

ARPALData: an R package for retrieving and analyzing air quality and weather data from ARPA Lombardia (Italy)

Supplementary Material S2

Technical note on ARPALData functions

Release 1.5.0
Updated on December, 2023

Paolo Maranzano^{1,2*} and Andrea Algieri³

^{1*}Department of Economics, Management and Statistics (DEMS),
University of Milano-Bicocca (UniMiB), Piazza dell'Ateneo Nuovo, 1,
Milano, 20126, Italy.

²Fondazione Eni Enrico Mattei (FEEM), Corso Magenta, 63, Milano,
20123, Italy.

³Department of Lodi and Pavia, U.O. Attività produttive e controlli,
Agenzia Regionale per la Protezione dell'Ambiente Lombardia (ARPA
Lombardia), Via Nino Bixio, 13, Pavia, 27100, Italy.

*Corresponding author(s). E-mail(s): paolo.maranzano@unimib.it;

A note on the ARPALData software

In this note we discuss in detail the technical characteristics of the software **ARPALData**, a R package (R Core Team, 2020) aiming at retrieving, handling and analyzing air quality and weather data for the Lombardy region, Northern Italy. The note intends to support the main paper **ARPALData: an R package for retrieving and analyzing air quality and weather data from ARPA Lombardia (Italy)** (*Environmental and Ecological Statistics*, 2024).

It is worth noting that **ARPALData** complies with the CRAN policy that requires packages using Internet resources to fail gracefully with an informative message if the resource is not available or has changed. In addition, many of the functions admit an optional **verbose** argument that prints some auxiliary information on the screen that informs the user about which operations the command is performing and some object-specific information.

The *ARPALData* software is freely available on the Comprehensive R Archive Network (CRAN) of R at the webpage <https://cran.r-project.org/web/packages/ARPALData/index.html> since July 2021. The current available release is 1.5.0 (September 2023). *ARPALData* runs on Microsoft Windows, Linux and Apple Mac computers. No special hardware is required to run the software other than a standard desktop computer. Future releases of the software will potentially include stations or sensors selection using coordinates (i.e., quadrants, single coordinates, NUTS levels); interactive and dynamic maps; and spatial interpolation (e.g., IDW) for the whole region or a user-defined area.

In order to extend the discussion about the need to develop a unified platform at the national level, since 2023 part of the **ARPALData** team has been involved in the development of a complementary software to retrieve air quality data at the European level, namely **EEAaq** (Tassan Mazzocco et al, 2023), from the open database of the European Environment Agency.

List of available functions from ARPALData

The *ARPALData* package provides the users with 17 functions, which can be re-grouped into four different categories: 1) Download air quality and weather measurements; 2) Data exploration, quality check and graphical representation; Class check; 4) Auxiliary functions. The download functions aim at downloading and formatting the observations according to several filtering inputs provided by the user. The format checking functions aim to verify that the input objects belong to appropriate object classes valid for the application of the analysis functions. In fact, some representation or analysis functions developed in the package are only applicable to objects of class *ARPALdf*. The analysis and representation functions allow to compute descriptive statistics and graphical representation of the input data. Eventually, other functions are provided within the package as auxiliary commands for specific tasks.

Command	Category	Description
<code>get_ARPA_Lombardia_AQ_data</code>	Download	Retrieves air quality data from the ground monitoring network
<code>get_ARPA_Lombardia_W_municipal_data</code>	Download	Retrieves meteorological data from the ground monitoring network
<code>get_ARPA_Lombardia_AQ_municipal_data</code>	Download	Retrieves air quality data at the municipal level
<code>get_ARPA_Lombardia_AQ_registry</code>	Download	Retrieves the metadata/registry for ground air quality network
<code>get_ARPA_Lombardia_W_registry</code>	Download	Retrieves the metadata/registry for ground weather network
<code>get_ARPA_Lombardia_AQ_municipal_registry</code>	Download	Retrieves the metadata/registry for air quality data at the municipal level
<code>ARPALdf_Summary</code>	Analysis	Computes summary descriptive statistics by station and by year, and produces descriptive plots of the sample
<code>ARPALdf_Summary_map</code>	Analysis and Mapping	Maps (gereference) summary descriptive statistics (see <code>ARPALdf_Summary</code>) by station, by year or by municipality
<code>map_Lombardia_stations</code>	Mapping	Maps (georeference) the coordinates of the stations in a <i>ARPALdf</i> object
<code>get_Lombardia_geospatial</code>	Mapping	Returns the geometries of Lombardy at several levels (from LAU, NUTS3 and NUTS2)
<code>get_ARPA_Lombardia_zoning</code>	Mapping	Returns the geometries of the ARPA Lombardia's zoning of the region (see the main paper for details)
<code>Time_aggregate</code>	Auxiliary	Aggregates an <i>ARPALdf</i> object to a lower time frequency (e.g., from daily to monthly) using several aggregation functions
<code>registry_KNN_dist</code>	Auxiliary	Maps (gereference) summary descriptive statistics (see <code>ARPALdf_Summary</code>) by station, by year or by municipality
<code>is_ARPALdf</code>	Class check	Checks if a R object is of class <i>ARPALdf</i>
<code>is_ARPALdf_AQ</code>	Class check	Checks if a R object is of class <i>ARPALdf_AQ</i>
<code>is_ARPALdf_W</code>	Class check	Checks if a R object is of class <i>ARPALdf_W</i>
<code>is_ARPALdf_AQ_mun</code>	Class check	Checks if a R object is of class <i>ARPALdf_AQ_mun</i>

