Package: SUNGEO (via r-universe)

September 11, 2024

Type Package

Title Sub-National Geospatial Data Archive: Geoprocessing Toolkit

Version 1.3.0

Date 2024-05-14

Author Yuri M. Zhukov, Jason Byers, Marty Davidson

Maintainer Yuri M. Zhukov <zhukov@umich.edu>

Description Tools for integrating spatially-misaligned GIS datasets.
Part of the Sub-National Geospatial Data Archive System.

URL https://www.sungeo.org/, https://github.com/zhukovyuri/SUNGEO

License GPL-2

Encoding UTF-8

LazyData TRUE

Depends R (>= 2.10)

Imports
sf,data.table,dplyr,RCurl,jsonlite,terra,raster,stringr,stats,methods,purrr,measurements,RANN,cartogram,packcircles,rmap

RoxygenNote 7.3.1

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

RemoteUrl https://github.com/zhukovyuri/sungeo

RemoteRef HEAD

RemoteSha e8601d291247fb197d88243fde41407d1689380f

Contents

`available_data` *Data availability through SUNGEO API*

Description

Census of geospatial and processed data files available to download using `SUNGEO::get_data()`.

Usage

```
available_data
```

Format

List of 42 data.table objects Geoset:GADM :Classes 'data.table' and 'data.frame': 249 obs. of 4 variables Geoset:GAUL :Classes 'data.table' and 'data.frame': 242 obs. of 4 variables Geoset:geoBoundaries :Classes 'data.table' and 'data.frame': 197 obs. of 4 variables Geoset:GRED :Classes 'data.table' and 'data.frame': 74 obs. of 4 variables Geoset:HEXGRID :Classes 'data.table' and 'data.frame': 199 obs. of 4 variables Geoset:MPIDR :Classes 'data.table' and 'data.frame': 52 obs. of 4 variables Geoset:NHGIS :Classes 'data.table' and 'data.frame': 1 obs. of 4 variables Geoset:PRIOGRID :Classes 'data.table' and 'data.frame': 199 obs. of 4 variables Geoset:SHGIS :Classes 'data.table' and 'data.frame': 68 obs. of 4 variables

country_iso3 Codes for available countries (ISO 3166-1 alpha-3). Character string.

country_name Names of available countries. Character string.

geoset_years Years of available historical boundary files. Character string.

space_units Available spatial units of analysis. Character string.

Elections:LowerHouse:CLEA :Classes 'data.table' and 'data.frame': 168 obs. of 6 variables Demographics:Ethnicity:EPR :Classes 'data.table' and 'data.frame': 180 obs. of 6 variables Demographics:Ethnicity:GREG :Classes 'data.table' and 'data.frame': 234 obs. of 6 variables Demographics:Population:GHS :Classes 'data.table' and 'data.frame': 257 obs. of 6 variables Events:PoliticalViolence:ABADarfur :Classes 'data.table' and 'data.frame': 1 obs. of 6 variables Events:PoliticalViolence:ACLED :Classes 'data.table' and 'data.frame': 100 obs. of 6 variables Events:PoliticalViolence:BeissingerProtest :Classes 'data.table' and 'data.frame': 15 obs. of 6 variables Events:PoliticalViolence:BeissingerRiot :Classes 'data.table' and 'data.frame': 15 obs. of 6 variables Events:PoliticalViolence:BeissingerUkraine :Classes 'data.table' and 'data.frame': 1 obs. of 6 variables Events:PoliticalViolence:COCACW :Classes 'data.table' and 'data.frame': 1 obs. of 6 variables Events:PoliticalViolence:ESOCAfghanistanWITS :Classes 'data.table' and 'data.frame': 1 obs. of 6 variables Events:PoliticalViolence:ESOCIraqSIGACT :Classes 'data.table' and 'data.frame': 1 obs. of 6 variables Events:PoliticalViolence:ESOCIraqWITS :Classes 'data.table' and 'data.frame': 1 obs. of 6 variables Events:PoliticalViolence:ESOCMexicoDrugRelatedMurders :Classes 'data.table' and 'data.frame': 1 obs. of 6 variables Events:PoliticalViolence:ESOCMexicoHomicide :Classes 'data.table' and 'data.frame': 1 obs. of 6 variables Events:PoliticalViolence:ESOCPakistanBFRS :Classes 'data.table' and 'data.frame': 1 obs. of 6 variables Events:PoliticalViolence:ESOCPakistanWITS :Classes 'data.table' and 'data.frame': 1 obs. of 6 variables Events:PoliticalViolence:GED :Classes 'data.table' and 'data.frame': 121 obs. of 6 variables Events:PoliticalViolence:Lankina :Classes 'data.table' and 'data.frame': 1 obs. of 6 variables Events:PoliticalViolence:NIRI :Classes 'data.table' and 'data.frame': 12 obs. of 6 variables Events:PoliticalViolence:NVMS :Classes 'data.table' and 'data.frame': 1 obs. of 6 variables Events:PoliticalViolence:PITF :Classes 'data.table' and 'data.frame': 133 obs. of 6 variables Events:PoliticalViolence:SCAD :Classes 'data.table' and 'data.frame': 60 obs. of 6 variables Events:PoliticalViolence:yzCaucasus2000 :Classes 'data.table' and 'data.frame': 1 obs. of 6 variables Events:PoliticalViolence:yzChechnya :Classes 'data.table' and 'data.frame': 1 obs. of 6 variables Events:PoliticalViolence:yzLibya :Classes 'data.table' and 'data.frame': 1 obs. of 6 variables Events:PoliticalViolence:yzUkraine2014 :Classes 'data.table' and 'data.frame': 1 obs. of 6 variables Infrastructure:Roads:gRoads :Classes 'data.table' and 'data.frame': 240 obs. of 6 variables Infrastructure:NightLights:DMSP :Classes 'data.table' and 'data.frame': 257 obs. of 6 variables PublicHealth:Covid19:JHUCSSEC19 :Classes 'data.table' and 'data.frame': 207 obs. of 6 variables Terrain:Elevation:ETOPO1 :Classes 'data.table' and 'data.frame': 256 obs. of 6 variables Terrain:LandCover:GLCC :Classes 'data.table' and 'data.frame': 257 obs. of 6 variables Weather:AirTemperatureAndPrecipitation:NOAA :Classes 'data.table' and 'data.frame': 209 obs. of 6 variables

country_iso3 Codes for available countries (ISO 3166-1 alpha-3). Character string.

country_name Names of available countries. Character string.

year_range Range of available years for data topic. Character string.

time_units Available time units. Character string.

space_units Available spatial units. Character string.

geosets Names of available geographic boundary data sources. Character string.

Source

Sub-National Geospatial Data Archive System: Geoprocessing Toolkit (updated March 17, 2023).

`cc_dict` *Country code dictionary*

Description

Reference table of country names and ISO-3166 codes, adapted from `countrycode` package.

Usage

```
cc_dict
```

Format

data.table object, with 8626 obs. of 3 variables:

country_name Country names. Character string.

country_name_alt Alternative spellings of country names, ASCII characters only. Character string.

country_iso3 Country codes (ISO 3166-1 alpha-3). Character string.

Source

Vincent Arel-Bundock. Package countrycode: Convert Country Names and Country Code, version 1.40. CRAN (October 12, 2022).

`clea_deu2009` *Constituency level results for lower chamber legislative elections, Germany 2009.*

Description

A simple feature collection containing the spatial geometries of electoral constituency borders, and data on turnout levels, votes shares and other attributes of lower chamber legislative elections.

Usage

```
clea_deu2009
```

Format

Simple feature collection with 16 features and 10 fields. geometry type: MULTIPOLYGON. dimension: XY. bbox: xmin: 5.867281 ymin: 47.27096 xmax: 15.04388 ymax: 55.05902. epsg (SRID): 4326. proj4string: +proj=longlat +datum=WGS84 +no_defs.

cst Constituency number. Numeric.

cst_n Constituency name. Character.

ctr Country number. Numeric.

ctr_n Country name. Character.

yrmo Year and month of election (YYYYMM). Character.

to1 Turnout in first round. Numeric.

vv1 Number of valid votes in first round. Numeric.

pvs1_margin Popular vote share margin in first round. Numeric.

incumb_pty_n Incumbent party name.

win1_pty_n Party name of popular vote share winner in first round. Character.

