Package: ColOpenData (via r-universe)

September 26, 2024

Title Download Colombian Demographic, Climate and Geospatial Data

Version 0.3.0

Description Downloads wrangled Colombian socioeconomic, geospatial,population and climate data from DANE <https://www.dane.gov.co/> (National Administrative Department of Statistics) and IDEAM <https://ideam.gov.co> (Institute of Hydrology, Meteorology and Environmental Studies). It solves the problem of Colombian data being issued in different web pages and sources by using functions that allow the user to select the desired database and download it without having to do the exhausting acquisition process.

License MIT + file LICENSE

URL https://github.com/epiverse-trace/ColOpenData, https://epiverse-trace.github.io/ColOpenData/

BugReports https://github.com/epiverse-trace/ColOpenData/issues

Depends R (>= 3.3.0)

Imports checkmate, config, dplyr, magrittr, rlang, sf, stringdist, tidyr, utils

Suggests ggplot2, knitr, leaflet, rmarkdown, spelling, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/Needs/website epiverse-trace/epiversetheme

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Repository https://epiverse-trace.r-universe.dev

RemoteUrl https://github.com/epiverse-trace/ColOpenData

RemoteRef HEAD

RemoteSha 9cef612ad7f3b88398f91bc8bfdc87a50c152dad

Contents

`aggregate_climate` *Aggregate climate data for different frequencies*

Description

Aggregate time series downloaded climate data to day, month or year. Only observations under the tags `TSSM_CON`, `TMN_CON`, `TMX_CON`, `PTPM_CON`, and `BSHG_CON` can be aggregated, since are the ones where methodology for aggregation is explicitly provided by the source.

Usage

```
aggregate_climate(climate_data, frequency)
```

Arguments

climate_data	data.frame obtained from download functions. Only observations under the same tag can be aggregated.
frequency	character with the aggregation frequency: ("day", "month" or "year").

Value

data.frame object with the aggregated data.

Examples

```
lat <- c(4.172817, 4.172817, 4.136050, 4.136050, 4.172817)
lon <- c(-74.749121, -74.686169, -74.686169, -74.749121, -74.749121)
polygon <- sf::st_polygon(x = list(cbind(lon, lat)))
geometry <- sf::st_sfc(polygon)
roi <- sf::st_as_sf(geometry)
ptpm <- download_climate_geom(roi, "2022-11-01", "2022-12-31", "PTPM_CON")
monthly_ptpm <- aggregate_climate(ptpm, "month")
head(monthly_ptpm)
```

---

climate_tags *climate_tags*

Description

dictionary for climate tags

Usage

```
data(climate_tags)
```

Format

An object of class list of length 2.

Details

Dictionary for climate tags

code_to_name_dep *Retrieve departments' DIVIPOLA names from codes*

Description

Retrieve departments' DIVIPOLA official names from their DIVIPOLA codes.

Usage

```
code_to_name_dep(department_code)
```

Arguments

department_code	character vector with the DIVIPOLA codes of the departments.

Value

character vector with the DIVIPOLA name of the departments.

Examples

```
dptos <- c("73", "05", "11")
code_to_name_dep(dptos)
```

code_to_name_mun *Retrieve municipalities' DIVIPOLA names from codes*

Description

Retrieve municipalities' DIVIPOLA official names from their DIVIPOLA codes.

Usage

```
code_to_name_mun(municipality_code)
```

Arguments

municipality_code	character vector with the DIVIPOLA codes of the municipalities.

Value

character vector with the DIVIPOLA name of the municipalities.

Examples

```
mpios <- c("73001", "11001", "05615")
code_to_name_mun(mpios)
```

`datasets_list` *datasets_list*

Description

list of datasets description in English and Spanish

Usage

```
data(datasets_list)
```

Format

An object of class `list` of length 2.

Details

List containing both datasets description in English and Spanish

---

`divipola_table` *Retrieve DIVIPOLA table*

Description

Retrieve DIVIPOLA table including departments and municipalities. DIVIPOLA codification includes individual codes for each department and municipality following the political and administrative division.

Usage

```
divipola_table()
```

Value

`data.frame` object with DIVIPOLA table.

Examples

```
divipola <- divipola_table()
```

download_climate *Download climate from named geometry (municipality or department)*

Description

Download climate data from stations contained in a municipality or department. This data is retrieved from local meteorological stations provided by IDEAM.