Table S1 List of ARPALData available for the users.

A. Data download functions

The `ARPALData` package contains three functions that allow for downloading data from the ARPA Lombardia database. Each function is identified by the prefix `get_ARPA_Lombardia`, while the suffix determines the type of data to be downloaded:

- `AQ_data` returns the observed air quality measurements collected by the ARPA Lombardy ground-based survey system;
- `W_data` returns the observed meteorological measurements collected by the ground-based monitoring network;
- `AQ_municipal_data` returns the air quality estimates for each municipality in Lombardy.

The `get_ARPA_Lombardia` commands share the same set of input arguments, which enable the user to customize the query regarding the variables and period of interest, the sampling frequency, and the grouping statistics. For instance, it is possible to concurrently download the average daily concentration of a certain pollutant and the maximum daily concentration of another pollutant. The period of interest is specified by the user through the start and the end dates (optionally stating the exact time), meaning that it is possible to download measurements pertaining to infra-annual sub-periods, but also to periods crossing one or more years. The user can also choose whether to consider the entire monitoring network (or all municipalities) or only a subset by entering the codes of the stations of interest. These codes are provided through the metadata functions presented in the following sections, which enable filtering and selection of stations based on user-specific criteria.

The input arguments are the following:

- `IDStation`: (numeric) indicates the identification number of the stations of interest to be downloaded. If set to `NULL` (default), the entire regional network is downloaded. The list of active sensors and stations can be retrieved from the metadata functions;
- `Date_begin`: (character) the first date-time to download. Date must be either declared using `"YYYY-MM-DD"` or `"YYYY-MM-DD hh:mm:ss"` format. Default is `Date_begin = "2022-01-01"`;
- `Date_end`: (character) the last date-time to download. Date must be either declared using `"YYYY-MM-DD"` or `"YYYY-MM-DD hh:mm:ss"` format. Default is `Date_end = "2022-12-31"`;
- `Frequency`: (character) the temporal frequency at which the observations are desired. It can be `"10mins"` (only for weather data), `"hourly"`, `"daily"`, `"weekly"`, `"monthly"` or `"yearly"`. Default is `Frequency = "hourly"`;
- `Var_vec`: (character) indicates which variables should be downloaded. If set to `NULL`, all the available variables are included in the output. If `NULL` (default) all the variables are averaged;
- `Fns_vec`: (character) indicates the aggregation function to apply to the selected variables. Available functions are `mean` (default), `median`, `min`, `max`, `sum` and `qPP` (for the PP^{th} percentile). `Var_vec` and `Fns_vec` must have the same number of elements;