Source

Constituency-Level Elections Archive (CLEA) https://electiondataarchive.org/

---

clea_deu2009_df *Constituency level results for lower chamber legislative elections, Germany 2009.*

Description

A data.frame object containing the geographic centroids of electoral contituencies, and data on turnout levels, votes shares and other attributes of lower chamber legislative elections.

Usage

```
clea_deu2009_df
```

Format

data.frame with 16 observations and 12 variables.

cst Constituency number. Numeric.

cst_n Constituency name. Character.

ctr Country number. Numeric.

ctr_n Country name. Character.

yrmo Year and month of election (YYYYMM). Character.

to1 Turnout in first round. Numeric.

vv1 Number of valid votes in first round. Numeric.

pvs1_margin Popular vote share margin in first round. Numeric.

incumb_pty_n Incumbent party name.

win1_pty_n Party name of popular vote share winner in first round. Character.

longitude Longitude of constituency centroid. Numeric.

latitude Latitude of constituency centroid. Numeric.

Source

Constituency-Level Elections Archive (CLEA) https://electiondataarchive.org/

`clea_deu2009_pt` *Constituency level results for lower chamber legislative elections, Germany 2009.*

Description

A simple feature collection containing the geographic centroids of electoral contituencies, and data on turnout levels, votes shares and other attributes of lower chamber legislative elections.

Usage

```
clea_deu2009_pt
```

Format

Simple feature collection with 16 features and 10 fields. geometry type: POINT. dimension: XY. bbox: xmin: 6.953882 ymin: 48.54535 xmax: 13.40315 ymax: 54.18635. epsg (SRID): 4326. proj4string: +proj=longlat +datum=WGS84 +no_defs.

cst Constituency number. Numeric.

cst_n Constituency name. Character.

ctr Country number. Numeric.

ctr_n Country name. Character.

yrmo Year and month of election (YYYYMM). Character.

to1 Turnout in first round. Numeric.

vv1 Number of valid votes in first round. Numeric.

pvs1_margin Popular vote share margin in first round. Numeric.

incumb_pty_n Incumbent party name.

win1_pty_n Party name of popular vote share winner in first round. Character.

Source

Constituency-Level Elections Archive (CLEA) https://electiondataarchive.org/

df2sf *Convert data.frame object into simple features object*

Description

Function takes in x-, y-coordinates, and a data.frame of variables (optional) and returns an SFC object

Usage

```
df2sf(
  x_coord,
  y_coord,
  input_data = NULL,
  file = NULL,
  n_max = Inf,
  start = 0,
  projection_input = "EPSG:4326",
  zero.policy = FALSE,
  show_removed = FALSE
)
```

Arguments

x_coord	Numeric vector with longitude or easting projected coordinates. When input_data or file is supplied, can be either column name or numeric vector of the same length as nrow(input_data).
y_coord	Numeric vector with latitude or northing projected coordinates. Must be equal to the vector length of x_coord. When input_data or file is supplied, can be either column name or numeric vector of the same length as nrow(input_data).
input_data	Optional data frame object, containing x_coord and y_coord. nrow(input_data) must be equal to the vector length of x_coord. NOTE: Rows corresponding to non-usable coordinates are removed from the final output.
file	Optional path to csv file. Overrides input_data.
n_max	Maximum number of rows to read in file. Default is Inf.
start	Number of rows to skip in file. Default is 0 (start on first row).
projection_input	Projection string associated with x_coord and y_coord. Default is '+proj=longlat'.
zero.policy	If TRUE, removes rows where corresponding coordinates equals (0,0). Default is FALSE.
show_removed	If TRUE, returns a vector of indices corresponding to non-usable coordinates. Default is FALSE.

Value

If show_removed==FALSE, returns an sf object, with rows corresponding to non-usable coordinates removed. If show_removed==TRUE, returns a list, with an sf object (Spatial_Coordinates), and a vector of indices corresponding to non-usable coordinates removed (Removed_Rows).

Examples

```
# Coordinates supplied as vectors
## Not run:
data(clea_deu2009_df)
out_1 <- df2sf(x_coord=clea_deu2009_df$longitude,y_coord = clea_deu2009_df$latitude)
class(out_1)
plot(out_1$geometry)

## End(Not run)
# Coordinates supplied as column mames
## Not run:
out_2 <- df2sf(x_coord="longitude",y_coord ="latitude", input_data = clea_deu2009_df)
plot(out_2["geometry"])

## End(Not run)
# Load from external file
## Not run:
tmp <- tempfile()
write.csv(clea_deu2009_df,file=tmp)
out_3 <- df2sf(x_coord="longitude",y_coord ="latitude", file=tmp)
plot(out_3["geometry"])

## End(Not run)
```

---

fix_geom *Fix polygon geometries*

Description

Function to check validity and fix broken geometries in simple features polygon objects

Usage

```
fix_geom(x, n_it = 10)
```

Arguments

x	Polygon layer to be checked and fixed. sf object.
n_it	Number of iterations. Default is 10. Numeric..

Value

Returns a sf polygon object, with self-intersections and other geometry problems fixed.

Examples

```
# Assignment of a single variable (sums)
## Not run:
data(clea_deu2009)
out_1 <- fix_geom(clea_deu2009)

## End(Not run)
```

---

geocode_osm *Geocode addresses with OpenStreetMap*

Description

Function to find geographic coordinates of addresses and place names, using OpenStreetMap's Nominatum API.

Usage

```
geocode_osm(
  query,
  match_num = 1,
  return_all = FALSE,
  details = FALSE,
  user_agent = NULL
)
```

Arguments

query	Address or place name to be geocoded. Character string.
match_num	If query matches multiple locations, which match to return? Default is 1 (highest-ranking match, by relevance). Numeric.
return_all	Should all matches be returned? Overrides match_num if TRUE. Default is FALSE. Logical.
details	Should detailed results be returned? Default is FALSE. Logical.
user_agent	Valid User-Agent identifying the application for OSM-Nominatum. If none supplied, function will attempt to auto-detect. Character string.

Details

Note that Nominatim Usage Policy stipulates an absolute maximum of 1 request per second (https://operations.osmfoundation.org/policies/nominatim/). For batch geocoding of multiple addresses, please use geocode_osm_batch.

Value

A `data.frame` object. If `details=FALSE`, contains fields

- "query". User-supplied address query(ies). Character string.
- "osm_id". OpenStreetMap ID. Character string.
- "address". OpenStreetMap address. Character string.
- "longitude". Horizontal coordinate. Numeric.
- "latitude". Vertical coordinate. Numeric.

If `details=TRUE`, contains additional fields

- "osm_type". OpenStreetMap ID. Character string.
- "importance". Relevance of Nominatum match to query, from 0 (worst) to 1 (best). Numeric.
- "bbox_ymin". Minimum vertical coordinate of bounding box. Numeric.
- "bbox_ymax". Maximum vertical coordinate of bounding box. Numeric.
- "bbox_xmin". Minimum horizontal coordinate of bounding box. Numeric.
- "bbox_xmax". Maximum horizontal coordinate of bounding box. Numeric.

Examples

```
# Geocode an address (top match only)
## Not run:
geocode_osm("Michigan Stadium")

## End(Not run)
# Return detailed results for top match
## Not run:
geocode_osm("Michigan Stadium", details=TRUE)

## End(Not run)
# Return detailed results for all matches
## Not run:
geocode_osm("Michigan Stadium", details=TRUE, return_all = TRUE)

## End(Not run)
```

`geocode_osm_batch` *Batch geocode addresses with OpenStreetMap*

Description

Function to find geographic coordinates of multiple addresses and place names, using OpenStreetMap's Nominatum API.

Usage

```
geocode_osm_batch(
  query,
  delay = 1,
  return_all = FALSE,
  match_num = 1,
  details = FALSE,
  user_agent = NULL,
  verbose = FALSE
)
```

Arguments

query	Addresses or place names to be geocoded. Character string.
delay	Delay between requests. Default is 1 second. Numeric.
return_all	Should all matches be returned? Overrides match_num if TRUE. Default is FALSE. Logical.
match_num	If query matches multiple locations, which match to return? Default is 1 (highest-ranking match, by relevance). Numeric.
details	Should detailed results be returned? Default is FALSE. Logical.
user_agent	Valid User-Agent identifying the application for OSM-Nominatum. If none supplied, function will attempt to auto-detect. Character string.
verbose	Print status messages and progress? Default is FALSE. Logical.

Details

Wrapper function for geocode_osm. Because Nominatim Usage Policy stipulates an absolute maximum of 1 request per second, this function facilitates batch geocoding by adding a small delay between queries (https://operations.osmfoundation.org/policies/nominatim/).