Usage

```
download_climate(code, start_date, end_date, tag)
```

Arguments

code	character with the DIVIPOLA code for the area (2 digits for departments and 5 digits for municipalities).
start_date	character with the first date to consult in the format "YYYY-MM-DD". (First available date is "1920-01-01").
end_date	character with the last date to consult in the format "YYYY-MM-DD". (Last available date is "2023-05-31").
tag	character containing climate tag to consult. Please use cliamte_tags() to check IDEAM tags.

Value

data.frame object with observations from the stations in the area.

Examples

```
ptpm <- download_climate("73148", "2021-11-14", "2021-11-20", "PTPM_CON")
head(ptpm)
```

download_climate_geom *Download climate data from geometry*

Description

Download climate data from stations contained in a Region of Interest (ROI/geometry). This data is retrieved from local meteorological stations provided by IDEAM.

Usage

```
download_climate_geom(geometry, start_date, end_date, tag)
```

Arguments

geometry	sf object containing the geometry for a given ROI. The geometry can be either a POLYGON or MULTIPOLYGON.
start_date	character with the first date to consult in the format "YYYY-MM-DD". (First available date is "1920-01-01").
end_date	character with the last date to consult in the format "YYYY-MM-DD". (Last available date is "2023-05-31").
tag	character containing climate tag to consult.

Value

data.frame object with observations from the stations in the area.

Examples

```
lat <- c(4.172817, 4.172817, 4.136050, 4.136050, 4.172817)
lon <- c(-74.749121, -74.686169, -74.686169, -74.749121, -74.749121)
polygon <- sf::st_polygon(x = list(cbind(lon, lat)))
geometry <- sf::st_sfc(polygon)
roi <- sf::st_as_sf(geometry)
ptpm <- download_climate_geom(roi, "2022-11-14", "2022-11-20", "PTPM_CON")
head(ptpm)
```

---

download_climate_stations

Download climate data from stations

Description

Download climate data from IDEAM stations by individual codes.This data is retrieved from local meteorological stations provided by IDEAM.

Usage

```
download_climate_stations(stations, start_date, end_date, tag)
```

Arguments

stations	data.frame containing the stations' codes and location. data.frame must be retrieved from the function stations_in_roi()
start_date	character with the first date to consult in the format "YYYY-MM-DD". (First available date is "1920-01-01").
end_date	character with the last date to consult in the format "YYYY-MM-DD". (Last available date is "2023-05-31").
tag	character containing climate tag to consult.

Value

`data.frame` object with observations from the stations in the area.

Examples

```
lat <- c(4.172817, 4.172817, 4.136050, 4.136050, 4.172817)
lon <- c(-74.749121, -74.686169, -74.686169, -74.749121, -74.749121)
polygon <- sf::st_polygon(x = list(cbind(lon, lat)))
geometry <- sf::st_sfc(polygon)
roi <- sf::st_as_sf(geometry)
stations <- stations_in_roi(roi)
ptpm <- download_climate_stations(
  stations, "2022-11-14", "2022-11-20", "PTPM_CON"
)
head(ptpm)
```

---

download_demographic *Download demographic dataset*

Description

This function downloads demographic datasets from the National Population and Dwelling Census (CNPV) of 2018.

Usage

```
download_demographic(dataset)
```

Arguments

`dataset`	character with the demographic dataset name. Please use `list_datasets("demographic", "EN")` or `list_datasets("demographic", "ES")` to check available datasets.

Value

`data.frame` object with downloaded data.

Examples

```
house_under_15 <- download_demographic("DANE_CNPVH_2018_1HD")
head(house_under_15)
```

`download_geospatial` *Download geospatial dataset*

Description

This function downloads geospatial datasets from the National Geostatistical Framework at different levels of spatial aggregation. These datasets include a summarized version of the National Population and Dwelling Census (CNPV) with demographic and socioeconomic information for each spatial unit.

Usage

```
download_geospatial(
  spatial_level,
  simplified = TRUE,
  include_geom = TRUE,
  include_cnpv = TRUE
)
```

Arguments

`spatial_level`	character with the spatial level to be consulted: • `"DPTO"` or `"department"`: Department. • `"MPIO"` or `"municipality"`: Municipality. • `"MPIOCL"` or `"municipality_class"`: Municipality including class. • `"SETU"` or `"urban_sector"`: Urban Sector. • `"SETR"` or `"rural_sector"`: Rural Sector. • `"SECU"` or `"urban_section"`: Urban Section. • `"SECR"` or `"rural_section"`: Rural Section. • `"ZU"` or `"urban_zone"`: Urban Zone. • `"MZN"` or `"block"`: Block.
`simplified`	logical for indicating if the downloaded spatial data should be a simplified version of the geometries. Simplified versions are lighter but less precise, and are only recommended for easier applications like plots. Default is `TRUE`.
`include_geom`	logical for including (or not) the spatial geometry. Default is `TRUE`. If `TRUE`, the function will return an `"sf"` `data.frame`.
`include_cnpv`	logical for including (or not) CNPV demographic and socioeconomic information. Default is `TRUE`.