- **by_sensor**: (logic) indicates whether the output `data.frame` should be structured in two-way panel format (`by_sensor = FALSE`) or in a three-way panel format (`by_sensor = TRUE`);
- **parallel**: (logic) indicates if the download has to be performed serially (`parallel = FALSE`) or using parallel processing (`parallel = TRUE`). If '`parallel = FALSE`' (default), data downloading is performed using a sequential/serial approach and additional parameters '`parworkers`' and '`parfuturetype`' are ignored. When '`parallel = TRUE`', data downloading is performed using parallel computing through the `Futureverse` setting. More detailed information about parallel computing in the `Futureverse` can be found at the following webpages: <https://future.futureverse.org/> and <https://cran.r-project.org/web/packages/future.apply/vignettes/future.apply-1-overview.html>;
- **parworkers**: (numeric value to be considered only if `parallel = TRUE`, i.e., parallel mode is active) declares the number of parallel workers to be activated for data download. If '`parallel = TRUE`' (parallel mode active), the user can declare the number of parallel workers to be activated using '`parworkers = integer number`'. By default ('`parworkers = NULL`'), the number of active workers is half of the available local cores;
- **parfuturetype**: (character value to be considered only if `parallel = TRUE`, i.e., parallel mode is active), declares the parallel computing strategy to be used according to the `Futureverse` syntax. If '`parallel = TRUE`' (parallel mode active), the user can declare the parallel strategy to be used according to the `Futureverse` syntax through '`parfuturetype`'. By default, the '`multisession`' (background R sessions on local machine) is used. In alternative, the '`multicore`' (forked R processes on local machine. Not supported by Windows and RStudio) setting can be used;
- **verbose**: (logic) toggle warnings and messages. If `verbose = TRUE` (default) the function prints on the screen some messages describing the progress of the tasks. If `verbose = FALSE` any message about the progression is suppressed.

Output

Download functions return as output an object of the classes `data.frame` and `ARPALdf` fully compatible with the *Tidyverse* (Wickham et al, 2019) and the graphical environment `ggplot2` (Wickham, 2016). Recall that each sensor measures one and only one meteorological or pollutant parameter and that a one-to-many relationship exists between stations and sensors. In addition, multiple pollutants or weather parameters can be measured at each station. The default output object is a long-shaped (or two-way panel) `data.frame` with dimensions $(N \times T) \times (K + 3)$, where T is the number of time stamps, N is the number of stations (or municipalities), and K the number of available measurements (airborne pollutants or weather variables). The first three columns are the time stamps (`Date`), the station unique identification numbers (`IDStation`) and the station names (`NameStation`), respectively. Notice that, the two-way panel takes on an exactly rectangular shape as for each station the same number of time stamps is reported. Thus, if a certain parameter is not measured at a specific station, a sequence of missing values is reported for all the time stamps. These missing values should not be subjected to any imputation analysis because they belong to

Parallel download

Data for air quality and weather are retrieved through the API Socrata Open Data (SODA), which provides a systematic access to Lombardy Region database, including the capability to filter, query, and aggregate data. ARPALData accesses to the Socrata functions through the package `RSocrata` (Devlin et al, 2023). Using appropriate queries, the API allows filtering the stations of interest (the parameter `IDStation` admits numeric values corresponding to the unique identification number of each station) and the subperiod of dates to be downloaded (set via `Date.begin` and `Date.end`).

By default, the software retrieves measurements serially as follows. Given a specific date range as input, to speed up data access, the interval is broken into twelve¹ non-overlapping contiguous sub-intervals. For each sub-interval, an appropriate query is constructed to sequentially interrogate the database. Finally, the resulting data blocks are stacked into a single `data.frame` object.

However, all the `get_ARPA.Lombardia_*.data` functions can adopt a parallel computation strategy (by setting `parallel = TRUE`) to improve speed of download. Parallel computation is performed through the `Futureverse` (Bengtsson, 2021), implemented within the packages `future` and `future.apply`. When the parallel mode is active, the user can declare the number of parallel workers to be activated using `parworkers = integer number`. By default (`parworkers = NULL`), the number of active workers is half of the available local cores. Also, the user can declare the parallel strategy to be used according to the `Futureverse` syntax through `parfuturetype`. By default, the "multisession" option (background R sessions on local machine) is used. As alternative, the "multicore" setting (forked R processes on local machine; not supported by Windows and RStudio) can be used.

We recall that, some large data sets or complex analyses, download step may require a remarkable computation time. In particular, when using the `get_ARPA.Lombardia.W.data` function to download meteorological data (which are provided at a 10 minutes frequency), the download time can be high even in parallel mode. It is therefore advisable to break long sequences of dates into smaller sequences (e.g., one year at a time) and repeat the download in a loop.

Examples

We refer the reader to extended vignette supporting the main paper as [Supplementary Material 1](#) or at the GitHub webpage <https://github.com/PaoloMaranzano/ARPALData>.

¹The number of blocks (twelve) was determined by the authors so that given a one-year interval, each block corresponds to approximately one month of observations.