Value

A data.frame object. If details=FALSE, contains fields

- "query". User-supplied address query(ies). Character string.
- "osm_id". OpenStreetMap ID. Character string.
- "address". OpenStreetMap address. Character string.
- "longitude". Horizontal coordinate. Numeric.
- "latitude". Vertical coordinate. Numeric.

If details=TRUE, contains additional fields

- "osm_type". OpenStreetMap ID. Character string.
- "importance". Relevance of Nominatum match to query, from 0 (worst) to 1 (best). Numeric.
- "bbox_ymin". Minimum vertical coordinate of bounding box. Numeric.
- "bbox_ymax". Maximum vertical coordinate of bounding box. Numeric.
- "bbox_xmin". Minimum horizontal coordinate of bounding box. Numeric.
- "bbox_xmax". Maximum horizontal coordinate of bounding box. Numeric.

Examples

```
# Geocode multiple addresses (top matches only)
## Not run:
geocode_osm_batch(c("Ann Arbor","East Lansing","Columbus"))

## End(Not run)
# With progress reports
## Not run:
geocode_osm_batch(c("Ann Arbor","East Lansing","Columbus"), verbose = TRUE)

## End(Not run)
# Return detailed results for all matches
## Not run:
geocode_osm_batch(c("Ann Arbor","East Lansing","Columbus"),
                  details = TRUE, return_all = TRUE)

## End(Not run)
```

---

get_data *Download data from SUNGEO server*

Description

Function to download data files through the SUNGEO API. Function produces a data.table object, corresponding to the user's choice of countries, topics, sources, and spatial and temporal units.

Usage

```
get_data(
  country_names = NULL,
  country_iso3 = NULL,
  geoset = "GADM",
  geoset_yr = 2018,
  space_unit = "adm1",
  time_unit = "year",
  topics = NULL,
  year_min = 1990,
  year_max = 2017,
  print_url = TRUE,
  print_time = TRUE,
  error_stop = FALSE,
  by_topic = TRUE,
  skip_missing = TRUE,
  cache_param = FALSE,
  short_message = TRUE
)
```

Arguments

country_names	Country name(s). Character string (single country) or vector of character strings (multiple countries).
country_iso3	Country code (ISO 3166-1 alpha-3). Character string (single country) or vector of character strings (multiple countries).
geoset	Name of geographic boundary set. Can be one of "GADM" (Database of Global Administrative Areas), "GAUL" (Global Administrative Unit Layers), "geoBoundaries", "GRED" (GeoReferenced Electoral Districts Datasets), "HEXGRID" (SUNGEO Hexagonal Grid), "MPIDR" (Max Planck Institute for Demographic Research Population History GIS Collection), "NHGIS" (National Historical Geographic Information System), "PRIOGRID" (PRIO-GRID 2.0), "SHGIS" (SUNGEO Historical GIS). Default is "GADM". Character string.
geoset_yr	Year of geographic boundaries. See get_info()['geosets'] for availability. Default is 2018. Integer.
space_unit	Geographic level of analysis. Can be one of "adm0" (country), "adm1" (province), "adm2" (district), "cst" (GRED electoral constituency), "hex05" (SUNGEO Hexagonal Grid cell), "prio" (PRIO-GRID cell). See get_info()['geosets'] for availability by geoset, country and topic. Default is "adm1". Character string.
time_unit	Temporal level of analysis. Can be one of "year", "month", "week". See get_info()['topics'] for availability by topic. Default is "year". Character string.
topics	Data topics. See get_info()['summary'] for full list. Character string (single topic) or vector of character strings (multiple topics).
year_min	Time range of requested data: start year. See get_info()['topics'] for availability by topic. Default is 1990. Integer.
year_max	Time range of requested data: end year. See get_info()['topics'] for availability by topic. Default is 2017. Integer.
print_url	Print url string of requested data to console? Default is TRUE. Logical.
print_time	Print processing time for API query to console? Default is TRUE. Logical.
error_stop	Error handling. If TRUE, function terminates request if an error is encountered. If FALSE, error is skipped and error message is recorded in a new message column. Default is FALSE. Logical.
by_topic	Break query down by topic and country? If TRUE, a separate request is sent to the API for each country and topic, and the results are combined on the client side. This ensures that data that are available for some, but not all countries are returned, rather than resulting in a failed request. If FALSE, a single request is sent to the API for all countries and topics, and the results are combined on the server side. Only data that are available for all countries are returned. Default is TRUE. Logical.
skip_missing	Skip missing data topics? If TRUE, missing data topics are skipped, columns are populated with NAs, and corresponding error message is recorded in a new message column. If FALSE, returns NULL results for missing topics. Default is TRUE. Logical.

cache_param	Store cached query on server? This can speed up processing for repeated queries. Default is FALSE. Logical.
short_message	Shorten error messages? If TRUE, a short, informative error message is recorded in the message column. If FALSE, full error message is recorded. Default is TRUE. Logical.

Value

data.table object, with requested data from SUNGEO API.

See Also

get_info

Examples

```
# Single country, single topic
## Not run:
out_1 <- get_data(country_name="Afghanistan",topics="Demographics:Population:GHS")
out_1

## End(Not run)

## Not run:
out_2 <- get_data(
 country_name=c("Afghanistan","Moldova"),
 topics=c("Demographics:Ethnicity:EPR","Demographics:Population:GHS"))
out_2

## End(Not run)

# Other boundary sets, spatial and time units
## Not run:
out_3 <- get_data(
 country_name="Albania",
 topics="Weather:AirTemperatureAndPrecipitation:NOAA",
 geoset="GAUL",geoset_yr=1990,space_unit="adm2",time_unit="month",
 year_min=1990,year_max=1991)
out_3

## End(Not run)
```

---

get_info *Information on available SUNGEO data files*

Description

This function reports the availability of data files on the SUNGEO server, searchable by country and topic.

Usage

```
get_info(country_names = NULL, country_iso3s = NULL, topics = NULL)
```

Arguments

country_names	Country name(s). Character string (single country) or vector of character strings (multiple countries).
country_iso3s	Country code (ISO 3166-1 alpha-3). Character string (single country) or vector of character strings (multiple countries).
topics	Data topics. See `get_info()` for full list. Character string (single topic) or vector of character strings (multiple topics).

Value

list object, with three slots: 'summary', 'topics', and 'geoset'.

See Also

get_data

Examples

```
# Get list of all available data
## Not run:
out_1 <- get_info()
out_1["summary"]
out_1["topics"]
out_1["geosets"]

## End(Not run)

# Get list of available data for a single country
## Not run:
out_2 <- get_info(country_names="Afghanistan")
out_2

## End(Not run)

# Get list of available data for a single topic
## Not run:
out_3 <- get_info(topics="Elections:LowerHouse:CLEA")
out_3

## End(Not run)

# Get list of available data for a multiple countries and topics
## Not run:
out_4 <- get_info(
                country_names=c("Afghanistan","Zambia"),
                topics=c("Elections:LowerHouse:CLEA","Events:PoliticalViolence:GED"))
out_4
```

```
## End(Not run)
```

`gpw4_deu2010` *Population count raster for Germany, 2010.*

Description

2.5 arc-minute resolution raster of estimates of human population (number of persons per pixel), consistent with national censuses and population registers, for the year 2010.

Usage

```
gpw4_deu2010
```

Format

class : SpatRaster dimensions : 186, 220, 1 (nrow, ncol, nlyr) resolution : 0.04166667, 0.04166667 (x, y) extent : 5.875, 15.04167, 47.29167, 55.04167 (xmin, xmax, ymin, ymax) coord. ref. : lon/lat WGS 84 (EPSG:4326) source(s) : memory name : gpw_v4_population_count_rev11_2010_2pt5_min min value : 0.00 max value : 92915.66

Source

Gridded Population of the World (GPW) v4: Population Count, v4.11 <doi:10.7927/H4JW8BX5>.

`hex_05_deu` *Hexagonal grid for Germany.*

Description

Regular hexagonal grid of 0.5 degree diameter cells, covering territory of Germany (2020 borders).

Usage

```
hex_05_deu
```

Format

Simple feature collection with 257 features and 3 fields. geometry type: POLYGON. dimension: XY. bbox: xmin: 5.375001 ymin: 46.76568 xmax: 15.375 ymax: 55.13726. epsg (SRID): 4326. proj4string: +proj=longlat +datum=WGS84 +no_defs.

HEX_ID Unique cell identifier. Character.

