Package: EEAaq (via r-universe)

March 9, 2025

Title Handle Air Quality Data from the European Environment Agency Data Portal

Version 1.0.0

Description This software downloads and manages air quality data from the European Environmental Agency (EEA) dataflow (<https://www.eea.europa.eu/data-and-maps/data/aqereporting-9>). See the web page <https://eeadmz1-downloads-webapp.azurewebsites.net/> for details on the EEA's Air Quality Download Service. The package allows dynamically mapping the stations, summarising and time aggregating the measurements and building spatial interpolation maps. See the web page <https://www.eea.europa.eu/en> for further information on EEA activities and history. Further details, as well as, an extended vignette of the main functions included in the package, are available at the GitHub web page dedicated to the project.

URL https://github.com/PaoloMaranzano/EEAaq_R

License GPL (>= 3)

Depends R (>= 2.10)

Imports arrow, curl, dplyr, ggplot2, ggpubr, gifski, grDevices, gstat, htmlwidgets, httr, leaflet, lubridate, raster, readr, sf, stats, tibble, tidyr, utils

Encoding UTF-8

RoxygenNote 7.3.2

Suggests knitr, rmarkdown, rvest, readxl, digest, gh, base64enc, stringr

NeedsCompilation no

Author Paolo Maranzano [aut, cre, cph] (<https://orcid.org/0000-0002-9228-2759>), Riccardo Borgoni [aut, cph] (<https://orcid.org/0000-0002-2520-3512>), Samir Doghmi [aut, cph], Agostino Tassan Mazzocco [aut, cph]

Maintainer Paolo Maranzano <pmaranzano.ricercastatistica@gmail.com>

Date/Publication 2025-02-06 19:30:06 UTC

Config/pak/sysreqs cmake libgdal-dev gdal-bin libgeos-dev make libicu-dev libpng-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev libx11-dev

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

RemoteUrl https://github.com/cran/EEAaq

RemoteRef HEAD

RemoteSha e7245aec7ee024146c4df8a0bb6949d5434b6d58

Contents

`code_extr` *Code_extr*

Description

This function extracts the numerical value from NUTS-level strings.

Usage

```
code_extr(level)
```

Arguments

`level`	A character vector representing NUTS-level codes (e.g., `c("NUTS2", "NUTS3")`).

Value

A sorted numeric vector containing the extracted NUTS levels.

EEAaq_export *Export and save an* EEAaq_df *class object*

Description

EEAaq_export saves an EEAaq_df class object as a *.csv* or a *.txt* file, and exports the associated shapefile as well.

Usage

```
EEAaq_export(data, filepath, format, shape = FALSE)
```

Arguments

data	an EEAaq_df class object.
filepath	character string giving the file path
format	character string giving the format of the file. It must be one of 'csv' and 'txt'.
shape	logical value (T or F). If TRUE the shapefile associated to the EEAaq_df object given in input is saved in the same directory specified in filepath. If FALSE (the default), only the data frame containing the data is saved.

Value

No return value, called for side effects.

Examples

```
#Download a dataset with the function EEAaq_get_data, which generate an EEAaq_df object.
data <- EEAaq_get_data(zone_name = "15146", NUTS_level = "LAU",
                       LAU_ISO = "IT", pollutants = "PM10",
                       from = "2023-01-01", to = "2023-05-31")
temp <- tempdir()
filepath <- paste0(temp, "/data.csv")
EEAaq_export(data = data, filepath = filepath, format = "csv", shape = TRUE)
```

EEAaq_get_data *Download air quality data at european level from the EEA download service*

Description

This function imports air quality datasets at european level, based on the zone, time and pollutant specifications. This function generates an `EEAaq_df` object, or an `EEAaq_df_sfc`.

