Package 'stlocationqc'

December 28, 2018

Title C3S Quality Control of Meteorological Stations Location

Version 0.1.0

Affiliation FCiencias.ID - Sciences Association for Research and Development (Faculty of Sciences University of Lisbon)

Description Metadata quality control functions developed for the Copernicus Data Rescue Service.

URL https://github.com/C3S-Data-Rescue-Lot1-WP3/stlocationqc

BugReports https://github.com/mcmventura/stlocationqc/issues

Depends R (>= 3.3.0)

License GPL-3

Encoding UTF-8

LazyData true

Imports raster, rgdal (>= 1.2), rgeos, sp, stats, utils

Suggests knitr

RoxygenNote 6.1.0

NeedsCompilation no

Author Clara Ventura [aut, cre]

Maintainer Clara Ventura <mcmventura@fc.ul.pt>

R topics documented:

compare_country 2
countries_polys 3
eraclim_uao_fp 4
get_country 4
get_country_shoreline 5
get_lon180 7
get_sea 8
ispd 9
order_data 9
seas_polys 11
stlocationqc 11
test_geocoord 12

Index **13**

compare_country *Compares the Given Country Names with the Determined Country Names.*

Description

Compares the given country names with the names assigned by the function get_country, get_country_shoreline or get_sea.

Usage

```
compare_country(countries_gv = NULL, countries, countries_sh = NULL, seas =
NULL, miss_seas = NULL, excl_coords = NULL)
```

Arguments

countries_gv	data frame with the coordinates and the given country names **lon | lat | country_gv**.
countries	data frame output of get_country.
countries_sh	data frame output of get_country_shoreline.
seas	data frame output of get_sea.
miss_seas	data frame output of get_sea.
excl_coords	data frame output of get_lon180 or test_geocoord.

Details

Input:

- A data frame with the coordinates and the given country names or a text file without header, the longitude in the first column, the latitude in the second column, both in decimal degrees. The given country name goes in the third column. The columns are separated by tabs and missing values must be codified as 'NA' in all fields.
- The output of get_country - **'countries'** -, get_country_shoreline - **'countries_sh'** -, get_sea - **'seas'** and **'miss_seas'** - and get_lon180 or test_geocoord - **'excl_coords'**. However, only the 'countries' data frame is compulsory because all those data frames may not exist.

Output:

- Several text files, until the maximum of 9, which have the following header: **country_gv |id | lon | lat | country | sovereignt | adm0_a3 | name_de | name_es | name_fr | name_pt**. Those files split the list of points by types of differences between the given and the determinated country name:
  - Missing given or determined country name
  - Equal country name
  - Given country name equals to sovereignty
  - Given country name in another language
  - Given country name in upper case
  - Given country name with some equal words - English or another language
  - Set of 3 letters equals at the beginning or at the end of the words
  - Points that fall into the sea
  - Different country name

Examples

```
## Not run:
##
## First run
test_geocoord(coords = eraclim_uao_fp)
## Then run sequentially until all points have names assigned
get_country(icoords = coords_ok)
get_country_shoreline(icoords = miss_countries, tol)
get_sea(icoords = miss_countries_sh)
## And finally
compare_country(countries_gv = eraclim_uao_fp, countries = countries,
countries_sh = countries_sh, seas = seas, miss_seas = miss_seas, excl_coords
= excl_coords)

## End(Not run)
```

`countries_polys` *Downloads Spatial Data with the World Countries Polygons and Saves them as R Objects.*

Description

Downloads from Natural Earth two shapefiles - 10 m and 50 m precision - with the world countries polygons and transforms them into two SpatialPolygonsDataFrame that are saved in a folder called 'polys', inside the user's working directory. The data that they contain is necessary for the functions `get_country` and `get_country_shoreline`.

Usage

```
countries_polys()
```

Details

The two SpatialPolygonsDataFrame are saved in the working directory, inside the 'polys' folder as .rda files and they are loaded from there when running the functions `get_country` and `get_country_shoreline`.

Shapefiles:

- http://www.naturalearthdata.com/http//www.naturalearthdata.com/download/10m/cultural/ne_10m_admin_0_countries.zip
- http://www.naturalearthdata.com/http//www.naturalearthdata.com/download/50m/cultural/ne_50m_admin_0_countries.zip

Shapefiles metadata:

- http://www.naturalearthdata.com/downloads/10m-cultural-vectors/10m-admin-0-countries/
- http://www.naturalearthdata.com/downloads/50m-cultural-vectors/50m-admin-0-countries-2/

References

Made with Natural Earth. Free vector and raster map data @ naturalearthdata.com.

eraclim_uao_fp *ERACLIM Upper-air stations*

Description

A dataset of coordinate pairs with the longitude in the range (-180 to +180) degrees and respective country, extracted from the *ERACLIM Upper-air Observations - Fixed Platforms* inventory.