HEX_X Longitude of cell centroid. Numeric.

HEX_Y Latitude of cell centroid. Numeric.

Source

SUNGEO

`highways_deu1992` *Roads polylines for Germany, 1992*

Description

Roads thematic layer from Digital Chart of the World. Subset: divided multi-lane highways.

Usage

```
highways_deu1992
```

Format

Simple feature collection with 1741 features and 5 fields. geometry type: MULTILINESTRING. dimension: XY. bbox: xmin: 5.750933 ymin: 47.58799 xmax: 14.75109 ymax: 54.80712 epsg (SRID): 4326. proj4string: +proj=longlat +datum=WGS84 +no_defs.

MED_DESCRI Is the road a divided multi-lane highway with a median? Character string.

RTT_DESCRI Primary or secondary route? Character string.

F_CODE_DES Feature code description (road or trail). Character string.

ISO ISO 3166-1 alpha-3 country code. Character string.

ISOCOUNTRY Country name. Character string.

Source

Defense Mapping Agency (DMA), 1992. Digital Chart of the World. Defense Mapping Agency, Fairfax, Virginia. (Four CD-ROMs). Available through DIVA-GIS: http://www.diva-gis.org/gData (accessed August 12, 2021).

`hot_spot` *Automatically calculate Local G hot spot intensity*

Description

Function automatically calculates the Local G hot spot intensity measure for spatial points, spatial polygons, and single raster layers. Uses RANN for efficient nearest neighbor calculation (spatial points and single raster layers only); users can specify the number of neighbors (k). Users can specify the neighborhood style (see spdep::nb2listw) with default being standardized weight matrix (W).

Usage

```
hot_spot(
  insert,
  variable = NULL,
  style = "W",
  k = 9,
  remove_missing = TRUE,
  NA_Value = 0,
  include_Moran = FALSE
)
```

Arguments

insert	Spatial point, spatial polygon, or single raster layer object. Acceptable formats include sf, SpatialPolygonsDataFrame, SpatialPointsDataFrame, and RasterLayer.
variable	Column name or numeric vector containing the variable from which the local G statistic will be calculated. Must possess a natural scale that orders small and large observations (i.e. number, percentage, ratio and not model residuals).
style	Style can take values 'W', 'B', 'C', 'U', 'mimax', 'S' (see nb2listw). Character string.
k	Number of neighbors. Default is 9. Numeric.
remove_missing	Whether to calculate statistic without missing values. If FALSE, substitute value must be supplied to NA_Value.
NA_Value	Substitute for missing values. Default value is 0. Numeric.
include_Moran	Calculate local Moran's I statistics. Default is FALSE. Logical.

Value

If input is sf, SpatialPolygonsDataFrame or SpatialPointsDataFrame object, returns sf object with same geometries and columns as input, appended with additional column containing Local G estimates (LocalG). If input is RasterLayer object, returns RasterBrick object containing original values (Original) and Local G estimates (LocalG).

Examples

```
# Calculate Local G for sf point layer

## Not run:
data(clea_deu2009_pt)
out_1 <- hot_spot(insert=clea_deu2009_pt, variable = clea_deu2009_pt$to1)
class(out_1)
plot(out_1["LocalG"])

## End(Not run)

# Calculate Local G for sf polygon layer (variable as numeric vector)

## Not run:
```

```
data(clea_deu2009)
out_2 <- hot_spot(insert=clea_deu2009, variable = clea_deu2009$to1)
summary(out_2$LocalG)
plot(out_2["LocalG"])

## End(Not run)

# Calculate Local G for sf polygon layer (variable as column name)

## Not run:
out_3 <- hot_spot(insert=clea_deu2009, variable = "to1")
summary(out_3$LocalG)
plot(out_3["LocalG"])

## End(Not run)

# Calculate Local G for sf polygon SpatialPolygonsDataFrame (variable as column name)

## Not run:
out_4 <- hot_spot(insert=as(clea_deu2009,"Spatial"), variable = "to1")
summary(out_4$LocalG)
plot(out_4["LocalG"])

## End(Not run)

# Calculate Local G for RasterLayer
## Not run:
data(gpw4_deu2010)
out_5 <- hot_spot(insert=gpw4_deu2010)
class(out_5)
terra::plot(out_5$LocalG)

## End(Not run)
```

`line2poly` *Line-in-polygon analysis*

Description

Function for basic geometry calculations on polyline features, within an overlapping destination polygon layer.

Usage

```
line2poly(
  linez,
  polyz,
  poly_id,
  measurez = c("length", "density", "distance"),
  outvar_name = "line",
```

```
  unitz = "km",
  reproject = TRUE,
  na_val = NA,
  verbose = TRUE
)
```

Arguments

linez	Source polyline layer. sf object.
polyz	Destination polygon layer. Must have identical CRS to linez. sf object.
poly_id	Name of unique ID column for destination polygon layer. Character string.
measurez	Desired measurements. Could be any of "length" (sum of line lengths by polygon), "density" (sum of line lengths divided by area of polygon) and/or "distance" (distance from each polygon to nearest line feature). Default is to report all three. Character string or vector of character strings.
outvar_name	Name (root) to be given to output variable. Default is "line". Character string.
unitz	Units of measurement (linear). Defaul is "km". Character string.
reproject	Temporarily reproject layers to planar projection for geometric operations? Defaul is TRUE. Logical.
na_val	Value to be assigned to missing values (line lengths and densities only). Defaul is NA. Logical or list.
verbose	Print status messages and progress? Default is TRUE. Logical.

Value

An sf polygon object, with summary statisics of linez features aggregated to the geometries of polyz.

If measurez = "lengths", contains fields with suffixes

- "_length". Sum of line lengths within each polygon, in km or other units supplied in unitz.

If measurez = "density", contains fields with suffixes

- "_length". Sum of line lengths within each polygon, in km or other units supplied in unitz.
- "_area". Area of each polygon, in km^2 or the square of linear units supplied in unitz.
- "_density". Sum of line lengths divided by area of each polygon, in km/km^2 or other units supplied in unitz.

If measurez = "distance", contains fields with suffixes

- "_distance". Distance from each polygon to nearest line feature, in km or other units supplied in unitz.

If measurez = c("length","density","distance") (default), contains all of the above.

Examples

```
# Road lengths, densities and distance from polygon to nearest highway
## Not run:
data(hex_05_deu)
data(highways_deu1992)
out_1 <- line2poly(linez = highways_deu1992,
                   polyz = hex_05_deu,
                   poly_id = "HEX_ID")
plot(out_1["line_length"])
plot(out_1["line_density"])
plot(out_1["line_distance"])

## End(Not run)

# Replace missing road lengths and densities with 0's, rename variables
## Not run:
out_2 <- line2poly(linez = highways_deu1992,
                   polyz = hex_05_deu,
                   poly_id = "HEX_ID",
                   outvar_name = "road",
                   na_val = 0)
plot(out_2["road_length"])
plot(out_2["road_density"])
plot(out_2["road_distance"])

## End(Not run)
```

---

make_ticker *Make date ticker*

Description

Function to create a table of consecutive dates, in SUNGEO-compliant format.

Usage

```
make_ticker(
  date_min = 19000101,
  date_max = as.integer(gsub("-", "", as.Date(Sys.Date())))
)
```

Arguments

date_min	Start date, in YYYYMMDD format. Default is 19000101. Integer.
date_max	End date, in YYYYMMDD format. Default is today. Integer.

Value

`data.table` object, with seven columns:

- `DATE`. Date in YYYYMMDD format. Integer.
- `DATE_ALT`. Date in `Date` (YYYY-MM-DD) format. Date.
- `TID`. Date ID, in consecutive integer format. Integer.
- `YRWK`. Week in YYYYWW format. Integer.
- `WID`. Weed ID, in consecutive integer format. Integer.
- `YRMO`. Month in YYYYMM format. Integer.
- `MID`. Month ID, in consecutive integer format. Integer.
- `YEAR`. Year in YYYY format. Integer.

Examples

```
# All dates from January 1, 1900 to today
## Not run:
out_1 <- make_ticker()
out_1

## End(Not run)

# All dates from January 1, 1200 to today
## Not run:
out_2 <- make_ticker(date_min=12000101)
out_2

## End(Not run)

# All dates from January 1, 1500 to December 31, 1899
## Not run:
out_3 <- make_ticker(date_min=15000101, date_max=18991231)
out_3

## End(Not run)
```

---

merge_list *Merge list of tables on common variable(s)*

Description

Function that finds a set of common columns in a list of tables, and merges the tables on these columns.

