Package ‘extRatum’

October 13, 2022

Title Summary Statistics for Geospatial Features

Version 1.0.4

Description Provides summary statistics of local geospatial features within a given geographic area. It does so by calculating the area covered by a target geospatial feature (i.e. buildings, parks, lakes, etc.). The geospatial features can be of any type of geospatial data, including point, polygon or line data.

License MIT + file LICENSE

Depends R (>= 3.3.0)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

SystemRequirements C++11, GDAL (>= 2.0.1), GEOS (>= 3.4.0), PROJ (>= 4.8.0)

Date/Publication 2021-01-18 16:50:12 UTC

Imports sf (>= 0.9.5), dplyr (>= 1.0.0), tidyr (>= 1.1.0)

NeedsCompilation no

Author Nikos Patias [aut, cre] (<https://orcid.org/0000-0002-6542-2330>), Francisco Rowe [aut]

Maintainer Nikos Patias <n.patias@liverpool.ac.uk>

Repository CRAN

R topics documented:

areal_calc ................................................................. 2
lines ............................................................................ 3
line_calc ................................................................. 3
points ........................................................................... 4
point_calc ................................................................. 5
pol_large ................................................................. 7
pol_small ................................................................. 7

Index 8
**areal_calc**

_Areal data calculation_

**Description**
Computes three different summary statistics: (1) _TotalArea_ total area of each polygon; (2) _AreaCovered_ area covered by a multipolygon object within a high order polygon; and, (3) _Ratio_ ratio between _AreaCovered_ and _TotalArea_ i.e. ratio between an area covered by a given set of features and total area of a higher-order geography polygon.

**Usage**
```
areal_calc(polygon_layer, higher_geo_lay, unique_id_code, crs)
```

**Arguments**
- **polygon_layer**: multipolygon object of class `sf`, `sfc` or `sfg`.
- **higher_geo_lay**: multipolygon object of class `sf`, `sfc` or `sfg`.
- **unique_id_code**: a string; indicating a unique ID column of `higher_geo_lay`, used as the summary areas.
- **crs**: coordinate reference system: integer with the EPSG code, or character based on `proj4string`.

**Details**
The function requires two sets of polygon data: high-order and low-order geographic polygons

**Value**
a tibble data frame object containing four columns is returned:
- the `unique_id_code` of `higher_geo_lay`
- the total area of each polygon in `higher_geo_lay` (_TotalArea_)
- the total area covered by `polygon_layer` features (_AreaCovered_)
- the ratio between the total area covered by `polygon_layer` and total area of `higher_geo_lay` polygon (_Ratio_)

**Examples**
```r
## Run areal_calc() using the packages' dummy data sets.
## The data sets are georeferenced on wgs84. However, a planar system is used to measure areas.
## For the examples provided here, points and polygons relate to the United Kingdom.
## So the British National Grid is used.

## Not run:
#outcome <- areal_calc(polygon_layer = pol_small, 
#higher_geo_lay = pol_large, 
```
lines

Description
Toy dataset of line data.

Usage
lines

Format
A geospatial file of six lines georeferenced in wgs84.

Source
Own dataset.

line_calc

Description
Computes three different summary statistics: (1) TotalArea total area of each polygon; (2) TotalLength total length of a multilinestring object within a polygon (3) Ratio ratio between TotalLength and TotalArea i.e. the ratio between the total length and total area of a higher-order geography polygon.

Usage
line_calc(line_layer, higher_geo_lay, unique_id_code, crs)

Arguments
line_layer multilinestring object of class sf, sfc or sfg.
higher_geo_lay multipolygon object of class sf, sfc or sfg.
unique_id_code a string; indicating a unique ID column of higher_geo_lay, used as the summary areas.
crs coordinate reference system: integer with the EPSG code, or character based on proj4string.
Value

A tibble data frame object containing four columns:
- the unique_id_code of higher_geo_lay
- the total area of each polygon in higher_geo_lay (TotalArea)
- the total length of line_layer features (TotalLength)
- the ratio between the total length of line_layer and the total area of higher_geo_lay polygon (Ratio).

Examples

```r
## Run line_calc() using the packages' dummy data sets.
## The data sets are georeferenced on wgs84. However, a planar system is used to measure areas.
## For the examples provided here, points and polygons relate to the United Kingdom.
## So the British National Grid is used.

## Not run:
outcome <- line_calc(
  line_layer = lines,
  higher_geo_lay = pol_large,
  unique_id_code = "large_pol_",
  crs = "epsg:27700")
## End(Not run)
```

points

- **Point geospatial layer.**

Description

Toy dataset of point data.