Usage

```
EEAaq_get_data(
  zone_name = NULL,
  NUTS_level = NULL,
  LAU_ISO = NULL,
  pollutants = NULL,
  from = NULL,
  to = NULL,
  quadrant = NULL,
  polygon = NULL,
  verbose = TRUE
)
```

Arguments

`zone_name`	character vector specifying the names of the zones to consider. The reference is the NUTS and LAU nomenclature by Eurostat. See *Details*.
`NUTS_level`	character that specify the level of NUTS or LAU, to which `zone_name` belongs. Allowed values are 'NUTS0', 'NUTS1', 'NUTS2', 'NUTS3', 'LAU'. For further information see *Details*.
`LAU_ISO`	a code to identify the corresponding ISO of the country since LAU_ID are not unique over Europe
`pollutants`	the pollutants for which to download data. It may be: • character vector representing the short names of the pollutants to analyse. The reference is the variable `Notation` in the dataset `pollutants` provided by this package. • numeric vector representing the codes of the pollutants to analyse. The reference is the variable `Code` in the dataset `pollutants` provided by this package.
`from`	the starting point of the time window to consider. It may be: • character containing a specific day of the year in the format `yyyy-mm-dd`
`to`	the ending point of the time window to consider. It may be: • character containing a specific day of the year in the format `yyyy-mm-dd`

	ID logic value (T or F). If TRUE (the default), the character specified in the parameter zone_name is the unique identifier code provided by Eurostat. The reference is the NUTS_ID column from the NUTS dataset or the LAU_ID from the LAU dataset. Also, in the case of using LATN_NAME from the NUTS dataset or LAU_NAME from the LAU dataset, TRUE must be specified. If FALSE , it is used when polygon or quadrant are not null.
quadrant	a list of bidimensional numeric vectors containing the coordinates in **WGS84** format. If the list has two elements, the function builds a square using the two coordinates as opposite extremes. If the list contains three or more elements, every point is a vertex of a polygon, in particular the convex hull of the specified points.
polygon	A sfc_POLYGON or sfc_MULTIPOLYGON class object The polygon can be imported via shapefile or other formats from the user.
verbose	logic value (T or F). If TRUE (the default) information about the function progress are printed. If FALSE no message is printed.

Details

Some specific notes:

- If the parameter zone_name corresponds to a valid CITY_NAME (i.e., not NULL in the dataset), the function will return the corresponding data. If no valid CITY_NAME is associated with the zone_name, the function attempts to retrieve all available data for the entire country and subsequently filter for the specified zone_name.
- For very small towns or certain countries, such as Turkey or Albania, data may not currently be available in the dataset. This limitation reflects the data unavailability at the the EEA Air Quality Viewer https://discomap.eea.europa.eu/App/AQViewer/index.html?fqn=Airquality_Dissem.b2g.AirQualityStatistics.
- If the parameters used in the query include polygon or quadrant, the function returns a EEAaq_df_sfc object. Otherwise, it returns an EEAaq_df object, which is a tibble dataframe.

The NUTS classification (Nomenclature of territorial units for statistics) is a hierarchical system for dividing up the economic territory of the EU and the UK. The levels are defined as in https://ec.europa.eu/eurostat/web/gisco/geodata/statistical-units/territorial-units-statistics, that is,

- **NUTS 0**: the whole country
- **NUTS 1**: major socio-economic regions
- **NUTS 2**: basic regions for the application of regional policies
- **NUTS 3**: small regions for specific diagnoses

Value

A data frame of class EEAaq_df, if zone_name is specified, and of class EEAaq_df_sfc if whether the parameter quadrant or polygon is specified.

Examples

```
# Download hourly NO2 concentration for Milan city (LAU = 15146) in 2023
data <- EEAaq_get_data(zone_name = "15146", NUTS_level = "LAU",LAU_ISO = "IT",
pollutants = c("NO2","CO"), from = "2023-01-01", to = "2023-12-31",  verbose = TRUE)
```

`EEAaq_get_dataframe` *EEAaq_get_dataframe*

Description

Retrieve one of the metadata (i.e., LAU, NUTS, stations, or pollutant) tables from the EEA dataflow. This function downloads and loads one dataset at a time from a predefined list of available datasets. Ensure that the dataset name is written correctly. See details for further details.