Usage

```
merge_list(lst)
```

Arguments

lst	List of tables to be merged. List object.

Value

data.table object

Examples

```
# Merge list of three tables with different common variables
## Not run:
A <- data.table::data.table(month=month.name,year=rep(1991:1992,each=12),A=rnorm(24))
B <- data.table::data.table(year=c(1991,1992),B=rbeta(2,1,1))
C <- data.table::data.table(month=month.name,C=runif(12))

out_1 <- merge_list(list(A,B,C))
out_1

## End(Not run)
```

---

nesting *Relative scale and nesting coefficients*

Description

Function to calculate relative scale and nesting metrics for changes of support from a source polygon layer to an overlapping (but spatially misaligned) destination polygon layer.

Usage

```
nesting(
  poly_from = NULL,
  poly_to = NULL,
  metrix = "all",
  tol_ = 0.001,
  by_unit = FALSE
)
```

Arguments

poly_from	Source polygon layer. sf object (polygon or multipolygon).
poly_to	Destination polygon layer. Must have identical CRS to poly_from. sf object (polygon or multipolygon).
metrix	Requested scaling and nesting metrics. See "details". Default is "all". Character string or vector of character strings.

`tol_`	Minimum area of polygon intersection, in square meters. Default is 0.001. Numeric.
`by_unit`	Include a by-unit decomposition of requested nesting metrics (if available)? Default is FALSE. Logical.

Details

Currently supported metrics (`metrix`) include:

- Relative scale ("rs"). Measures whether a change-of-support (CoS) task is one of aggregation or disaggregation, by calculating the share of source units that are smaller than destination units. Its range is from 0 to 1, where values of 1 indicate pure aggregation (all source units are smaller than destination units) and values of 0 indicate no aggregation (all source units are at least as large as destination units). Values between 0 and 1 indicate a hybrid (i.e. some source units are smaller, others are larger than target units).
- Relative nesting ("rn"). Measures how closely source and destination boundaries align, by calculating the share of source units that cannot be split across multiple destination units. Its range is from 0 to 1, where values of 0 indicate no nesting (every source unit can be split across multiple destination units) and values of 1 indicate full nesting (no source unit can be split across multiple destination units).
- Relative scale, symmetric ("rs_sym"). Alternative measure of "rs", which ranges from -1 to 1. It calculates a difference between two proportions: the share of source units that is smaller than destination units (i.e. "rs" from standpoint of source units), and the share that is larger (i.e. "rs" from standpoint of destination units). Values of -1 indicate pure disaggregation (all source units are larger than destination units), 1 indicates pure aggregation (all source units are smaller than destination units). Values of 0 indicate that all source units are the same size as target units.
- Relative nesting, symmetric ("rn_sym"). Alternative measure of "rn", which ranges from -1 to 1. It calculates a difference between two components: the nesting of source units within destination units (i.e. "rn" from standpoint of source units), and the nesting of destination units within source units (i.e. "rn" from standpoint of destination units. Values of 1 indicate that source units are perfectly nested within destination units; -1 indicates that destination units are perfectly nested within source units.
- Relative scale, alternative ("rs_alt"). Alternative measure of "rs", rescaled as a proportion of destination unit area. This measure can take any value on the real line, with positive values indicating aggregation and negative values indicating disaggregation.
- Relative nesting, alternative ("rn_alt"). Alternative measure of "rn", which places more weight on areas of maximum overlap. The main difference between this measure and "rn" is its use of the maximum intersection area for each source polygon instead of averaging over the quadratic term. Two sets of polygons are considered nested if one set is completely contained within another, with as few splits as possible. If none or only a sliver of a source polygon area falls outside a single destination polygon, those polygons are "more nested" than a case where half of a source polygon falls in destination polygon A and half falls into another polygon B.
- Relative scale, conditional ("rs_nn"). Alternative measure of "rs", calculated for the subset of source units that are not fully nested within destination units.
- Relative nesting, conditional ("rn_nn"). Alternative measure of "rn", calculated for the subset of source units that are not fully nested within destination units.

- Proportion intact ("p_intact"). A nesting metric that requires no area calculations at all. This measure ranges from 0 to 1, where 1 indicates full nesting (i.e. every source unit is intact/no splits), and 0 indicates no nesting (i.e. no source unit is intact/all are split).
- Proportion fully nested ("full_nest"). A stricter version of "p_intact". This measure ranges from 0 to 1, where 1 indicates full nesting (i.e. every source unit is intact/no splits AND falls completely inside the destination layer), and 0 indicates no nesting (i.e. no source unit is both intact and falls inside destination layer).
- Relative overlap ("ro"). Assesses extent of spatial overlap between source and destination polygons. This measure is scaled between -1 and 1. Values of 0 indicate perfect overlap (there is no part of source units that fall outside of destination units, and vice versa). Values between 0 and 1 indicate a "source underlap" (some parts of source polygons fall outside of destination polygons; more precisely, a larger part of source polygon area falls outside destination polygons than the other way around). Values between -1 and 0 indicate a "destination underlap" (some parts of destination polygons fall outside of source polygons; a larger part of destination polygon area falls outside source polygons than the other way around). Values of -1 and 1 indicate no overlap (all source units fall outside destination units, and vice versa). This is a theoretical limit only; the function returns an error if there is no overlap.
- Gibbs-Martin index of diversification ("gmi"). Inverse of "rn", where values of 1 indicate that every source unit is evenly split across multiple destination units, and 0 indicates that no source unit is split across any destination units.

It is possible to pass multiple arguments to `metrix` (e.g. `metrix=c("rn","rs")`). The default (`metrix="all"`) returns all of the above metrics.

The function automatically reprojects source and destination geometries to Lambert Equal Area prior to calculation, with map units in meters.

Values of `tol_` can be adjusted to increase or decrease the sensitivity of these metrics to small border misalignments. The default value discards polygon intersections smaller than 0.001 square meters in area.

Value

Named list, with numeric values for each requested metric in `metrix`. If `by_unit==TRUE`, last element of list is a data.table, with nesting metrics disaggregated by source unit, where the first column is a row index for the source polygon layer.

Examples

```
# Calculate all scale and nesting metrics for two sets of polygons
## Not run:
data(clea_deu2009)
data(hex_05_deu)
nest_1 <- nesting(
              poly_from = clea_deu2009,
              poly_to = hex_05_deu
              )
nest_1

## End(Not run)
```

```
# Calculate just Relative Nesting, in the opposite direction
## Not run:
nest_2 <- nesting(
              poly_from = hex_05_deu,
              poly_to = clea_deu2009,
              metrix = "rn"
              )
nest_2

## End(Not run)
```

`point2poly_krige` *Point-to-polygon interpolation, ordinary and universal Kriging method*

Description

Function for interpolating values from a source points layer to an overlapping destination polygon layer, using ordinary and universal kriging with automatic variogram fitting

Usage

```
point2poly_krige(
  pointz,
  polyz,
  rasterz = NULL,
  yvarz = NULL,
  xvarz = NULL,
  pycno_yvarz = NULL,
  funz = base::mean,
  use_grid = FALSE,
  nz_grid = 25,
  blockz = 0,
  pointz_x_coord = NULL,
  pointz_y_coord = NULL,
  polyz_x_coord = NULL,
  polyz_y_coord = NULL,
  messagez = ""
)
```

Arguments

`pointz`	Source points layer. `sf`, `sp`, or data frame object.
`polyz`	Destination polygon layer. Must have identical CRS to `pointz`. `sf`, `sp`, or data frame object.
`rasterz`	Source raster layer (or list of raster), with covariate(s) used for universal kriging. Must have identical CRS to `polyz`. `RasterLayer` object or list of `RasterLayer` objects.

yvarz	Names of numeric variable(s) to be interpolated from source points layer to destination polygons. Character string or vector of character strings.
xvarz	Names of numeric variable(s) for universal Kriging, in which yvarz is linearly dependent. Character string or vector of character strings.
pycno_yvarz	Names of spatially extensive numeric variables for which the pycnophylactic (mass-preserving) property should be preserved. Must be a subset of yvarz. Character string or vector of character strings.
funz	Aggregation function to be applied to values in rasterz and to interpolated values. Must take as an input a vector x. Default is mean. Function.
use_grid	Use regular grid as destination layer for interpolation, before aggregating to polygons? Default is FALSE.
nz_grid	Number of grid cells in x and y direction (columns, rows). Integer of length 1 or 2. Default is 25. Ignored if use_grid=FALSE.
blockz	Size of blocks used for Block Kriging, in meters. Integer of length 1 or 2. Default is 0.
pointz_x_coord	Name of numeric variable corresponding to a measure of longitude (Easting) in a data frame object for pointz. Character string.
pointz_y_coord	Name of numeric variable corresponding to a measure of Latitude (Northing) in a data frame object for pointz. Character string.
polyz_x_coord	Name of numeric variable corresponding to a measure of longitude (Easting) in a data frame object for polyz. Character string.
polyz_y_coord	Name of numeric variable corresponding to a measure of Latitude (Northing) in a data frame object for polyz. Character string.
messagez	Optional message to be printed during Kriging estimation. Character string.