B. Metadata

Metadata (or registry) about the monitoring network owned by ARPA Lombardia can be easily retrieved using the `get_ARPA_Lombardia_*.registry` functions. Indeed, `AQ_registry`, `W_registry` and `AQ_municipal_registry`, return the explanatory metadata for the air quality monitoring stations, the weather monitoring stations and for the municipalities, respectively. These functions do not need input arguments. Available metadata include: sensors and stations unique identification number (`IDSensor` and `IDStation`), station names (`NameStation`), pollutant name or weather parameter associated with the sensor (`Pollutant` or `Measure`), geographical coordinates (`Altitude`, `Latitude`, and `Longitude`), `Province` and `City` (municipality).

Metadata may evolve over time as a result of updates carried out by the regional agency on the stations. However, the software preserves the entire history of the monitoring networks as it contains information on both currently active (as of the date of the request) and disabled sensors. Specifically, for all sensors, the date of initial operation is available, while for disabled sensors, the date of deactivation is also given. The registries contain information on both the currently active sensors (`DateStop` is missing) and dis-activated sensors (`DateStop` is a non-null date).

The following code provides an example of how to get metadata for the three cases:

```
# Obtain metadata for weather ground monitoring network
get_ARPA_Lombardia_AQ_registry()
# Obtain metadata for air quality ground monitoring network
get_ARPA_Lombardia_AQ_registry()
# Obtain metadata for air quality at municipal level
get_ARPA_Lombardia_AQ_municipal_registry()
```

For the specific case of air quality stations, metadata additionally provides information on the ARPA zone classification and the station type classification. On the one hand, the stations type classification classifies the sites according to the EEA stations type taxonomy ([European Environmental Agency, 2023](#)), that is the stations are classified by the distribution/density of building in the area surrounding the station (Urban, Suburban, and Rural) and by predominant emission sources (Background, Traffic, and Industrial). On the other hand, the ARPA zone classification ([ARPA Lombardia, 2023](#)) has been defined by the Lombardy Region authorities through the Regional Decree 2605/2011 ([Giunta regionale della Lombardia, 2011](#)), and distinguished the regional territory into homogenous areas or agglomerations. The zoning reflects the main morphological characteristics of the regional territory and are used to evaluate the local compliance with national and international environmental objectives and limit values. Lombardy region is classified into seven subregions: three large urbanized areas (metropolitan areas of Milano, Bergamo and Brescia), urbanized areas in rural contexts (namely, the urbanized plain), rural areas (namely, the rural plain), mountainous areas, and valley bottoms. In the following sections, we will show how to retrieve the spatial geometries of the zoning for Lombardy.

C. Data exploration and graphical analysis

ARPALData package offers several function to explore, analyze and represent air quality and weather measurement accessed and processed through the package. The rationale is that the user should be able to verify through simple, but effective, descriptive tools the statistical properties and quality of the data of interest right after the download. The exploratory tools that **ARPALData** offers are of four categories: (1) descriptive statistics and graphs to summarize and depict the measured values both globally (over the whole sample) and locally (site-specific or year-specific statistics and graphs); (2) analytical tools and graphs to detect outliers or out-of-scale values (outlier detection analysis); (3) tools describing patterns of missing data to distinguish between cases of missing at random or structural missing caused, for example, by failures in the monitoring network; and (4) correlation analysis between variables at global level (over the whole sample) and local level (correlation for individual monitoring sites).

The above mentioned tools are available to the users through the command **ARPALdf_Summary**, which can be applied to any object of class **ARPALdf**. The command provides the user with both descriptive statistics summarizing the dataset and plots. The function pursues the goal to provide an overview of the statistical quality of the data at hand. The function can be customized by the user by properly setting the arguments, each of them providing information on specific statistical tools to be employed. Moreover, the functions admits a verbose setting, which prints on the screen some auxiliary information useful to understand the structure of the **data.frame**, such as the total number of observations, the number of monitoring sites included, the number of time stamps and the time frequency.

Output object

ARPALdf_Summary returns a list of **data.frames** containing summary descriptive statistics for a data frame of class **ARPALdf**. Summary statistics are computed for the overall sample (**Descr**), by Station ID (**Descr_by_IDStat**) and by year (**Descr_by_Year**). Eventually, by setting **verbose = T**, **ARPALdf_Summary** prints on the screen some auxiliary information useful to understand the structure of the **data.frame**, such as the total number of observations, the number of monitoring sites included, the number of time stamps and the time frequency.