Usage

```
eraclim_uao_fp
```

Format

A data frame with 6444 rows and 3 variables:

lon longitude in decimal degrees with three decimal places, that should be in the range (-180 to +180) but has out of bounds values and missing values

lat latitude in decimal degrees with three decimal places, that has some missing values

country_gv country name given in the inventory, that is written in different languages and has non ASCII characters

Source

http://eraclim-global-registry.rd.ciencias.ulisboa.pt/upperairdata.php

get_country *Determines Country Names for Coordinate Points Located in the Continent.*

Description

Given a list of geographic coordinates in the continent, determines the country name for each coordinate point. First uses a SpatialPolygonsDataFrame with 10 m precision to calculate the country names. Then if there are still points without a name assigned, uses a SpatialPolygonsDataFrame with 50 m precision, thus trying to calculate the name for less accurate points.

Usage

```
get_country(icoords)
```

Arguments

icoords	data frame with three columns: **id | lon | lat** where: 'id' is the row identifier for the coordinates in the original list of coordinates, 'lon' is the longitude in the range (-180, +180) and 'lat' is the latitude. Both coordinates are in decimal degrees.

Details

Input:

- The output of test_geocoord - **'coords_ok'** -, or the output of get_lon180 - **'coords_lon180'**, i.e., a data frame with three columns: **id | lon | lat**. To use the present function it is necessary that the coordinates do not have any errors or missing values. This requires previous running of test_geocoord or get_lon180 that in addition to testing the coordinates, creates the 'id' which is the row identifier for the coordinates in the original list of coordinates. After running sequentially get_country, get_country_shoreline and get_sea to assign the geographic names, the 'id' is necessary to display the coordinates in the initial order at the end of the process (which consists of a sucessive elimination of missing names), with or without all the names assigned.

Output:

- A text file with all the points for which a country name was determined. That file has the following header: id | lon | lat | country | sovereignt | adm0_a3 | name_de | name_es | name_fr | name_pt.
- A text file with the missing country names, if they exist.
- A .RData file with the output data frame(s). The data frame **'countries'** has the header **id | lon | lat | country | sovereignt | iso3 | subregion | continent**. The data frame **'miss_countries'** has the header **id | lon | lat** and it is the input of get_country_shoreline to determine the missing country names or the input of get_sea, if the user wants to verify if the still unamed points fall into to the sea and calculate the sea name were they are located.

References

Made with Natural Earth. Free vector and raster map data @ naturalearthdata.com.

Examples

```
## Not run:
## First run
get_lon180(coords = ispd)
## Or
test_geocoord(coords = eraclim_uao_fp)
## Then run
get_country(icoords = coords_ok)

## End(Not run)
```

---

get_country_shoreline *Determines Country Names for Coordinate Points Located in the Shoreline.*

Description

Determines country names for a list of coordinate points which are supposed to be located on land but without country names assigned by the function get_country because probably they are very close to the shoreline and have lack of precision. Due to this fact when the user runs get_sea

over that points they get ocean/sea names assigned. The present function uses a SpatialPolygonsDataFrame with 50 m precision and a buffer zone around the coordinate points. The buffer represents the tolerance and is given by the user. If the tolerance is 500 m, for example, and the point gets a country name, means that it belongs to that country with an error until 500 m towards the sea.

Usage

```
get_country_shoreline(icoords, tol)
```

Arguments

icoords	data frame with three columns: **id | lon | lat** where: 'id' is the row identifier for the coordinates in the original list of coordinates, 'lon' is the longitude in the range (-180, +180) and 'lat' is the latitude. Both coordinates are in decimal degrees.
tol	is the tolerance in meters. By default, tol = 500 meters.

Details

Input:

- The output of `get_country` - **'miss_countries'** -, i.e., a data frame with three columns: **id | lon | lat**.

Output:

- A text file with the points for which a country name was determined, if they exist. That file has the following header: id | lon | lat | country | sovereignt | adm0_a3 | name_de | name_es | name_fr | name_pt.
- A text file with the missing country names, if they exist.
- A .RData file with the output data frame(s). The data frame **'countries_sh'** has the header **id | lon | lat | country | sovereignt | iso3 | subregion | continent**. The data frame **'miss_countries_sh'** has the header **id | lon | lat** and it is the input of `get_sea` to determine the sea name were those points are located.

References

Made with Natural Earth. Free vector and raster map data @ naturalearthdata.com.

Examples

```
## Not run:
##
## First run
get_lon180(coords = ispd)
## Or
test_geocoord(coords = eraclim_uao_fp)
## Then run sequentially
get_country(icoords = coords_ok)
get_country_shoreline(icoords = miss_countries, tol)

## End(Not run)
```

`get_lon180` *Tests the Geographic Coordinates and Transforms the Longitude from (0, 360) to (-180, +180).*

Description

Given a list of geographic coordinates, first tests the coordinates for out of bounds values and missing values. Then transforms the longitude from (0, 360) to (-180, +180) degrees, considering 6 significative digits.