Details

This function performs Ordinary and Universal Kriging, automatically selecting a variogram model with the smallest residual sum of squares from the sample variogram. See autofitVariogram.

Unlike other available point-to-polygon interpolation techniques, this function currently only accepts numeric variables in varz and does not support interpolation of character strings.

Value

sf polygon object, with variables from pointz interpolated to the geometries of polyz.

Examples

```
# Ordinary Kriging with one variable
## Not run:
data(clea_deu2009)
data(clea_deu2009_pt)
out_1 <- point2poly_krige(pointz = clea_deu2009_pt,
                          polyz = clea_deu2009,
                          yvarz = "to1")
par(mfrow=c(1,2))
```

```
plot(clea_deu2009["to1"], key.pos = NULL, reset = FALSE)
plot(out_1["to1.pred"], key.pos = NULL, reset = FALSE)

## End(Not run)

# Ordinary Kriging with multiple variables
## Not run:
out_2 <- point2poly_krige(pointz = clea_deu2009_pt,
                          polyz = clea_deu2009,
                          yvarz = c("to1","pvs1_margin"))
par(mfrow=c(1,2))
plot(clea_deu2009["pvs1_margin"], key.pos = NULL, reset = FALSE)
plot(out_2["pvs1_margin.pred"], key.pos = NULL, reset = FALSE)

## End(Not run)

# Universal Kriging with one variable from a raster
## Not run:
data(gpw4_deu2010)
data(clea_deu2009)
data(clea_deu2009_pt)
out_3 <- point2poly_krige(pointz = clea_deu2009_pt,
                          polyz = clea_deu2009,
                          yvarz = "to1",
                          rasterz = gpw4_deu2010)
par(mfrow=c(1,2))
plot(clea_deu2009["to1"], key.pos = NULL, reset = FALSE)
plot(out_3["to1.pred"], key.pos = NULL, reset = FALSE)

## End(Not run)

# Block Kriging with block size of 100 km
## Not run:
data(clea_deu2009)
data(clea_deu2009_pt)
out_4 <- point2poly_krige(pointz = clea_deu2009_pt,
                          polyz = clea_deu2009,
                          yvarz = "to1",
                          blockz = 100000)
par(mfrow=c(1,2))
plot(clea_deu2009["to1"], key.pos = NULL, reset = FALSE)
plot(out_4["to1.pred"], key.pos = NULL, reset = FALSE)

## End(Not run)
```

`point2poly_simp` *Point-to-polygon interpolation, simple overlay method*

Description

Function for assigning values from a source point layer to a destination polygon layer, using simple point-in-polygon overlays

Usage

```
point2poly_simp(
  pointz,
  polyz,
  varz,
  char_varz = NULL,
  funz = list(function(x) {
     sum(x, na.rm = TRUE)
 }),
  na_val = NA,
  drop_na_cols = FALSE
)
```

Arguments

pointz	Source points layer. sf object.
polyz	Destination polygon layer. Must have identical CRS to pointz. sf object.
varz	Names of variable(s) to be assigned from source polygon layer to destination polygons. Character string or vector of character strings.
char_varz	Names of character string variable(s) in varz. Character string or vector of character strings.
funz	Aggregation function to be applied to variables specified in varz. Must take as an input a vector x. Function or list of functions.
na_val	Value to be assigned to missing values. Defaul is NA. Logical or list.
drop_na_cols	Drop columns with completely missing values. Defaul is FALSE. Logical.

Details

Assignment procedures are the same for numeric and character string variables. All variables supplied in varz are passed directly to the function specified in funz. If different sets of variables are to be aggregated with different functions, both varz and funz should be specified as lists (see examples below).

Value

Returns a sf polygon object, with variables from pointz assigned to the geometries of polyz.

Examples

```
# Assignment of a single variable (sums)
## Not run:
data(hex_05_deu)
```

```
data(clea_deu2009_pt)
out_1 <- point2poly_simp(pointz=clea_deu2009_pt,
                         polyz=hex_05_deu,
                         varz="vv1")
plot(out_1["vv1"])

## End(Not run)

# Replace NA's with 0's
## Not run:
out_2 <- point2poly_simp(pointz = clea_deu2009_pt,
                         polyz = hex_05_deu,
                         varz = "vv1",
                         na_val = 0)
plot(out_2["vv1"])

## End(Not run)

# Multiple variables, with different assignment functions
## Not run:
out_3 <- point2poly_simp(pointz = clea_deu2009_pt,
                         polyz = hex_05_deu,
                         varz = list(
                           c("to1","pvs1_margin"),
                           c("vv1"),
                           c("incumb_pty_n","win1_pty_n")),
                         funz = list(
                           function(x){mean(x,na.rm=TRUE)},
                           function(x){sum(x,na.rm=TRUE)},
                           function(x){paste0(unique(na.omit(x)),collapse=" | ") }),
                         na_val = list(NA_real_,0,NA_character_))

## End(Not run)
```

`point2poly_tess` *Point-to-polygon interpolation, tessellation method*

Description

Function for interpolating values from a source point layer to a destination polygon layer, using Voronoi tessellation and area/population weights.

Usage

```
point2poly_tess(
  pointz,
  polyz,
  poly_id,
  char_methodz = "aw",
  methodz = "aw",
```

```
  pop_raster = NULL,
  varz = NULL,
  pycno_varz = NULL,
  char_varz = NULL,
  char_assign = "biggest_overlap",
  funz = function(x, w) {
     stats::weighted.mean(x, w, na.rm = TRUE)
 },
  return_tess = FALSE,
  seed = 1
)
```

Arguments

pointz	Source points layer. sf object.
polyz	Destination polygon layer. Must have identical CRS to pointz. sf object.
poly_id	Name of unique ID column for destination polygon layer. Character string.
char_methodz	Interpolation method(s) for character strings. Could be either of "aw" (areal weighting, default) or "pw" (population weighting). See "details". Character string.
methodz	Interpolation method(s) for numeric covariates. Could be either of "aw" (areal weighting, default) and/or "pw" (population weighting). See "details". Character string or vector of character strings.
pop_raster	Population raster to be used for population weighting, Must be supplied if methodz="pw". Must have identical CRS to poly_from. raster or SpatRaster object.
varz	Names of numeric variable(s) to be interpolated from source polygon layer to destination polygons. Character string or list of character strings.
pycno_varz	Names of spatially extensive numeric variables for which the pycnophylactic (mass-preserving) property should be preserved. Character string or vector of character strings.
char_varz	Names of character string variables to be interpolated from source polygon layer to destination polygons. Character string or vector of character strings.
char_assign	Assignment rule to be used for variables specified in char_varz. Could be either "biggest_overlap" (default) or "all_overlap". See "details". Character string or vector of character strings.
funz	Aggregation function to be applied to variables specified in varz. Must take as an input a numeric vector x and vector of weights w. Function or list of functions.
return_tess	Return Voronoi polygons, in addition to destinaton polygon layer? Default is FALSE. Logical.
seed	Seed for generation of random numbers. Default is 1. Numeric.

Details

This function interpolates point data to polygons with a two-step process. In the first step (tessellation), each point is assigned a Voronoi cell, drawn such that (a) the distance from its borders to

the focal point is less than or equal to the distance to any other point, and (b) no gaps between cells remain. The second step (interpolation) performs a polygon-in-polygon interpolation, using the Voronoi cells as source polygons.

Currently supported integration methods in the second step (`methodz`) include:

- Areal weighting ("aw"). Values from `poly_from` weighted in proportion to relative area of spatial overlap between source features and geometries of `poly_to`.
- Population weighting ("pw"). Values from `poly_from` weighted in proportion to relative population sizes in areas of spatial overlap between source features and geometries of `poly_to`. This routine uses a third layer (supplied in `pop_raster`) to calculate the weights.

