Package ‘EEAaq’

February 10, 2026

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

Version 1.0.3

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 (>= 4.0)

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

Encoding UTF-8

RoxygenNote 7.3.3

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

VignetteBuilder knitr

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2026-02-10 11:50:07 UTC

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` export an `EEAaq_df` class object as a *.csv* or a *.txt* file.

Usage

```
EEAaq_export(data, filepath, format)
```

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'.

Value

No return value, called for side effects.

Examples

```
### Download PM10 data for the province (NUTS-3) of Milano
## (Italy) from January 1st to January 31st, 2023
`%>%` <- dplyr::`%>%`
IDstations <- EEAaq_get_stations(byStation = TRUE, complete = FALSE)
IDstations <- IDstations %>%
                dplyr::filter(NUTS3 %in% c("Milano")) %>%
                dplyr::pull(AirQualityStationEoICode) %>%
                unique()
data <- EEAaq_get_data(IDstations = IDstations, pollutants = "PM10",
                       from = "2024-01-01", to = "2025-01-31", verbose = TRUE)

### Export data to csv file
temp <- tempdir()
filepath <- paste0(temp, "/data.csv")
EEAaq_export(data = data, filepath = filepath, format = "csv")
```

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

Description

This function retrieves air quality datasets at european level, based on station, time and pollutant specifications. This function generates a `data.frame/tibble` object of class `EEAaq_df`.

Usage

```
EEAaq_get_data(
  IDstations = NULL,
  pollutants = NULL,
  from = NULL,
  to = NULL,
  verbose = TRUE
)
```

Arguments

`IDstations` Numeric value. Unique ID of the stations to retrieve.

`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` character defining the initial date of the period to be retrieved. The format is `yyyy-mm-dd`.

`to` character defining the final date of the period to be retrieved. The format is `yyyy-mm-dd`.

`verbose` logic value (T or F). If `TRUE` (the default) information about the function progress are printed. If `FALSE` no message is printed.

Details

Recall that stations and sensors are physically managed by national or local environmental protection agencies with their own specificities and rules. EEA operates as a collector of national environmental protection systems and harmonizes the information received by national offices. However, data provided can change on a country basis. For instance, time resolution, sampling frequency, spatial coverage, or the classifications (e.g., urban or rural) can differ country by country. Before downloading the data, we suggest to manage and filter the stations/sensors of interest through their metadata files (provided by `EEAaq_get_stations` or `EEAaq_get_dataframe`). See the examples and the vignette for practical examples.

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

```
`%>%` <- dplyr::`%>%`
### Download PM10 data for the province (NUTS-3) of Milano (Italy)
## from January 1st to January 31st, 2023
IDstations <- EEAaq_get_stations(byStation = TRUE, complete = FALSE)
IDstations <- IDstations %>%
                dplyr::filter(NUTS3 %in% c("Milano")) %>%
                dplyr::pull(AirQualityStationEoICode) %>%
                unique()
data <- EEAaq_get_data(IDstations = IDstations, pollutants = "PM10",
                       from = "2024-01-01", to = "2025-01-31", 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 and Eurostat dataflows. 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 information from the `EEAaq` GitHub folder one of the available metadata. Since the end of 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 (on GitHub) 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")
pollutant <- EEAaq_get_dataframe(dataframe = "pollutant")
stations <- EEAaq_get_dataframe(dataframe = "stations")
NUTS <- EEAaq_get_dataframe(dataframe = "NUTS")
```

`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 = TRUE, 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 = TRUE, complete = TRUE)
```

`EEAaq_idw_map` *Build a spatial interpolation map based on the Inverse Distance Weighting technique. The function* `EEAaq_idw_map` *requires 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. Interpolation maps may be exported as pdf, jpeg, png, gif and html.*

Description

Build a spatial interpolation map based on the Inverse Distance Weighting technique. The function `EEAaq_idw_map` requires 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. Interpolation maps may be exported as pdf, jpeg, png, gif and html.

Usage

```
EEAaq_idw_map(
  data = NULL,
  pollutant = NULL,
  aggr_fun,
  distinct = FALSE,
  gradient = TRUE,
  idp = 2,
  nmax = NULL,
  maxdist = NULL,
  NUTS_filler = NULL,
  NUTS_extborder = NULL,
  NUTS_intborder = NULL,
  dynamic = FALSE,
  tile = "Esri.WorldGrayCanvas",
  filepath = NULL,
  width = 1280,
  height = 720,
  res = 144,
  delay = 1,
  save = NULL,
  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`	character containing the aggregation function to use for computing the interpolation. It must be one of the statistics contained in `data`.
`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.
`NUTS_filler`	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. Recall that 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 follows: • **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 • **LAU**: municipality For instance if `NUTS_filler = "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', and 'LAU'.