Descriptive statistics

By default, **ARPALdf_Summary** returns for each variable included a set of central tendency measures (e.g., mean, median, quantiles), variability indicators (e.g., standard deviation and range) and descriptive statistics about missing values (e.g., count, percentage compared to total observations). However, the user can automatically compute the essential descriptive statistics (indicators of central tendency, variability, shape of distribution, and missing values) for all the variables in the dataset by station and by year. Statistics are calculated at overall/full level (default), by station (**by_IDStat = TRUE**) and by year (**by_Year = TRUE**). Furthermore, when more than one numerical

variable is present in the dataset, the user can compute the Person’s linear correlation index (argument `correlation = TRUE`) among all possible pairs of variables by station and for the overall sample.

Outlier detection

When the outlier detection analysis is activated (`outlier = FALSE`), the function returns an overview of the (potential) outlier values detected using the Hampel Filter (Pearson et al, 2016) and the box-plot rule. Let us denote a sequence of T measurements by X_t with $t = 1, \dots, T$. The Hampel Filter (or identifier) classifies as outliers all observations that fall outside the range $\pm 3 \times MADM$, where $MADM$ is the median absolute deviation from the median (i.e., $1.4826 \times \text{Median}(|X_t - \text{Median}(X_t)|)$). Notice that the correction factor 1.4826 adjusts the distance so that on average, the $MADM$ estimate is equal to the standard deviation for Gaussian data. On the other side, the box-plot rule classifies as outliers all the observations above the *typical upper value* $X_{0.75} + 1.5 \times IQR$ or below the *typical lower value* $X_{0.25} - 1.5 \times IQR$, where $IQR = X_{0.75} - X_{0.25}$ is the interquartile range computed as the difference between the 75° percentile of X_t (i.e., $X_{0.75}$) and the 25° percentile of X_t (i.e., $X_{0.25}$). Both identifiers use the median as the *typical* value because it is robust to both the presence of outliers and the positive skewness that environmental data are most likely to show (Mudelsee and Alkio, 2007). In practice it often happens that the box-plot rule is more moderate in identifying outliers than the Hampel Identifier. Therefore, their comparison and a potential disagreement can serve as robustness check for the user in a situation of uncertainty. Practically, the user is provided with the threshold values and the percentage of observations exceeding these values for each variable.

Missing values and gap length statistics

The user can also activate an exploratory missing data analysis based on gap length statistics (`gap_length = TRUE`). In particular, the function returns information about the temporal distribution of missing values (i.e., sequences of missing values) for each variable by station. Since we are dealing with time series data, we are interested in analyzing the gap length, that is, the temporal distance between one observation and the following. In particular, for each variable and for each station, the function summarizes the pattern of missing values by reporting summary statistics (e.g., average, variability, and quantiles of gap lengths) in the same temporal unit of the data (measured in hours, days, weeks or months depending on the frequency of the input data). For example, if the input dataset has a daily frequency, the gap statistics will be expressed in terms of days. Missing patterns can be detected by analyzing the gap statistics. For instance, if at a certain monitoring site the mean value of the gap length is greater than the minimum value, it means that at least one observation in the time series is missing. When other statistics, such as the median, maximum, percentiles, also differ from each other, then the number of missing observations increases. When all statistics are equal to 1, then the variable of interest does not present missing values.

Graphical analysis

The data exploration phase can be greatly enhanced and supplemented through the use of graphical analysis by station and for the overall sample. On the one side, `ARPALdf.Summary` allows to produce distributional box-plots (default), histograms (`histogram = TRUE`), and kernel density plots (`density = TRUE`) computed on the full sample. On the other, the user can employ the command `ARPALdf.Summary_map` to represent on a geo-referenced map any of the descriptive statistics computed by `ARPALdf.Summary` for the available monitoring sites (or for municipalities if municipal data are used). Consider, for example, a user who wants to depict on a map the average level of a certain pollutant over a given period for each available control unit and compare it with the amount of rainfall accumulated over the same period on the meteorological network. Also, consider a user interested in plotting the share of missing values of a certain variable by station and to compare it with the average site-specific gap lengths.

Examples

We refer the reader to extended vignette supporting the main paper as [Supplementary Material 1](#) or at the GitHub webpage <https://github.com/PaoloMaranzano/ARPALData>.

D. Temporal aggregation

Consider a generic `ARPALdf` object containing measurements at a given temporal frequency. Also, recall that air quality measurements are originally provided at the hourly frequency (e.g., `NO2` or `Ozone`) or daily frequency (e.g., `PM10` and `PM2.5`), whereas weather measurements are all provided with a 10 minutes frequency. The command `Time_aggregate` allows the user to reduce the temporal sampling rate (specified in `Frequency`) of all or user-selected variables (specified in `Var_vec`) belonging to the object by aggregating the observations to a specific temporal frequency (i.e., hourly, daily, weekly, monthly, or yearly) at each monitoring site according to a set of available aggregation functions (specified in `Fns_vec`) among the user can choose. In addition to the applicability to a generic object of class `ARPALdf`, `Time_aggregate` is automatically used when the user customize the download query by setting the argument `Frequency` to something different from `hourly` (default).