When a list of variables are supplied and one methods argument specified, then the chosen method will be applied to all variables.

When a list of of variables are supplied and multiple methods arguments specified, then weighting methods will be applied in a pairwise order. For example, specifying `varz = list(c("to1","pvs1_margin"), c("vv1"))` and `methodz = c('aw', 'pw')` will apply areal weighting to the the first set of variables (to1 and pvs1_margin) and population weighing to the second set (vv1).

Interpolation procedures are handled somewhat differently for numeric and character string variables. For numeric variables supplied in `varz`, "aw" and/or "pw" weights are passed to the function specified in `funz`. If different sets of numeric variables are to be aggregated with different functions, both `varz` and `funz` should be specified as lists (see examples below).

For character string (and any other) variables supplied in `char_varz`, "aw" and/or "pw" weights are passed to the assignment rule(s) specified in `char_assign`. Note that the `char_varz` argument may include numerical variables, but `varz` cannot include character string variables.

Currently supported assignment rules for character strings (`char_assign`) include:

- "biggest_overlap". For each variable in `char_varz`, the features in `poly_to` are assigned a single value from overlapping `poly_from` features, corresponding to the intersection with largest area and/or population weight.
- "all_overlap". For each variable in `char_varz`, the features in `poly_to` are assigned all values from overlapping `poly_from` features, ranked by area and/or population weights (largest-to-smallest) of intersections.

It is possible to pass multiple arguments to `char_assign` (e.g. `char_assign=c("biggest_overlap","all_overlap")`), in which case the function will calculate both, and append the resulting columns to the output.

Value

If `return_tess=FALSE`, returns a `sf` polygon object, with variables from `pointz` interpolated to the geometries of `polyz`.

If `return_tess=TRUE`, returns a list, containing

- "result". The destination polygon layer. `sf` object.
- "tess". The (intermediate) Voronoi tessellation polygon layer. `sf` object.

Examples

```
# Interpolation of a single variable, with area weights
## Not run:
data(hex_05_deu)
data(clea_deu2009_pt)
out_1 <- point2poly_tess(pointz = clea_deu2009_pt,
                            polyz = hex_05_deu,
                            poly_id = "HEX_ID",
                            varz = "to1")
plot(out_1["to1_aw"])

## End(Not run)

# Extract and inspect tessellation polygons
## Not run:
out_2 <- point2poly_tess(pointz = clea_deu2009_pt,
                            polyz = hex_05_deu,
                            poly_id = "HEX_ID",
                            varz = "to1",
                            return_tess = TRUE)
plot(out_2$tess["to1"])
plot(out_2$result["to1_aw"])

## End(Not run)

# Interpolation of multiple variables, with area and population weights
## Not run:
data(gpw4_deu2010)
out_3 <- point2poly_tess(pointz = clea_deu2009_pt,
                         polyz = hex_05_deu,
                         poly_id = "HEX_ID",
                         methodz = c("aw","pw"),
                         varz = list(
                           c("to1","pvs1_margin"),
                           c("vv1")
                         ),
                         pycno_varz = "vv1",
                         funz = list(
                           function(x,w){stats::weighted.mean(x,w)},
                           function(x,w){sum(x*w)}
                           ),
                         char_varz = c("incumb_pty_n","win1_pty_n"),
                         pop_raster = gpw4_deu2010)
plot(out_3["vv1_pw"])

## End(Not run)
```

---

poly2poly_ap *Area and population weighted polygon-to-polygon interpolation*

Description

Function for interpolating values from a source polygon layer to an overlapping (but spatially misaligned) destination polygon layer, using area and/or population weights.

Usage

```
poly2poly_ap(
  poly_from,
  poly_to,
  poly_to_id,
  geo_vor = NULL,
  methodz = "aw",
  char_methodz = "aw",
  pop_raster = NULL,
  varz = NULL,
  pycno_varz = NULL,
  char_varz = NULL,
  char_assign = "biggest_overlap",
  funz = function(x, w) {
     stats::weighted.mean(x, w, na.rm = TRUE)
 },
  seed = 1
)
```

Arguments

poly_from	Source polygon layer. sf object.
poly_to	Destination polygon layer. Must have identical CRS to poly_from. sf object.
poly_to_id	Name of unique ID column for destination polygon layer. Character string.
geo_vor	Voronoi polygons object (used internally by point2poly_tess). sf object.
methodz	Area interpolation method(s). Could be either of "aw" (areal weighting, default) and/or "pw" (population weighting). See "details". Character string or vector of character strings.
char_methodz	Interpolation method(s) for character strings. Could be either of "aw" (areal weighting, default) or "pw" (population weighting). See "details". Character string.
pop_raster	Population raster to be used for population weighting, Must be supplied if methodz="pw". Must have identical CRS to poly_from. raster or SpatRaster object.
varz	Names of numeric variable(s) to be interpolated from source polygon layer to destination polygons. Character string or vector of character strings.
pycno_varz	Names of spatially extensive numeric variables for which the pycnophylactic (mass-preserving) property should be preserved. Character string or vector of character strings.
char_varz	Names of character string variables to be interpolated from source polygon layer to destination polygons. Character string or vector of character strings.

char_assign	Assignment rule to be used for variables specified in char_varz. Could be either "biggest_overlap" (default) or "all_overlap". See "details". Character string or vector of character strings.
funz	Aggregation function to be applied to variables specified in varz. Must take as an input a numeric vector x and vector of weights w. Function or list of functions.
seed	Seed for generation of random numbers. Default is 1. Numeric.

Details

Currently supported integration methods (methodz) include:

- Areal weighting ("aw"). Values from poly_from weighted in proportion to relative area of spatial overlap between source features and geometries of poly_to.
- Population weighting ("pw"). Values from poly_from weighted in proportion to relative population sizes in areas of spatial overlap between source features and geometries of poly_to. This routine uses a third layer (supplied in pop_raster) to calculate the weights.

It is possible to pass multiple arguments to methodz (e.g. methodz=c("aw","pw")), in which case the function will calculate both sets of weights, and append the resulting columns to the output.

Interpolation procedures are handled somewhat differently for numeric and character string variables. For numeric variables supplied in varz, "aw" and/or "pw" weights are passed to the function specified in funz. If different sets of numeric variables are to be aggregated with different functions, both varz and funz should be specified as lists (see examples below).

For character string (and any other) variables supplied in char_varz, "aw" and/or "pw" weights are passed to the assignment rule(s) specified in char_assign. Note that the char_varz argument may include numerical variables, but varz cannot include character string variables.

Currently supported assignment rules for character strings (char_assign) include:

- "biggest_overlap". For each variable in char_varz, the features in poly_to are assigned a single value from overlapping poly_from features, corresponding to the intersection with largest area and/or population weight.
- "all_overlap". For each variable in char_varz, the features in poly_to are assigned all values from overlapping poly_from features, ranked by area and/or population weights (largest-to-smallest) of intersections.

It is possible to pass multiple arguments to char_assign (e.g. char_assign=c("biggest_overlap","all_overlap")), in which case the function will calculate both, and append the resulting columns to the output.

Value

sf polygon object, with variables from poly_from interpolated to the geometries of poly_to.

Examples

```
# Interpolation of a single variable, with area weights
## Not run:
data(clea_deu2009)
data(hex_05_deu)
out_1 <- poly2poly_ap(poly_from = clea_deu2009,
```

```
               poly_to = hex_05_deu,
               poly_to_id = "HEX_ID",
               varz = "to1"
              )

## End(Not run)

# Interpolation of multiple variables, with area weights
## Not run:
out_2 <- poly2poly_ap(
               poly_from = clea_deu2009,
               poly_to = hex_05_deu,
               poly_to_id = "HEX_ID",
               varz = list(
                 c("to1","pvs1_margin"),
                 c("vv1") ),
               pycno_varz = "vv1",
               funz = list(
                 function(x,w){stats::weighted.mean(x,w)},
                 function(x,w){sum(x*w)} ),
               char_varz = c("incumb_pty_n","win1_pty_n")
              )

## End(Not run)

# Interpolation of a single variable, with population weights
## Not run:
data(gpw4_deu2010)
out_3 <- poly2poly_ap(poly_from = clea_deu2009,
                          poly_to = hex_05_deu,
                          poly_to_id = "HEX_ID",
                          varz = "to1",
                          methodz = "pw",
                          pop_raster = gpw4_deu2010)

## End(Not run)

# Interpolation of a single variable, with area and population weights
## Not run:
out_4 <- poly2poly_ap(poly_from = clea_deu2009,
                          poly_to = hex_05_deu,
                          poly_to_id = "HEX_ID",
                          varz = "to1",
                          methodz = c("aw","pw"),
                          pop_raster = gpw4_deu2010)

## End(Not run)
```

`sf2raster` *Convert simple features object into regularly spaced raster*

Description

This function takes in an sf spatial object (polygon or point) and returns a regularly spaced RasterLayer. Reverse translation option allows users to create an sf polygon object from the regularly spaced RasterLayer. This function can also conver the sf object into a cartogram with a user-specified variable name.