Value

`data.frame` object with downloaded data.

Examples

```
departments <- download_geospatial("department")
head(departments)
```

`download_pop_projections` *Download population projections*

Description

This function downloads population projections and back projections taken from the National Population and Dwelling Census of 2018 (CNPV), adjusted after COVID-19. Available years are different for each spatial level:

- `"national"`: 1950 - 2070.
- `"national"` with sex: 1985 - 2050.
- `"department"`: 1985 - 2050.
- `"department"` with sex: 1985 - 2050.
- `"municipality"`: 1985 - 2035.
- `"municipality"` with sex: 1985 - 2035.
- `"municipality"` with sex and ethnic groups: 2018 - 2035.

Usage

```
download_pop_projections(
  spatial_level,
  start_year,
  end_year,
  include_sex = FALSE,
  include_ethnic = FALSE
)
```

Arguments

`spatial_level`	character with the spatial level to be consulted. Can be either `"national"`, `"department"` or `"municipality"`.
`start_year`	numeric with the start year to be consulted.
`end_year`	numeric with the end year to be consulted.
`include_sex`	logical for including (or not) division by sex. Default is `FALSE`.
`include_ethnic`	logical for including (or not) division by ethnic group (only available for `"municipality"`). Default is `FALSE`.

Value

`data.frame` object with downloaded data.

Examples

```
pop_proj <- download_pop_projections("national", 2020, 2030)
head(pop_proj)
```

---

geospatial_dictionaries *geospatial_dictionaries*

Description

dictionaries of variables presented in geospatial datasets

Usage

```
data(geospatial_dictionaries)
```

Format

An object of class `list` of length 2.

Details

Dictionaries for geospatial datasets in English and Spanish

---

geospatial_dictionary *Download data dictionaries*

Description

Retrieve geospatial data dictionaries to understand internal tags and named columns. Dictionaries are available in English and Spanish.

Usage

```
geospatial_dictionary(spatial_level, language = "ES")
```

Arguments

spatial_level	character with the spatial level to be consulted: • "DPTO" or "department": Department. • "MPIO" or "municipality": Municipality. • "MPIOCL" or "municipality_class": Municipality including class. • "SETU" or "urban_sector": Urban Sector. • "SETR" or "rural_sector": Rural Sector. • "SECU" or "urban_section": Urban Section. • "SECR" or "rural_section": Rural Section. • "ZU" or "urban_zone": Urban Zone. • "MZN" or "block": Block.
language	character with the language of the dictionary variables ("EN" or "ES". Default is "ES".

Value

data.frame object with geospatial data dictionary.

Examples

```
dict <- geospatial_dictionary("setu", "EN")
head(dict)
```

---

get_climate_tags *List climate (IDEAM) tags*

Description

Retrieve available climate tags to be consulted. The list is only available in Spanish.

Usage

```
get_climate_tags(language = "ES")
```

Arguments

language	character with the language of the tags ("EN" or "ES". Default is "ES".

Value

data.frame object with available climate tags.

Examples

```
dict <- get_climate_tags("ES")
head(dict)
```

list_datasets *Download list of available datasets*

Description

List all available datasets by name, including group, source, year, level, category and description.

Usage

```
list_datasets(module = "all", language = "ES")
```

Arguments

module	character with module to be consulted ("demographic", "geospatial" or "climate"). Default is "all".
language	character with the language of dataset details ("EN" or "ES". Default is "ES".

Value

data.frame object with the available datasets.

Examples

```
list <- list_datasets("geospatial", "EN")
head(list)
```

look_up *Filter list of available datasets based on keywords given by the user*

Description

List available datasets containing user-specified keywords in their descriptions.

Usage

```
look_up(keywords, module = "all", logic = "or", language = "EN")
```

Arguments

keywords	character or vector of characters to be look up in the description.
module	character with module to be consulted ("demographic", "geospatial", "climate"). Default is "all".
logic	A character string specifying the matching logic. Can be either "or" or "and". Default is "or": • logic = "or": Matches rows containing at least one of the specified keywords in their descriptions. • logic = "and": Matches rows containing all of the specified keywords in their descriptions.
language	character with the language of the keywords ("EN" or "ES". Default is "EN".