Usage

```
EEAaq_get_dataframe(dataframe = NULL)
```

Arguments

`dataframe`	name of the `data.frame` to retrieve. Select among: • 'LAU': `data.frame` containing metadata information on all the local administrative units (i.e., municipalities) in Europe according to the NUTS nomenclature by Eurostat. Information includes geometries. • 'NUTS: `data.frame` containing metadata information on all the major socio-economic regions in Europe according to the NUTS nomenclature by Eurostat. Information includes geometries. • 'stations': `data.frame` containing metadata information on all the monitoring stations maintained (both currencly active and de-activated) by the EEA and available in `EEAaq`. Information include: unique identifiers, extended descriptions, and technical details on operations and data collected. • 'pollutant': `data.frame` containing metadata information on all the available pollutants monitored by the EEA and available in `EEAaq`. Information include: unique identifiers, extended descriptions, and unit of measure.

Details

The function retrieves from the EEAaq GitHub folder one of the following metadata: since 2024, the data EEA air quality retrieving dataflow is undergoing a major re-organization. In particular, since January 2025, raw data are accessible only through an online platform/dashboard. While `EEAaq` is build to explicitly deal with the automatic and constantly-updated system for raw data, the same process is not always possible for the metadata. Indeed, most of the metadata information are updated and require relevant pre-processing (i.e., data manipulation and cleaning) steps to make them consistent with the main database on pollutants concentrations. For this reasons, all the metadata files are periodically pre-processed and updated by the package maintainers. For issues with the data or code, please contact the development team at pmaranzano.ricercastatistica@gmail.com

Value

a dataframe

Examples

```
LAU <- EEAaq_get_dataframe(dataframe= "LAU")
```

`EEAaq_get_stations` *Download EEA measurement station information dataset*

Description

Download the updated dataset from EEA, containing measurement station information. For further information about the variables see `stations`.

Usage

```
EEAaq_get_stations(byStation = FALSE, complete = TRUE)
```

Arguments

`byStation`	Logic value (T or F). If `TRUE` the dataset is organized by station (one row for each measurement station). If `FALSE` the dataset is organized by sampling point. Each station have multiple sampling points.
`complete`	Logic value (T or F). If `TRUE`, the dataset contains all the variables given by the EEA. If `FALSE` the dataset contains only a few variables, the most importants. For further details about the variables, see `stations`.

Details

Note that, for very small towns or certain countries, such as Turkey or Albania, data may not currently be available in the dataset. This limitation reflects the data unavailability at the the EEA Air Quality Viewer https://discomap.eea.europa.eu/App/AQViewer/index.html?fqn=Airquality_Dissem.b2g.AirQualityStatistics.

Value

A tibble containing the stations information. Further details available here `stations`.

Examples

```
EEAaq_get_stations(byStation = FALSE, complete = TRUE)
```

EEAaq_idw_map *Build a spatial interpolation map based on the Inverse Distance Weighting technique.*

Description

EEAaq_idw_map recieves as input a EEAaq_taggr_df or a EEAaq_taggr_df_sfc class object and produces a spatial interpolation map. Depending on the time frequency of the aggregation, multiple maps are generated, one for each timestamp. It may be exported as pdf, jpeg, png, gif and html.

Usage

```
EEAaq_idw_map(
  data = NULL,
  pollutant = NULL,
  aggr_fun,
  bounds_level = NULL,
  distinct = FALSE,
  gradient = TRUE,
  idp = 2,
  nmax = NULL,
  maxdist = NULL,
  dynamic = FALSE,
  fill_NUTS_level = NULL,
  tile = "Esri.WorldGrayCanvas",
  save = NULL,
  filepath = NULL,
  width = 1280,
  height = 720,
  res = 144,
  delay = 1,
  verbose = TRUE
)
```

Arguments

data	an object of class EEAaq_taggr_df or EEAaq_taggr_df_sfc, which is the output of the EEAaq_time_aggregate function.
pollutant	vector containing the pollutant for which to build the map. It must be one of the pollutants contained in data.
aggr_fun	charachter containing the aggregation function to use for computing the interpolation. It must be one of the statistics contained in data.
bounds_level	character containing the NUTS level or LAU for which draw internal boundaries. Admissible values are 'NUTS0', 'NUTS1', 'NUTS2', 'NUTS3', 'LAU'.