`NUTS_extborder`	character containing the NUTS level or LAU for which draw external boundaries. Admissible values are 'NUTS0', 'NUTS1', 'NUTS2', 'NUTS3', 'LAU'.
`NUTS_intborder`	character containing the NUTS level or LAU for which draw internal boundaries. Admissible values are 'NUTS0', 'NUTS1', 'NUTS2', 'NUTS3', 'LAU'.
`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.
`tile`	character representing the name of the provider tile. To see the full list of the providers, run `providers`. For further information see `addProviderTiles`.
`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"`.
`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`).
`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:
### Filter all the stations installed in the city (LAU) of Milano (Italy)
IDstations <- EEAaq_get_stations(byStation = FALSE, complete = FALSE)
 `%>%` <- dplyr::`%>%`
IDstations <- IDstations %>%
                dplyr::filter(LAU_NAME == "Milano") %>%
                dplyr::pull(AirQualityStationEoICode) %>%
                unique()
### Download NO2 measurement for the city of Milano from January 1st to December 31st, 2023
data <- EEAaq_get_data(IDstations = IDstations, pollutants = "NO2",
                       from = "2023-01-01", to = "2023-01-31", verbose = TRUE)

### Monthly aggregation: compute station-specific monthly minimum,
## average, and maximum NO2 concentrations
t_aggr <- EEAaq_time_aggregate(data = data, frequency = "monthly",
                               aggr_fun = c("mean", "min", "max"))

### Static IDW interpolation of the average NO2 concentrations for the whole Lombardy
###   region (NUTS_extborder = "NUTS2"). Interpolated values are then aggregated at the provincial
###    level (NUTS_filler = "NUTS3")
EEAaq_idw_map(data = t_aggr, pollutant = "NO2", aggr_fun = "mean",
              distinct = TRUE, gradient = FALSE,
              dynamic = FALSE,
              NUTS_filler = "NUTS3", NUTS_extborder = "NUTS2")

### Dynamic IDW interpolation map (interactive leafleat) of the average
## NO2 concentrations for the whole Lombardy
###   region (NUTS_extborder = "NUTS2"). Interpolated values are then aggregated at the municipal
###    level (NUTS_filler = "LAU")
EEAaq_idw_map(data = t_aggr, pollutant = "NO2", aggr_fun = "mean",
              distinct = TRUE, gradient = FALSE,
              dynamic = TRUE,
              NUTS_filler = "LAU", NUTS_extborder = "NUTS2", NUTS_intborder = "LAU")

## End(Not run)
```

`EEAaq_import` *Reverse function of* `EEAaq_export`*. Reads an* `EEAaq_df` *object from a .txt or .csv file saved through* `EEAaq_export`.

Description

Reverse function of `EEAaq_export`. Reads an `EEAaq_df` object from a .txt or .csv file saved through `EEAaq_export`.

Usage

```
EEAaq_import(file_data)
```

Arguments

`file_data`	file path of the 'csv' or 'txt' file containing the air quality data to import.

Value

No return value, called for side effects.

Examples

```
`%>%` <- dplyr::`%>%`
### Download PM10 data for the province (NUTS-3) of Milano (Italy)
## from January 1st to January 31st, 2023
IDstations <- EEAaq_get_stations(byStation = TRUE, complete = FALSE)
IDstations <- IDstations %>%
                dplyr::filter(NUTS3 %in% c("Milano")) %>%
                dplyr::pull(AirQualityStationEoICode) %>%
                unique()
data <- EEAaq_get_data(IDstations = IDstations, pollutants = "PM10",
                       from = "2023-01-01", to = "2023-01-31", verbose = TRUE)

### Export data to csv file
temp <- tempdir()
filepath <- paste0(temp, "/data.csv")
EEAaq_export(data = data, filepath = filepath, format = "csv")

### Import the EEAaq_df object saved in the previous code line
EEAaq_import(file_data = filepath)
```

`EEAaq_map_stations` *Create a static or dynamic (interactive leaflet) map representing the geographical locations of the stations based on a user-defined input dataset of class* `EEAaq_df` *or* `EEAaq_df_sfc`.

Description

Create a static or dynamic (interactive leaflet) map representing the geographical locations of the stations based on a user-defined input dataset of class `EEAaq_df` or `EEAaq_df_sfc`.

Usage