Usage

```
sf2raster(
  polyz_from = NULL,
  pointz_from = NULL,
  input_variable = NULL,
  reverse = FALSE,
  poly_to = NULL,
  return_output = NULL,
  return_field = NULL,
  aggregate_function = list(function(x) mean(x, na.rm = TRUE)),
  reverse_function = list(function(x) mean(x, na.rm = TRUE)),
  grid_dim = c(1000, 1000),
  cartogram = FALSE,
  carto_var = NULL,
  message_out = TRUE,
  return_list = FALSE
)
```

Arguments

polyz_from	Source polygon layer. sf object.
pointz_from	Source point layer. sf object.
input_variable	Name of input variable from source layer. Character string.
reverse	Reverse translation from raster layer to sf polygon object (polygon features only). Default is FALSE.
poly_to	Destination polygon layer for reverse conversion. Must be specified if reverse=TRUE. sf object.
return_output	Return output for reverse conversion. Must be specified if reverse=TRUE.
return_field	Return field for reverse conversion. Must be specified if reverse=TRUE.
aggregate_function	Aggregation function to be applied to variables specified in input_variable. Must take as an input a numeric vector x. Function or list of functions. Default is mean.
reverse_function	Aggregation function for reverse conversion. Must be specified if reverse=TRUE. Function or list of functions. Default is mean.
grid_dim	Dimensions of raster grid. Numerical vector of length 2 (number of rows, number of columns). Default is c(1000,1000).
cartogram	Cartogram transformation. Logical. Default is FALSE.

carto_var	Input variable for cartogram transformation. Must be specified if cartogram=TRUE. Character string.
message_out	Print informational messages. Logical. Default is TRUE.
return_list	Return full set of results, including input polygons, centroids and field raster. Default is FALSE. Logical.

Value

If return_list=FALSE (default) and reverse=FALSE (default), returns RasterLayer object, with cell values corresponding to input_variable.

If return_list=TRUE and input layer is polygon, returns a list containing

- "return_output". Output raster, with values corresponding to input_variable. RasterLayer object.
- "return_centroid". Raster of centroids, with values corresponding to input_variable. RasterLayer object.
- "poly_to". Source polygons, with columns corresponding to input_variable and auto-generated numerical ID Field. sf object.
- "return_field". Output raster, with values corresponding to auto-generated numerical ID Field. RasterLayer object.

If return_list=TRUE and input layer is points, returns a list containing

- "return_output". Output raster, with values corresponding to input_variable. RasterLayer object.
- "return_point". Source points, with column corresponding to input_variable.

If reverse=TRUE, returns an sf polygon layer, with columns corresponding to input_variable and auto-generated numerical ID Field.

Examples

```
# Rasterization of polygon layer.
## Not run:
data(clea_deu2009)
out_1 <- sf2raster(polyz_from = utm_select(clea_deu2009),
                   input_variable = "to1")
terra::plot(out_1)

## End(Not run)
# Rasterization of point layer
## Not run:
data(clea_deu2009_pt)
out_2 <- sf2raster(pointz_from = utm_select(clea_deu2009_pt),
                   input_variable = "to1",
                   grid_dim = c(25,25))
terra::plot(out_2)

## End(Not run)
# Cartogram (vote turnout scaled by number of valid votes)
```

```
## Not run:
out_3 <- sf2raster(polyz_from = utm_select(clea_deu2009),
                   input_variable = "to1",
                   cartogram = TRUE,
                   carto_var = "vv1")
terra::plot(out_3)

## End(Not run)
# Polygonization of cartogram raster
## Not run:
out_4a <- sf2raster(polyz_from = utm_select(clea_deu2009),
                    input_variable = "to1",
                    cartogram = TRUE,
                    carto_var = "vv1",
                    return_list = TRUE)
out_4 <- sf2raster(reverse = TRUE,
                   poly_to = out_4a$poly_to,
                   return_output = out_4a$return_output,
                   return_field = out_4a$return_field)
terra::plot(out_4)

## End(Not run)
```

`smart_round` *Smart numerical rounding function*

Description

Function to round numerical values with minimal information loss (e.g. to avoid "0.000" values in tables).

Usage

```
smart_round(x, rnd = 0, return_char = TRUE)
```

Arguments

`x`	Vector of values to be rounded. Numeric.
`rnd`	Requested number of decimal places. Default is 0. Non-negative integer.
`return_char`	Return rounded values as character string? Default is TRUE. Logical.

Details

Rounds the values in its first argument to the specified number of decimal places (default 0). If brute-force rounding produces zero values (e.g. "0.00"), the number of decimal places is expanded to include the first significant digit.

Value

If `return_char=TRUE`, returns a character string of same length as `x`. If `return_char=FALSE`, returns a numerical vector of same length as `x`.

Examples

```
# Round a vector of numbers, character string output (best for tables)
## Not run:
out_1 <- smart_round(c(.0013,2.3,-1,pi),rnd=2)
out_1

## End(Not run)

# Round a vector of numbers, numerical output
## Not run:
out_2 <- smart_round(c(.0013,2.3,-1,pi),rnd=2,return_char=FALSE)
out_2

## End(Not run)
```

`SUNGEO` SUNGEO

Description

Sub-National Geospatial Data Archive System: Geoprocessing Toolkit

Details

See the README on [GitHub](https://github.com/zhukovyuri/SUNGEO#readme)

`update_bbox` *Update bounding box of sf object*

Description

Function to update the coordinates of the bounding box of sf vector data objects (e.g. after cropping or subsetting).

Usage

```
update_bbox(sfobj)
```

Arguments

`sfobj`	Layer to be updated. `sf` object.

Value

sf object, with corrected bounds.

Examples

```
# Update bbox for subset of sf object
## Not run:
data(clea_deu2009)
out_1 <- update_bbox(clea_deu2009[clea_deu2009$cst_n%in%c("Berlin"),])
out_1

# Bounding box of full dataset
data.table::as.data.table(clea_deu2009)[,sf::st_bbox(geometry)]

# Bounding box of subset (incorrect)
data.table::as.data.table(clea_deu2009)[cst_n%in%c("Berlin"),sf::st_bbox(geometry)]

# Corrected bounding box
data.table::as.data.table(out_1)[,sf::st_bbox(geometry)]

## End(Not run)
```

---

utm_select *Automatically convert geographic (degree) to planar coordinates (meters)*

Description

Function to automatically convert simple feature, spatial and raster objects with geographic coordinates (longitude, latitude / WGS 1984, EPSG:4326) to planar UTM coordinates. If the study region spans multiple UTM zones, defaults to Albers Equal Area.

Usage

```
utm_select(x, max_zones = 5, return_list = FALSE)
```

Arguments

x	Layer to be reprojected. sf, sp, SpatRaster or RasterLayer object.
max_zones	Maximum number of UTM zones for single layer. Default is 5. Numeric.
return_list	Return list object instead of reprojected layer (see Details). Default is FALSE. Logical.

Details

Optimal map projection for the object x is defined by matching its horizontal extent with that of the 60 UTM zones. If object spans multiple UTM zones, uses either the median zone (if number of zones is equal to or less than max_zones) or Albers Equal Area projection with median longitude as projection center (if number of zones is greater than max_zones).

Value

Re-projected layer. `sf` or `RasterLayer` object, depending on input.

If `return_list=TRUE`, returns a list object containing

- "x_out". The re-projected layer. `sf` or `RasterLayer` object, depending on input.
- "proj4_best".proj4string of the projection. Character string.

Examples

```
# Find a planar projection for an unprojected (WSG 1984) hexagonal grid of Germany
## Not run:
data(hex_05_deu)
out_1 <- utm_select(hex_05_deu)

## End(Not run)
# Find a planar projection for a raster
## Not run:
data(gpw4_deu2010)
out_2 <- utm_select(gpw4_deu2010)

## End(Not run)
```

Index