In the current release, sample measurements can be aggregate using the following descriptive statistics: average, median, minimum, maximum, sum, quantile, standard deviation, variance, variability coefficient, skewness, and kurtosis. Particularly interesting are the aggregations of wind speed and direction. In fact, for wind speed and direction only mean, min, and max aggregations are available. Moreover, their aggregation must always concern the two variables jointly and never individually. Calculation of the minimum and maximum require only identification of the speed and direction in which the strongest speed was recorded (maximum and minimum of the direction taken individually lose their meaning). For the averaging, however, the circular nature of the direction is preserved by computing the vector average involving the east-west (u) and north-south (v) components and the wind flow (Grange, 2014). The aggregation procedure is automatically applied during the data download stage when the user wants to customize the query so that the output dataset is aggregated to a specific temporal frequency (i.e., hourly, daily, weekly, monthly, or yearly).

For example, it is possible to aggregate the daily `NO2` values contained in a `data.frame` called `Data_input` by calculating weekly averages and maximum values using the following syntax:

```
# Aggregate daily NO2 values to weekly averages and max values
Time_aggregate(Dataset = Data_input,
               Frequency = "weekly",
               Var_vec = c("NO2", "NO2"),
               Fns_vec = c("mean", "max"))
```

E. Mapping and geometries

Given as input an object of class `ARPALdf`, it is possible to map the locations (through their latitude and longitude) of the monitoring sites included in the sample by using the command `map_Lombardia_stations`. When the command is applied to an `ARPALdf` object containing single time raw data (i.e., data referring to a single specific time stamp) from one of `get_ARPA_Lombardia_*.data` functions, it will produce a map of the values at each of the available monitoring sites.

The function `get_Lombardia_geospatial` returns a polygon `sf` object containing the geometries of Lombardy at different spatial level. The available spatial levels are expressed according to the Eurostat's Nomenclature of Territorial Units for Statistics (NUTS) classification ([Eurostat, 2023](#)). The function has a single parameter `NUTS_level`, which specifies the desired NUTS level, i.e. `NUTS_level = "LAU"` for municipal geometries, `NUTS_level = "NUTS3"` for provincial geometries and `NUTS_level = "NUTS2"` for the regional geometries. Similarly, the function `get_ARPA_Lombardia_zoning` returns the geometries (polygonal shape file, `sf` object) and a map of the ARPA zoning of Lombardy.

Eventually, the user can retrieve the exact boundaries of ARPA zones through the function `get_ARPA_Lombardia_zoning`, which returns the geometries (polygonal shape file, `sf` object) and a map of the ARPA zoning for Lombardy.

The zoning geometries can be retrieved and plotted using the following code:

```
zoning <- ARPALData::get_ARPA_Lombardia_zoning(plot_map = FALSE)
p1 <- ggplot(data = zoning, aes(fill = Zone)) +
  geom_sf() +
  labs(title = "ARPA Lombardia zoning", x = "Longitude", y = "Latitude") +
  theme(legend.position = "bottom", legend.text = element_text(size = 9)) +
  guides(fill=guide_legend(nrow=7,byrow=TRUE))
```

Geometries for municipalities (LAUs) and provinces (NUTS3) can be retrieved using the following commands:

```
LAU <- ARPALData::get_Lombardia_geospatial(NUTS_level = "LAU")
p2 <- ggplot(data = LAU) +
  geom_sf() +
  labs(title = "LAUs of Lombardy", x = "Longitude", y = "Latitude") +
  theme(legend.position = "right", legend.text = element_text(size = 9))
NUTS3 <- ARPALData::get_Lombardia_geospatial(NUTS_level = "NUTS3")
p3 <- ggplot(data = NUTS3) +
  geom_sf() +
  labs(title = "Provinces of Lombardy", x = "Longitude", y = "Latitude") +
  theme(legend.position = "right", legend.text = element_text(size = 9))
```