distinct	logic value (T or F). If TRUE, each map generated is printed and saved in distinct pages (for instance if data has a monthly frequency in a yearly time window, 12 distinct plots are generated). If FALSE (the default), the maps are printed in a single page.
gradient	logic value (T or F). If TRUE (the default) the maps generated are colored with a continuous color scale. If FALSE, the color scale is discrete.
idp	numeric value that specify the inverse distance weighting power. For further information see idw.
nmax	numeric value; specify the number of nearest observations that should be used for the inverse distance weighting computing, where nearest is defined in terms of the space of the spatial locations. By default, all observations are used. For further information see idw
maxdist	numeric value; only observations within a distance of maxdist from the prediction location are used for the idw computation. By default, all observations are used. If combined with nmax, both criteria apply.
dynamic	logic value (T or F). If TRUE the function creates a Leaflet map widget using **htmlwidgets** (for further information see leaflet). If FALSE (the default) the maps generated are static.
fill_NUTS_level	character containing the NUTS level or LAU for which to aggregate the idw computing, in order to obtain a uniform coloring inside each area at the specified level. (For instance if fill_NUTS_level = "LAU", each municipality is filled by the mean value of the pollutant concentration, computed by the idw, of each pixel inside the respective municipality). Allowed values are 'NUTS0', 'NUTS1', 'NUTS2', 'NUTS3', 'LAU'.
tile	character representing the name of the provider tile. To see the full list of the providers, run providers. For further information see addProviderTiles.
save	character representing in which extension to save the map. Allowed values are 'jpeg', 'png', 'pdf' (if dynamic = FALSE), 'gif' (if dynamic = FALSE & distinct = TRUE), 'html' (if dynamic = TRUE).
filepath	a character string giving the file path.
width, height	the width and the height of the plot, expressed in pixels (by default width = 1280, height = 720). This parameters are available only for save 'jpeg', 'png' and 'gif'. For further information see png or jpeg.
res	the nominal resolution in ppi which will be recorded in the bitmap file, if a positive integer (by default res = 144). This parameter is available only for save 'jpeg', 'png'. For further information see png or jpeg.
delay	numeric value specifying the time to show each image in seconds, when save = "gif".
verbose	logic value (T or F). If TRUE (the default) information about the function progress are printed. If FALSE no message is printed.

Details

EEAaq_idw_map create a spatial interpolation map, based on the Inverse Distance Weighting method (Shepard 1968). This method starts from the available georeferenced data and estimates the value

of the variable in the points where it's unknown as a weighted average of the known values, where weights are given by an inverse function of the distance of every point from the fixed stations. The greater the distance of a point from a station, the smaller the weight assigned to the values of the respective station for the computing of that unknown point. Given the sampling plan s_i for $i = 1, ..., n$, which represent the location of the air quality stations, the pollutant concentration value $Y(s_i) = Y_i$ represents the value of the pollutant concentration detected by the site s_i and u is the point for which the value of the concentration in unknown.

$$\hat{Y}(u) = \sum_{i=1}^{n} Y_i \omega_i(u),$$

where

$$\omega_i(u) = \frac{g(d(s_i, u))}{\sum_{i=1}^{n} g(d(s_i, u))}$$

represent the weights assigned to each location s_i and $d(s_i, u)$ is the distance between u and s_i.

Value

cosa restituisce la funzione

Examples

```
## Not run:
# Download daily NO2 data in 2023 for Milan city (LAU)
data <- EEAaq_get_data(zone_name = "15146", NUTS_level = "LAU",LAU_ISO = "IT",
pollutants = "NO2", from = "2023-01-01", to = "2023-12-31",  verbose = TRUE)
# Monthly aggregation
t_aggr <- EEAaq_time_aggregate(data = data, frequency = "monthly",
aggr_fun = c("mean", "min", "max"))

# One map created
EEAaq_idw_map(data = t_aggr, pollutant = "NO2", aggr_fun = "mean",
distinct = TRUE, gradient = FALSE, dynamic = TRUE, fill_NUTS_level = "LAU")


# Let's try to change the parameters fill_NUTS_level and dynamic:
# now we are going to use a dataset containing PM10 concentrations
# in Milan province (NUTS 3), during 2023
data <- EEAaq_get_data(zone_name = "Centro (IT)", NUTS_level = "NUTS1",
pollutant = "PM10", from = "2023-01-01", to = "2023-12-31")
# Yearly aggregation
t_aggr <- EEAaq_time_aggregate(data = data, frequency = "yearly", aggr_fun = "mean")

# Let us generate one dynamic map, containing the municipalities inside the Milan province
# filled with the mean concentration value for 2023, computed via IDW:
EEAaq_idw_map(data = t_aggr, pollutant = "PM10", aggr_fun = "mean",
distinct = TRUE, gradient = FALSE, dynamic = TRUE, fill_NUTS_level = "NUTS3")

## End(Not run)
```