```
EEAaq_map_stations(
  data = NULL,
  NUTS_extborder = NULL,
  NUTS_intborder = 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.
`NUTS_extborder`	character containing the NUTS level or LAU for which draw external boundaries. Admissible values are 'NUTS0', 'NUTS1', 'NUTS2', 'NUTS3', 'LAU'. Recall that 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 follows: • **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 • **LAU**: municipality
`NUTS_intborder`	character containing the NUTS level or LAU for which draw internal boundaries. Admissible values are 'NUTS0', 'NUTS1', 'NUTS2', 'NUTS3', 'LAU'.
`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

```
`%>%` <- dplyr::`%>%`
### Retrieve all the stations measuring PM10 in Belgium
IDstations <- EEAaq_get_stations(byStation = FALSE, complete = FALSE)
IDstations <- IDstations %>%
  dplyr::filter(ISO %in% c("BE"),
                AirPollutant %in% "PM10") %>%
  dplyr::pull(AirQualityStationEoICode) %>%
  unique()

### Download the corresponding data froom December 1st to December 31st, 2021
data <- EEAaq_get_data(IDstations = IDstations, pollutants = "PM10",
                       from = "2021-12-01", to = "2021-12-31", verbose = TRUE)

### Static map of available stations across the whole country. External borders are given by the
###     union of the available regions (NUTS-2), while municipalities
## (LAUs) are used as inner borders.
EEAaq_map_stations(data = data,
                  NUTS_extborder = "NUTS2", NUTS_intborder = "LAU",
                  color = TRUE, dynamic = FALSE)
### Dynamic (interactive leaflet) map of available stations across the whole country.
## External borders are given by the
```

```
###   union of the available regions (NUTS-2), while provinces (NUTS-3) are used as inner borders.
EEAaq_map_stations(data = data,
                   NUTS_extborder = "NUTS2", NUTS_intborder = "NUTS3",
                   color = TRUE, dynamic = 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

```
### Download PM10 data for the province (NUTS-3) of Milano (Italy)
##from January 1st to January 31st, 2023
IDstations <- EEAaq_get_stations(byStation = TRUE, complete = FALSE)
`%>%` <- dplyr::`%>%`
IDstations <- IDstations %>%
                dplyr::filter(NUTS3 %in% c("Milano")) %>%
                dplyr::pull(AirQualityStationEoICode) %>%
                unique()
data <- EEAaq_get_data(IDstations = IDstations, pollutants = "PM10",
                       from = "2023-01-01", to = "2023-01-31", verbose = TRUE)

### Compute summary statistics
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

```
### Filter all the stations installed in the city (LAU) of Milano (Italy)
IDstations <- EEAaq_get_stations(byStation = FALSE, complete = FALSE)
`%>%` <- dplyr::`%>%`
IDstations <- IDstations %>%
                dplyr::filter(LAU_NAME == "Milano") %>%
                dplyr::pull(AirQualityStationEoICode) %>%
                unique()
### Download NO2 measurement for the city of Milano from January 1st to December 31st, 2023
data <- EEAaq_get_data(IDstations = IDstations, pollutants = "NO2",
                       from = "2023-01-01", to = "2023-01-31", verbose = TRUE)

### Monthly aggregation: compute station-specific monthly minimum,
## average, and maximum NO2 concentrations
t_aggr <- EEAaq_time_aggregate(data = data, frequency = "monthly",
                               aggr_fun = c("mean", "min", "max"))

### Weekly aggregation: compute station-specific monthly average
##and standard deviation concentrations
t_aggr <- EEAaq_time_aggregate(data = data, frequency = "weekly",
                               aggr_fun = c("mean", "sd"))
```

`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 PM10 data for the province (NUTS-3) of Milano (Italy)
# from January 1st to January 31st, 2023
`%>%` <- dplyr::`%>%`
IDstations <- EEAaq_get_stations(byStation = TRUE, complete = FALSE)
IDstations <- IDstations %>%
                dplyr::filter(NUTS3 %in% c("Milano")) %>%
                dplyr::pull(AirQualityStationEoICode) %>%
                unique()
data <- EEAaq_get_data(IDstations = IDstations, pollutants = "PM10",
                       from = "2023-01-01", to = "2023-01-31", 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