Usage

points

Format

A geospatial file of ten points georeferenced in wgs84.

Source

Own dataset.
**point_calc**

**Point data calculation**

**Description**

Computes three different summary statistics: (1) TotalArea total area of each polygon; (2) NoPoints number of multipoint objects within a given polygon; and, (3) Ratio ratio between NoPoints and TotalArea covered within a polygon.

**Usage**

```r
point_calc(
  point_data,  # multipoint object of class sf, sfc or sfg.
  higher_geo_lay,  # multipolygon object of class sf, sfc or sfg.
  unique_id_code,  # a string; indicating a unique ID column of higher_geo_lay, used as the summary areas.
  class_col,  # a string; indicating a column name for point_data containing information on a target point classification. This is used when total_points = FALSE.
  crs,  # coordinate reference system: integer with the EPSG code, or character based on proj4string.
  total_points = TRUE  # logical; if the target is to measure the total number of points set to TRUE, by setting to FALSE, it returns the total number of points by class. If missing, it defaults to TRUE.
)
```

**Arguments**

- `point_data` multipoint object of class sf, sfc or sfg.
- `higher_geo_lay` multipolygon object of class sf, sfc or sfg.
- `unique_id_code` a string; indicating a unique ID column of higher_geo_lay, used as the summary areas.
- `class_col` a string; indicating a column name for point_data containing information on a target point classification. This is used when total_points = FALSE.
- `crs` coordinate reference system: integer with the EPSG code, or character based on proj4string.
- `total_points` logical; if the target is to measure the total number of points set to TRUE, by setting to FALSE, it returns the total number of points by class. If missing, it defaults to TRUE.

**Details**

The function requires two sets of data: a layer of geographic polygons, and a layer of points.

If points have been categorised into classes, the function can return the same summary measures for each class if total_points = FALSE by specifying the column that contains the classification in class_col.
Value

if total_points = TRUE: A tibble data frame objects containing four columns is returned:

- the unique_id_code of higher_geo_lay
- the total area of each polygon in higher_geo_lay (TotalArea)
- the total number of point features point_data (NoPoints), and
- the ratio between the total number of point features point_data and the the total area of higher_geo_lay polygon (Ratio).

if total_points = FALSE: A list of three tibble data frame objects is returned.

- The object PointsLong contains three columns: the unique_id_code of higher_geo_lay, the class_col of point_data, the number of point features point_data by class (NoPoints), the total area of each polygon in higher_geo_lay (TotalArea) and the ratio between the number of point features by class point_data and the the total area of higher_geo_lay polygon (Ratio).
- The object PointsCountWide: Returns the point counts of PointsLong by unique_id_code and class_col in a wide format.
- The object PointsRatioWide: Returns the ratio of PointsLong by unique_id_code and class_col in a wide format.

Examples

# Run point_calc() using the packages' dummy data sets.
# The data sets are georeferenced on wgs84. However, a planar system is used to measure areas.
# For the examples provided here, points and polygons relate to the United Kingdom.
# So the British National Grid is used.

# Not run:
# This example returns the total points count and ratio
# outcome1 <- point_calc(
# point_data = points,
# higher_geo_lay = pol_large,
# unique_id_code = "large_pol",
# crs = "epsg:27700",
# total_points = TRUE)

# This example returns the points count and ratio by class
# outcome2 <- point_calc(
# point_data = points,
# higher_geo_lay = pol_large,
# unique_id_code = "large_pol",
# class_col = "class_name",
# crs = "epsg:27700",
# total_points = FALSE)
# End(Not run)
**pol_large**

*Large polygons geospatial layer.*

**Description**

Toy dataset of polygon data.

**Usage**

`pol_large`

**Format**

A geospatial file of three polygons georeferenced in wgs84.

**Source**

Own dataset.

---

**pol_small**

*Small polygons geospatial layer.*

**Description**

Toy dataset of polygon data.

**Usage**

`pol_small`

**Format**

A geospatial file of eight polygons georeferenced in wgs84.

**Source**

Own dataset.
Index

* datasets
  lines, 3
  points, 4
  pol_large, 7
  pol_small, 7

areal_calc, 2

line_calc, 3
lines, 3

point_calc, 5
points, 4
pol_large, 7
pol_small, 7