EEAaq_import *Reverse function of* EEAaq_export. *Reads an* EEAaq_df *object*

Description

Given the file containing the data saved with EEAaq_export, and the associated shapefile, EEAaq_read imports the EEAaq_df class object.

Usage

```
EEAaq_import(file_data, file_shape)
```

Arguments

file_data	file path of the 'csv' or 'txt' file containing the air quality data to import.
file_shape	file path of the shapefile associated to the dataset used in file_data

Value

No return value, called for side effects.

Examples

```
#Download a dataset with the function EEAaq_get_data, which generate an EEAaq_df object.
data <- EEAaq_get_data(zone_name = "15146", NUTS_level = "LAU",LAU_ISO = "IT",
pollutants = "PM10", from = "2023-01-01", to = "2023-05-31",  verbose = TRUE)

temp <- tempdir()
filepath <- paste0(temp, "/data.csv")
#Export the dataset and the associated shape
EEAaq_export(data = data, filepath = filepath, format = "csv", shape = TRUE)
#Import the EEAaq_df object saved in the previous code line
EEAaq_import(file_data = filepath, file_shape = paste0(temp, "/data.shp"))
```

EEAaq_map_stations *Create a map representing the stations based on the data given in input or the information specified in the parameters*

Description

Build static or dynamic maps, representing the location of the stations that detects the specified pollutants. It recieves in input an EEAaq_df or an EEAaq_df_sfc class object, or, alternatively, it's possible to specify the required zones and pollutants with the same nomenclature system of the EEAaq_get_data function.

Usage

```
EEAaq_map_stations(
  data = NULL,
  pollutant = NULL,
  zone_name = NULL,
  NUTS_level = NULL,
  ID = FALSE,
  bounds_level = NULL,
  color = TRUE,
  dynamic = FALSE
)
```

Arguments

data	an EEAaq_df or EEAaq_df_sfc class object, which is the output of the EEAaq_get_data function.
pollutant	character vector containing the short names of the pollutants for which locate the stations.
zone_name	character vector specifying the names of the zones to consider. The reference is the NUTS and LAU nomnclature by Eurostat.
NUTS_level	chracter that specify the level of NUTS or LAU, to which the zone_name belongs.
ID	logic value (T or F). If TRUE the character specified in the parameter zone_name is the unique identifier code provided by Eurostat. If FALSE (the default) it indicates that was specified the full name in latin characters.
bounds_level	character containing the NUTS level or LAU for which draw internal boundaries. Admissible values are "NUTS0", "NUTS1", "NUTS2", "NUTS3", "LAU" and it must be of a lower level then the one specified in the parameter NUTS_level.
color	logical value (T or F). If TRUE (the default) the points are colored based on the pollutant they are able to detect. If FALSE the points have the same color.
dynamic	logical value (T or F). If TRUE the map is interactive and dynamic. If FALSE (the default) the map is static.

Value

A map representing the specified area and the points representing the location of the stations able to detect the specified pollutants.