Value

data.frame object with the available datasets containing information related to the consulted keywords.

Examples

```
found <- look_up(c("sex", "age"), "demographic", "and", "EN")
head(found)
```

---

merge_geo_demographic *Match and merge geospatial and demographic datasets*

Description

This function adds the key information of a demographic dataset to a geospatial dataset based on the spatial aggregation level. Since the smallest level of spatial aggregation present in the demographic datasets is municipality, this function can only merge with geospatial datasets that present municipality or department level.

Usage

```
merge_geo_demographic(demographic_dataset, simplified = TRUE)
```

Arguments

demographic_dataset	character with the demographic dataset name. Please use list_datasets("demographic", "EN") or list_datasets("demographic", "ES") to check available datasets.
simplified	logical for indicating if the downloaded spatial data should be a simplified version of the geometries. Simplified versions are lighter but less precise, and are recommended for easier applications like plots. Default is TRUE.

Value

`data.frame` object with the merged data.

Examples

```
merged <- merge_geo_demographic("DANE_CNPVV_2018_9VD", TRUE)
head(merged)
```

`name_to_code_dep` *Retrieve departments' DIVIPOLA codes from names*

Description

Retrieve departments' DIVIPOLA codes from their names.

Usage

```
name_to_code_dep(department_name)
```

Arguments

`department_name`
: character vector with the names of the departments.

Value

character vector with the DIVIPOLA codes of the departments.

Examples

```
dptos <- c("Tolima", "Huila", "Amazonas")
name_to_code_dep(dptos)
```

name_to_code_mun *Retrieve municipalities' DIVIPOLA codes from names*

Description

Retrieve municipalities' DIVIPOLA codes from their names. Since there are municipalities with the same names in different departments, the input must include two vectors: one for the departments and one for the municipalities in said departments. If only one department is provided, it will try to match all municipalities in the second vector inside that department. Otherwise, the vectors must be the same length.

Usage

```
name_to_code_mun(department_name, municipality_name)
```

Arguments

department_name	character vector with the names of the departments containing the municipalities.
municipality_name	character vector with the names of the municipalities.

Value

character vector with the DIVIPOLA codes of the municipalities.

Examples

```
dptos <- c("Huila", "Antioquia")
mpios <- c("Pitalito", "Turbo")
name_to_code_mun(dptos, mpios)
```

name_to_standard_dep *Translate department names to official departments' DIVIPOLA names*

Description

Department names are usually manually input, which leads to multiple errors and lack of standardization. This functions translates department names to their respective official names from DIVIPOLA.

Usage

```
name_to_standard_dep(department_name)
```

Arguments

department_name
: character vector with the names to be translated.

Value

character vector with the DIVIPOLA name of the departments.

Examples

```
dptos <- c("Bogota DC", "San Andres")
name_to_standard_dep(dptos)
```

---

name_to_standard_mun *Translate municipality names to official municipalities' DIVIPOLA names*

Description

Municipality names are usually manually input, which leads to multiple errors and lack of standardization. This functions translates municipality names to their respective official names from DIVIPOLA.

Usage

```
name_to_standard_mun(department_name, municipality_name)
```

Arguments

department_name
: character vector with the names of the departments containing the municipalities.

municipality_name
: character vector with the names to be translated.

Value

character vector with the DIVIPOLA name of the municipalities.

Examples

```
dptos <- c("Bogota", "Tolima")
mpios <- c("Bogota DC", "CarmendeApicala")
name_to_standard_mun(dptos, mpios)
```

stations_in_roi *Stations in region of interest*

Description

Download and filter climate stations contained inside a region of interest (ROI).

Usage

```
stations_in_roi(geometry)
```

Arguments

geometry	sf object containing the geometry for a given ROI. The geometry can be either a POLYGON or MULTIPOLYGON.

Value

data.frame object with the stations contained inside the consulted geometry.

Examples

```
lat <- c(5.166278, 5.166278, 4.982247, 4.982247, 5.166278)
lon <- c(-75.678072, -75.327859, -75.327859, -75.678072, -75.678072)
polygon <- sf::st_polygon(x = list(cbind(lon, lat)))
geometry <- sf::st_sfc(polygon)
roi <- sf::st_as_sf(geometry)
stations <- stations_in_roi(roi)
head(stations)
```

Index