Eventually, we can use `map_Lombardia_stations` to plot the locations of air quality stations monitoring PM_{10} .

```
### Download registry table of air quality monitoring sites
reg <- get_ARPA_Lombardia_AQ_registry()
### Filter sensors:
#   Stations activated before 2019
#   Currently on service / active
#   Measuring PM10
reg <- reg %>%
  filter(DateStart <= "2019-01-01",
         is.na(DateStop),
         Pollutant %in% c("PM10"))
p4 <- ARPALData::map_Lombardia_stations(data = reg)
```

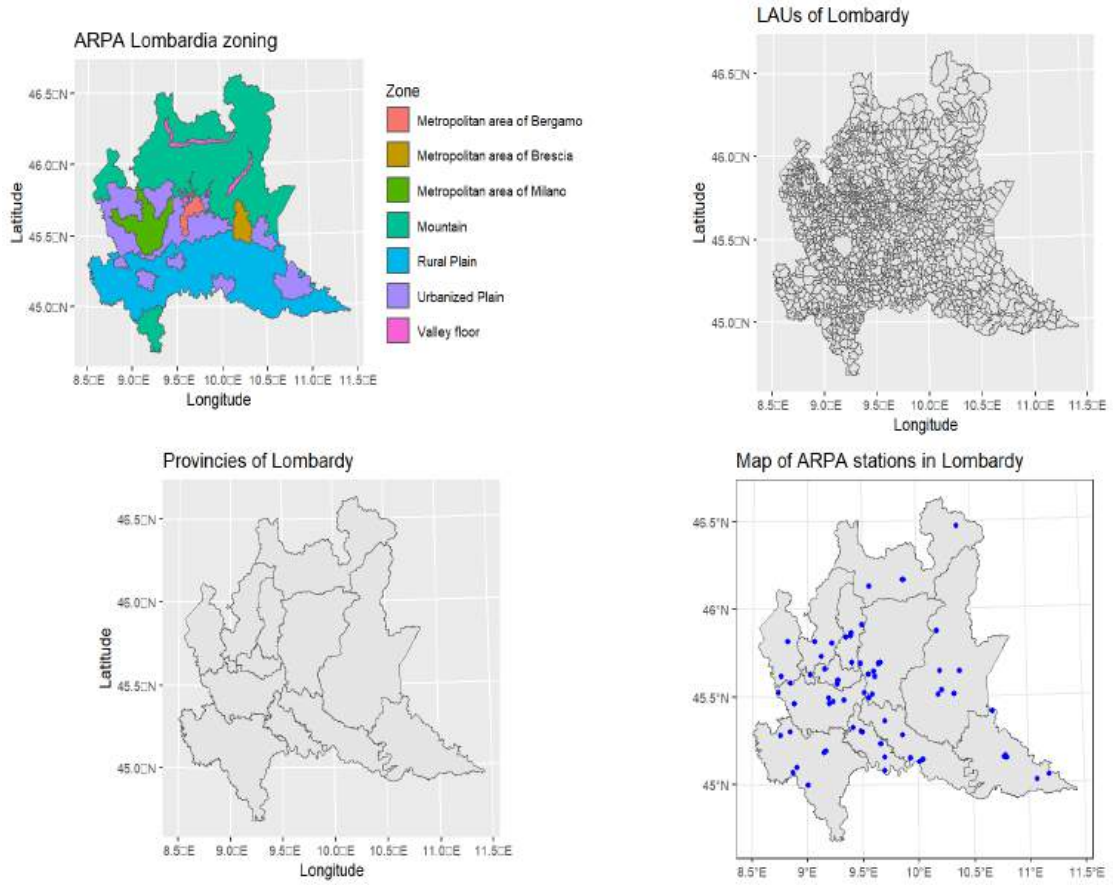


Figure S3 Combined maps produced using ARPALData

F. Locations matching

The `registry_KNN_dist` function allows to identify, for each site included in the `data.frame` X, the K-nearest-neighbors among those included in the `data.frame` Y. The matching is performed by computing the Geodesic (or Great Circle) distance between the coordinates of the points. This operation is very useful when one intends to detect the closest meteorological monitoring sites to specific air quality stations. In fact, the two networks are heterotopic and only in a few cases a site monitors both pollution and weather. In this way, it is possible to know which sites perform a dual function and at what distance the closest stations are located. The `reg.X` and `reg.Y` arguments of this function must necessarily be two registry tables, while the K value indicates how many neighbors are to be scanned.