Usage

```
## If there are the coordinates data frame
get_lon180(coords)
##
## If there are the coordinates text file
get_lon180(coords = NULL)
```

Arguments

`coords`	data frame with the geographic coordinates in decimal degrees, the longitude in the first column, the latitude in the second column and the column names: **lon | lat** (other columns can exist, however are unnecessary for this function).

Details

Input:

- A **text file** without header with the geographic coordinates in decimal degrees: the longitude in the first column and the latitude in the second column, separated by tabs (other columns could exist, however are unnecessary for this function). Missing values must be codified as 'NA' in all fields.
- Or a **data frame** with that same format and the column names: **lon | lat**.

Output:

- A text file named 'coords_lon180' with three columns id | lon | lat, where: 'id' is the row identifier for the coordinates in the original list of coordinates, 'lon' is the longitude in the range (-180, +180) and 'lat' is the latitude. Both coordinates are in decimal degrees.
- If there are errors, the function writes a text file with the coordinate pairs containing at least one coordinate out of bounds.
- If there are missing coordinates, the function writes a text file with the coordinate pairs containing at least one missing coordinate.
- A .RData file with the output data frame(s) that has/have the column names: **id | lon | lat**. The data frame **'coords_lon180'** can be used as input parameter of `get_country` to determine the country names. Eventually is created the data frame **'excl_coords'** with the erroneous and/or missing coordinates.

Examples

```
## Not run:
get_lon180(coords = ispd)

## End(Not run)
```

`get_sea` *Determines Sea Names for Points Located in a Marine Area.*

Description

Given a list of geographic coordinates in marine areas of the world, determines the geographical name of the sea, ocean, bay, gulf, fjord, etc. were each one of the points is located. Uses a SpatialPolygonsDataFrame with 10 m precision containing the polygons and sea names, which covers most of the marine territories on Earth.

Usage

```
get_sea(icoords)
```

Arguments

`icoords`	data frame with three columns: **id | lon | lat** where: 'id' is the row identifier for the coordinates in the original list of coordinates, 'lon' is the longitude in the range (-180, +180) and 'lat' is the latitude. Both coordinates are in decimal degrees.

Details

Input:

- The output of `get_country` - **'miss_countries'** -, or the output of `get_country_shoreline`, - **'miss_countries_sh'** -, i.e., a data frame with three columns: **id | lon | lat**.

Output:

- A text file with the points for which a sea name was determined, if they exist. That file has the following header: **id | lon | lat | sea**.
- A text file with the missing sea names, if they exist.
- A .RData file with the output data frame(s). The data frame **'seas'** has the header **id | lon | lat | sea**. The eventual data frame **'miss_seas'** has the header **id | lon | lat**.

References

Made with Natural Earth. Free vector and raster map data @ naturalearthdata.com.

Examples

```
## Not run:
## First run
get_lon180(coords = ispd)
## Or
test_geocoord(coords = eraclim_uao_fp)
## Then run sequentially
get_country(icoords = coords_ok)
get_country_shoreline(icoords = miss_countries, tol)
get_sea(icoords = miss_countries_sh)

## End(Not run)
```

---

ispd *ISPD stations*

Description

A dataset of coordinate pairs with the longitude in the range (0 to 360) degrees, extracted from the list of meteorological stations belonging to *ISPD - International Surface Pressure Databank*.

Usage

```
ispd
```

Format

A data frame with 83814 rows and 2 variables:

lon longitude in decimal degrees (0 to 360) with two decimal places

lat latitude in decimal degrees (-90 to +83.65) with two decimal places

References

http://reanalyses.org/observations/international-surface-pressure-databank

---

order_data *Reorder the List of Coordinates by the Initial Order.*

Description

After running sequentially `get_lon180` or `test_geocoord` and `get_country`, `get_country_shoreline` and/or `get_sea` to assign the geographic names, this function uses the output of those. It combines by row the data frames with a description - geographic name or indication of some problem with the coordinates (if that occurs) - and orders the data by the same order as in the input text file or data frame (given by the user, with the list of coordinates).

Usage

```
order_data(countries, countries_sh = NULL, seas = NULL, miss_seas =
NULL, excl_coords = NULL)
```

Arguments

countries	data frame output of get_country.
countries_sh	data frame output of get_country_shoreline.
seas	data frame output of get_sea.
miss_seas	data frame output of get_sea.
excl_coords	data frame output of get_lon180 or test_geocoord.

Details

Input:

- The output of get_country - **'countries'** -, get_country_shoreline - **'countries_sh'** -, get_sea - **'seas'** and **'miss_seas'** - and get_lon180 or test_geocoord - **'excl_coords'**. However, only the 'countries' data frame is compulsory because all those data frames may not exist.