Examples

```
# Using as example PM data in Lombardia (Italy) during the whole 2022,
# it's possible to map the stations in two ways.
# First of all specifying the zone information:
EEAaq_map_stations(pollutant = c("PM10", "PM2.5"),
zone_name = "Lombardia",NUTS_level = "NUTS2", ID = FALSE, color = FALSE)
#In this case each point have the same color.
```

```
# Alternatively, it is possible to use the data already downloaded in the parameter data,
# coloring the points based on the pollutants the respective station detects.
data <- EEAaq_get_data(zone_name = "15146", NUTS_level = "LAU",LAU_ISO = "IT",
pollutants = "PM10", from = "2023-01-01", to = "2023-05-31",  verbose = TRUE)
EEAaq_map_stations(data = data, color = TRUE)
```

EEAaq_summary *Generate an* EEAaq_df *data summary*

Description

This function, applied to an EEAaq_df or EEAaq_df_sfc class object, produces a list of data frames, containing relevant information about the data, such as descriptive statistics, missing values statistics, gap length and correlation.

Usage

```
EEAaq_summary(data = NULL, verbose = TRUE)
```

Arguments

data	an EEAaq_df or EEAaq_df_sfc class object, which is the output of the EEAaq_get_data function.
verbose	logic value (T or F). If TRUE (the default) messages about the function progress are printed. If FALSE no message is printed.

Value

The function EEAaq_summary computes and return a list of summary statistics of the dataset given in data. In particular the elements of the list are:

- Summary global missing count, missing rate, negative count, minimum, maximum, mean and standard deviation, organized by pollutant.
- Summary_byStat list of data frames, one for each different station, containing the descriptive statistics (missing count, missing rate, negative count, minimum, maximum, mean and standard deviation), organized by station.
- gap_length one data frame for each pollutant, containing the gap length organized by station.
- Corr_Matrix if data contains more than one pollutant, the correlation matrix between pollutans is provided, organised by station.

Examples

```
data <- EEAaq_get_data(zone_name = "15146", NUTS_level = "LAU",LAU_ISO = "IT",
pollutants = "PM10", from = "2023-01-01", to = "2023-05-31",  verbose = TRUE)

EEAaq_summary(data)
```

EEAaq_time_aggregate *Time aggregation of an* EEAaq_df *class object.*

Description

EEAaq_time_aggregate compute a time aggregation of an EEAaq_df or EEAaq_df_sfc class object, based on the specified frequency and the aggregation functions aggr_fun.

Usage

```
EEAaq_time_aggregate(
  data = NULL,
  frequency = "monthly",
  aggr_fun = c("mean", "min", "max")
)
```

Arguments

data	an EEAaq_df or EEAaq_df_sfc class object, which is the output of the EEAaq_get_data function.
frequency	vector containing the time frequency for which to aggregate the data object. Admissible values are 'yearly', 'monthly', 'weekly', 'daily' 'hourly'.
aggr_fun	character vector containing one or more agregation functions. Admissible values are 'mean', 'median', 'min', 'max', 'sd', 'var', 'quantile_pp' (where pp is a number in the range [0,1], representing the required percentile).

Value

A EEAaq_taggr_df or a EEAaq_taggr_df_sfc class object, which is a tibble containing the required time aggregation.

Examples

```
data <- EEAaq_get_data(zone_name = "15146", NUTS_level = "LAU",LAU_ISO = "IT", pollutants = "PM10",
from = "2023-01-01", to = "2023-12-31",  verbose = TRUE)
EEAaq_time_aggregate(data = data, frequency = "monthly", aggr_fun = c("mean", "min", "max"))
EEAaq_time_aggregate(data = data, frequency = "yearly", aggr_fun = "mean")
```

get_LAU *Get LAU data*

Description

Local Administrative Units (LAUs) are the building blocks of the NUTS classification and correspond to the municipalities and communes within the EU To get the final dataframe we combine two dataset: one taken from Eurostat (https://ec.europa.eu/eurostat/web/nuts/local-administrative-units)that includes City names and City IDs, essential for querying and associations. The other one taken from EEA which provides LAU information. The Latter dataset is updated automatically by selecting the most recent shapefile (SHP) available online. While The Eurostat dataset URL needs to be manually updated with the latest download link to ensure the City-related data is current.