In the following example we aim at finding the 3-nearest-neighbors weather stations measuring temperature and wind speed for only the air quality stations measuring PM_{10} .

```

### Download registry table of air quality monitoring sites
regAQ <- get_ARPA_Lombardia_AQ_registry()
### Filter sensors:
#       Stations activated before 2019
#       Currently on service / active
#       Measuring PM10
regAQ <- regAQ %>%
  filter(DateStart <= "2019-01-01",
         is.na(DateStop),
         Pollutant %in% c("PM10"))

### Download registry table of air quality monitoring sites
regW <- get_ARPA_Lombardia_W_registry()
### Filter sensors:
#       Stations activated before 2019
#       Currently on service / active
#       Measuring temperature and wind speed
regW <- regW %>%
  filter(DateStart <= "2019-01-01",
         is.na(DateStop),
         Measure %in% c("Temperature", "Wind_speed"))

### Matching the first 3-NN
match3 <- registry_KNN_dist(regAQ, regW, k=3)[[1]]

```

G. Functions for checking object classes

The package includes four auxiliary functions (i.e., `is_ARPALdf`, `is_ARPALdf_AQ`, `is_ARPALdf_W`, and `is_ARPALdf_AQ_mun`) checking the R classes of an input object. The output of any *ARPALData* data download and reshape function belongs to the `ARPALdf` class. When the `data.frame` concerns air quality data, in addition to the previous one, a further class `ARPALdf_AQ` is appended, while if it concerns meteorological data the class `ARPALdf_W` is added. Finally, for municipal air quality data, the `data.frame` will also belong to the class of type `ARPALdf_AQ_mun`.

Such functions are necessary to ensure that the inputs of the data analysis, representation and reshape functions contain the appropriate information (e.g. station identifiers, date, names of the measurements) in order to execute them correctly. For example, in geographic mapping of selected stations it is necessary that the input `data.frame` contains the columns `Longitude` and `Latitude`, typical elements of an object of class `ARPALdf`.

References

- ARPA Lombardia (2023) Mappa della zonizzazione della lombardia (accessed on september 22nd 2023). URL <https://www.arpalombardia.it/temi-ambientali/aria/rete-di-rilevamento/zonizzazione/>
- Bengtsson H (2021) A unifying framework for parallel and distributed processing in r using futures. The R Journal 13(2):208–227. <https://doi.org/10.32614/RJ-2021-048>, URL <https://doi.org/10.32614/RJ-2021-048>
- Devlin H, D. P, Schenk, Jr. T, et al (2023) RSocrata: Download or Upload 'Socrata' Data Sets. URL <https://CRAN.R-project.org/package=RSocrata>, r package version 1.7.15-1
- European Environmental Agency E (2023) Classification of monitoring stations and criteria to include them in eea's assessments products. Report, European Environmental Agency, EEA, URL <https://www.eea.europa.eu/themes/air/air-quality-concentrations/classification-of-monitoring-stations-and>
- Eurostat (2023) Nuts - nomenclature of territorial units for statistics (accessed on september 22nd, 2023). URL <https://ec.europa.eu/eurostat/web/nuts/background>
- Grange S (2014) Technical note: Averaging wind speeds and directions. <https://doi.org/10.13140/RG.2.1.3349.2006>
- Giunta regionale della Lombardia (2011) Delibera di giunta regionale (dgr) n. 2605 del 30 novembre 2011. URL <https://www.regione.lombardia.it/wps/portal/istituzionale/HP/DettaglioRedazionale/servizi-e-informazioni/Enti-e-Operatori/ambiente-ed-energia/Inquinamento-atmosferico/zonizzazione-territorio-regionale/zonizzazione-territorio-regionale>
- Mudelsee M, Alkio M (2007) Quantifying effects in two-sample environmental experiments using bootstrap confidence intervals. Environmental Modelling & Software 22(1):84–96. <https://doi.org/10.1016/j.envsoft.2005.12.001>
- Pearson RK, Neuvo Y, Astola J, et al (2016) Generalized hampel filters. EURASIP Journal on Advances in Signal Processing 2016(1):1–18
- R Core Team (2020) R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, URL <https://www.R-project.org/>
- Tassan Mazzocco A, Maranzano P, Borgoni R (2023) EEAAq: Handle Air Quality Data from the European Environment Agency Data Portal. URL <https://CRAN.R-project.org/package=EEAAq>, r package version 0.0.3
- Wickham H (2016) ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, URL <https://ggplot2.tidyverse.org>

Wickham H, Averick M, Bryan J, et al (2019) Welcome to the tidyverse. Journal of Open Source Software 4(43):1686. <https://doi.org/10.21105/joss.01686>