Output:

- A text file with the coordinate pairs by the inicial order and the respective geographic name. That file has the following header: **id | lon | lat | geo_name**. If a country or sea name could not be assigned because of the polygon's data set lack of coverage, the description in the field 'geo_name' will be 'MISSING_NAME'. If a name could not be assigned because the coordinates are missing or out of bounds, the description will be 'ERRONEOUS-MISSING_COORDS'.

Examples

```
## Not run:
##
## First run
get_lon180(coords = ispd)
## Or
test_geocoord(coords = eraclim_uao_fp)
## Then run sequentially:
get_country(icoords = coords_ok)
get_country_shoreline(icoords = miss_countries, tol)
get_sea(icoords = miss_countries_sh)
## And finally
## For the ISPD case
order_data(countries = countries, countries_sh = countries_sh, seas = seas)
## For the ERACLIM_UAO_FP
order_data(countries = countries, countries_sh = countries_sh, seas = seas,
excl_coords = excl_coords)

## End(Not run)
```

`seas_polys` *Downloads Spatial Data with the World Seas Polygons and Saves it as a R Object.*

Description

Downloads from Natural Earth a shapefile with the world seas polygons of 10 m precision and then transforms it into a SpatialPolygonsDataFrame that is saved in a folder called 'polys', inside the user's working directory. The data it contains is necessary for the function `get_sea` to work.

Usage

```
seas_polys()
```

Details

The SpatialPolygonsDataFrame is saved in the working directory, inside the 'polys' folder as a .rda file and it is loaded from there when running the function `get_sea`.

Shapefile:

- http://www.naturalearthdata.com/http//www.naturalearthdata.com/download/10m/physical/ne_10m_geography_marine_polys.zip

Shapefile metadata:

- http://www.naturalearthdata.com/downloads/10m-physical-vectors/10m-physical-labels/

References

Made with Natural Earth. Free vector and raster map data @ naturalearthdata.com.

`stlocationqc` *stlocationqc: C3S Quality Control of Meteorological Stations Location*

Description

stlocationqc: C3S Quality Control of Meteorological Stations Location

stlocationqc functions

`countries_polys`, `seas_polys`, `get_lon180`, `test_geocoord`, `get_country`, `get_country_shoreline`, `get_sea`, `order_data` and `compare_country`.

`test_geocoord` *Tests the Geographic Coordinates.*

Description

Given a list of geographic coordinates, tests if the longitude belongs to (-180, +180) degrees and if the latitude belongs to (-90, +90) degrees. Also tests the coordinates for missing values.

Usage

```
## If there are the coordinates data frame
test_geocoord(coords)
##
## If there are the coordinates text file
test_geocoord(coords = NULL)
```

Arguments

`coords`	data frame with the geographic coordinates in decimal degrees, the longitude in the first column, the latitude in the second column and the column names: **lon | lat** (other columns could exist, however are unnecessary for this function).

Details

Input:

- A **text file** without header with the geographic coordinates in decimal degrees: the longitude in the first column and the latitude in the second column, separated by tabs (other columns could exist, however are unnecessary for this function). Missing values must be codified as 'NA' in all fields.
- Or a **data frame** with that same format and the column names: **lon | lat**.

Output:

- A text file named 'coords_ok' with three columns id | lon | lat, where: 'id' is the row identifier for the coordinates in the original list of coordinates, 'lon' is the longitude in the range (-180, +180) and 'lat' is the latitude. Both coordinates are in decimal degrees.
- If there are errors, the function writes a text file with the coordinate pairs containing at least one coordinate out of bounds.
- If there are missing coordinates, the function writes a text file with the coordinate pairs containing at least one missing coordinate.
- A .RData file with the output data frame(s) that has/have the column names: **id | lon | lat**. The data frame **'coords_ok'** can be used as input parameter of `get_country` to determine the country names. Eventually is created the data frame **'excl_coords'** with the erroneous and/or missing coordinates.

Examples

```
## Not run:
test_geocoord(coords = eraclim_uao_fp)

## End(Not run)
```

Index

∗Topic **datasets**
eraclim_uao_fp, 4
ispd, 9

compare_country, 2, *11*
countries_polys, 3, *11*

eraclim_uao_fp, 4

get_country, *2, 3*, 4, *5–12*
get_country_shoreline, *2, 3*, *5*, 5, *8–11*
get_lon180, *2*, *5*, 7, *9–11*
get_sea, *2*, *5, 6*, 8, *9–11*

ispd, 9

order_data, 9, *11*

seas_polys, *11*, 11
stlocationqc, 11
stlocationqc-package *(*stlocationqc*)*, 11

test_geocoord, *2*, *5*, *9–11*, 12