Usage

```
get_LAU(year = "Null")
```

Arguments

year	expressed as four digit (YYYY)

Value

A tibble containing LAUs information with selected columns (e.g., ISO, LAU_ID, NUTS3_ID and geometry).

get_NUTS *Get NUTS*

Description

It automatically updates the dataset by identifying the most recent available file, accessing the corresponding page, and downloading the SHP file at the 1:20 Million scale with the EPSG:4326 reference system from this website (https://gisco-services.ec.europa.eu/distribution/v2/nuts/)

Usage

```
get_NUTS(year = "Null")
```

Arguments

year	expressed as four digit (YYYY)

Value

A tibble containing LAUs information with selected columns (NUTS_ID, LEVL_CODE...)

get_pollutants *Get pollutant*

Description

Retrieve Pollutant Data from EEA Vocabulary (https://dd.eionet.europa.eu/vocabulary/aq/pollutant) Downloads and processes pollutant data from the EEA (European Environment Agency) vocabulary database. The data includes relevant information such as pollutant names, codes, and descriptions.

Usage

```
get_pollutants()
```

Value

A tibble containing pollutant information with selected columns (e.g., URI, notation, and extracted code).

get_stations *Get Station Data*

Description

This function downloads detailed information for each SamplingPointId. It performs a spatial join to merge the spatial information of LAU and NUTS (specifically, the geometries of LAU and the geometry of stations) and fills in the missing data for CITY_NAME and CITY_ID (retrieved from https://discomap.eea.europa.eu/App/AQViewer/index.html?fqn=Airquality_Dissem.b2g.AirQualityStatistics) through a left join based on the AirQualityStationEoICode column. These values are essential for querying the endpoint. The missing_cities file was obtained manually (from 2000 to 2024) because the website did not allow downloading more than 100,000 rows at a time. The data was collected in multiple batches, filtering SamplingPoints using the following criteria:

- Filter on data used in AQ Report: yes
- Filter on data coverage: yes For each station, the column AirQualityStationEoICode (identical for all sensors at the same station) was used to select the first row containing unique values for CITY_NAME and CITY_ID. No station reported more than one value for this pair of columns. To support future uploads, it is necessary to integrate updated AirQualityStationEoICode values.

Usage

```
get_stations()
```

Value

a tibble

handle_dates *Handle Dates based on Dataset Ranges*

Description

This function handles dates based on the respective dataset. According to the documentation:

- Data from 2024 onwards corresponds to Unverified data transmitted continuously (Up-To-Date/UTD/E2a).
- Data from 2013 to the begin of 2023 corresponds to Verified data (E1a) reported by countries by 30 September each year for the previous year.
- Data delivered before 2012 corresponds to Historical Airbase data. The range for E1 is extended until 31/12/2023 because the observations are already validated, and no data for 2023 is retrieved when considering E2.

Usage

```
handle_dates(from, to)
```

Arguments

from	StartDate (in "YYYY-MM-DD" format).
to	EndDate (in "YYYY-MM-DD" format).

Value

A list of datasets with associated date ranges and descriptions.

is_EEAaq_df *Check if a given object is an* EEAaq_df *class object*

Description

Given an object as input, is_EEAaq_df verify that the given object belongs to the EEAaq_df class.

Usage

```
is_EEAaq_df(data)
```

Arguments

data	the object for which verify the if it belongs to the EEAaq_df class.

Value

logical value (T ot F). If TRUE the object given in input is an EEAaq_df object. If FALSE the object doesn't belong to the EEAaq_df class.

Examples

```
#Download a dataset with the function EEAaq_get_data, which generate an EEAaq_df object.
data <- EEAaq_get_data(zone_name = "15146", NUTS_level = "LAU",LAU_ISO = "IT",
pollutants = "PM10", from = "2023-01-01", to = "2024-08-29",  verbose = TRUE)

#Check if the imported object belongs to the EEAaq_df class
is_EEAaq_df(data = data)
```

---

my_summarise *Aggregate data based on a specific statistic*

Description

Given data and the aggregation function desired, this function compute a time aggregation of the data.

Usage

```
my_summarise(data, fun_aggr)
```

Arguments

data	An EEAaq_df or EEAaq_df_sfc class object, which is the output of the EEAaq_get_data function.
fun_aggr	Vector character containing the aggregation function for which to time aggregate.

Value

A tibble with the required aggregation.

Index