
gadm.extract("gadm36_levels.gpkg", "Nordics.gpkg",
             c("Denmark", "Finland", "Iceland", "Norway", "Sweden"),
             level = 0:4, delete_layer = TRUE)

## End(Not run)
```

`overlap.weight` *Weight Raster Cells by Overlapping Polygons*

Description

Weight raster cells by overlapping polygons to avoid over-counting when aggregating by polygons

Usage

```
overlap.weight(raster, polygons, count = FALSE, warn = TRUE)
```

Arguments

`raster`	a RasterLayer object.
`polygons`	a SpatialPolygons, SpatialPolygonsDataFrame, or simple feature collection with at least two features. The function will still work with only one polygon, but values will be unchanged, and the result will be equivalent to `mask(raster, polygons)`.
`count`	a logical indicating whether to return a raster with the count of polygons intersecting each cell, or a raster with original values weighted by 1/number of intersecting polygons.
`warn`	include warnings? Most common is that the returned raster will be an intersection of the raster and the polygons. Default `TRUE`.

Details

This function takes a raster and a set of polygons as arguments. It counts the number of polygons that intersect each raster cell. It can return either a raster with the count of the number of intersecting polygons as cell values or the original raster with cell values weighted by 1 / the number of intersecting polygons (the default behavior). Cells that do not intersect any polygons will receive a value of `NA`. If the extent of the polygons is less than the extent of the raster, then the function will warn that it is cropping the raster to the polygons' extent.

Value

a RasterLayer object.

Examples

```
library(sf)
library(raster)
polys_t <- st_sfc(list(st_polygon(list(rbind(c(2,2), c(2,6),
                                             c(6,6), c(6,2),
                                             c(2, 2)))),
                       st_polygon(list(rbind(c(8,8), c(4,8),
                                             c(4,4), c(8,4),
                                             c(8,8))))),
```

```
                  crs = st_crs('OGC:CRS84'))
raster_t <- raster(nrows = 10, ncols = 10, xmn = 0,
                   xmx = 10, ymn = 0, ymx = 10,
                   vals = 1:100,
                   crs = CRS(st_crs(polys_t)$proj4string))
overlap.weight(raster_t, polys_t)
```

`point.poly.dist` *Point-Polygon Distances*

Description

Calculate the maximum or minimum possible distance from a point to the edge of a given polygon.

Usage

```
point.poly.dist(point, poly, max = TRUE, by_element = FALSE)
```

Arguments

`point`	A simplefeatures object of class point.
`poly`	A simplefeatures object of class polygon or multipolygon.
`max`	Logical; return maximum or minimum distance? default `TRUE`
`by_element`	Logical; return total maximum or minimum, or for each input point? default `FALSE`

Value

Maximum or minimum distance between a point and a polygon.

Examples

```
library(sf)
polys <- st_sfc(st_polygon(list(rbind(c(0,0), c(0,1), c(1,1), c(1,0), c(0,0)))),
crs = st_crs('OGC:CRS84'))
points <- st_sfc(st_multipoint(rbind(c(.25, .5), c(.75, .5), c(.5, .5))),
                 crs = st_crs('OGC:CRS84'))
point.poly.dist(points, polys)
```

projectUTM *Project to UTM*

Description

Project an object in latitude/longitude to UTM.

Usage

```
projectUTM(x)

## S3 method for class 'sf'
projectUTM(x)

## S3 method for class 'sfc'
projectUTM(x)

## S3 method for class 'SpatialPointsDataFrame'
projectUTM(x)

## S3 method for class 'SpatialPoints'
projectUTM(x)

## S3 method for class 'SpatialPolygonsDataFrame'
projectUTM(x)

## S3 method for class 'SpatialPolygons'
projectUTM(x)
```

Arguments

x	An sf or sp object in latitude-longitude CRS.

Value

An sf or sp object projected to UTM CRS.

Examples

```
library(sf)
nc <- st_read(system.file("shape/nc.shp", package="sf"))
st_crs(projectUTM(nc))
```

theme_rw *Blank Theme*

Description

A ggplot theme with no grid elements or gray background.

Usage

```
theme_rw()
```

Value

A ggplot theme object.

Examples

```
ggplot2::ggplot(mtcars, ggplot2::aes(x = hp, y = mpg)) +
ggplot2::geom_point() +
theme_rw()
```

UTM.functions *UTM Convenience Functions*

Description

Functions for converting latitude-longitude data to UTM.

Usage

```
long2UTM(long)

UTMzones(long)

chooseUTM(long)
```

Arguments

long	A vector of longitude values.

Value

UTM vector of zone numbers.

UTM vector of zone numbers.

UTM zone number.

Examples

```
long2UTM(c(-90, 0, 90))
UTMzones(c(-90, 90, 90))
chooseUTM(c(-90, -80, -70))
```

Index