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

Supplementary Material S1

Guided examples and case studies

Release 1.5.0
Updated on December, 2023

Paolo Maranzano[1,2*] and Andrea Algieri[3]

[1]Department of Economics, Management and Statistics (DEMS), University of Milano-Bicocca (UniMiB), Piazza dell'Ateneo Nuovo, 1, Milano, 20126, Italy.
[2]Fondazione Eni Enrico Mattei (FEEM), Corso Magenta, 63, Milano, 20123, Italy.
[3]Department 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;

Guided examples and case studies for ARPALData

In this vignette we present three guided examples in which **ARPALData** is used to retrieve and analyze air quality and weather data across Lombardy. Data and codes for the three examples have been produced on December 7th, 2023. The full **R** vignette of the three examples is also available at the GitHub webpage https://github.com/PaoloMaranzano/ARPALData.

We initialize the code by recalling the necessary **R** libraries: **tidyverse** for tidy data manipulation, **ggplot2** and **ggpubr** graphical representations, and **latex2exp** for symbols in texts and plots (Meschiari, 2022).

```
### call required packages
require( ARPALData)
require(tidyverse)
require(ggplot2)
require(ggpubr)
require(latex2exp)
```

Case study 1: describing air quality according to the local context

In the first example, we download and analyze weekly concentrations of PM_{10} observed across the whole air quality ground monitoring network. We are interesting in describing the average and maximum weekly concentrations from 2019 to 2022 according to the contexts in which the stations are installed. To do so, we characterize PM_{10} concentrations according to the station types and the regional zoning described in the main paper. In addition, functions from the **Tidyverse** (Wickham et al, 2019) and **ggplot2** (Wickham, 2016) will be extensively used to show how the outputs of **ARPALData** are fully compatible with them.

We start by downloading the actual registry (metadata) concerning the air quality ground monitoring network. The registry is stored in a **data.frame** called **reg**. We are interested in retrieving information about the currently active sites monitoring PM_{10}, and installed before January 1st, 2019. The full list of stations is then filtered according to the above criteria.

```
### 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 NO2
reg <- reg %>%
  filter(DateStart <= "2019-01-01",
         is.na(DateStop),
         Pollutant %in% c("PM10"))
```

By computing the frequency table of the available monitoring sites, we notice that the number of stations is not evenly distributed among the station types. In fact, Surburban-Traffic (ST), Suburban-Industrial (SI), Urban-Industrial (UI), and Rural-Industrial (RI) count less than 3 stations each. Thus, to keep consistency when aggregating observations, we further filter the stations list by removing the above classes of sites. The final number of stations is 59.

```
### Stations type included in the sample
table(reg$ARPA_stat_type)

### Remove Suburban-Traffic, Suburban-Industrial, Urban-Industrial and Rural-
    Industrial due to low sample sizes
reg <- reg %>%
  filter(ARPA_stat_type %notin% c("ST","SI","UI","RI"))
```

We can now proceed with the data download step. For stations that comply with the filters set above (`ID_station = reg$IDStation`), we collect weekly observations (`Frequency = "weekly"`) of PM_{10} from 2019 to 2022 (`Date_begin = "2019-01-01"` and `Date_end = "2022-12-31"`). Measurements are downloaded under a parallel setup (`parallel = TRUE`). Original air quality data provided by ARPA Lombardia have an hourly frequency. To obtain the weekly average and weekly maximum concentrations, observations are aggregated by setting `Var_vec = c("PM10","PM10")` and `Fns_vec = c("mean","max")`. Once the desired dataset has been dowloaded, the metadata are associated to each measurement through a `left_join` operation (i.e., we keep only those stations listed in the raw data table `Data_AQ`) using as primary key the column `IDStation`.

```
### Download weekly average and weekly maximum of PM10 for all the available
    stations from 2019 to 2022. Parallel computation activated
Data_AQ <-  get_ARPA_Lombardia_AQ_data(ID_station = reg$IDStation,
                                       Date_begin = "2019-01-01",
                                       Date_end = "2022-12-31",
                                       Frequency = "weekly",
                                       Var_vec = c("PM10","PM10"),
                                       Fns_vec = c("mean","max"),
                                       parallel = TRUE)

### Associating metadata (ARPA type classification and ARPA zoning) to observations
reg_red <- reg %>%
  select(IDStation,ARPA_zone,ARPA_stat_type)
Data_AQ <- left_join(x = Data_AQ, y = reg_red, by = c("IDStation"))
```

To exploit the information about the local contexts in which stations are located, we aggregate the weekly average and weekly maximum values by station types and zoning, respectively. We start by aggregating the weekly average PM_{10} concentrations computing the lower and upper bounds (minimum and maximum), as well the central value (mean), by station type. The resulting time series are then plotted (Figure S1) into four sub-panels, each of them corresponding to a different type of station. For each panel, the maximum weekly concentrations are depicted in red, the average concentrations in green, while the minimum concentrations are in blue.

```
### Compute weekly descriptive statistics for average NO2 concentrations by station
    type:
# Weekly minimum of average PM10 by station type
# Weekly mean of average PM10 by station type
# Weekly maximum of average PM10 by station type
data_by_type <- Data_AQ %>%
  group_by(Date,ARPA_stat_type) %>%
  summarise(
    min_avgPM10 = min(PM10_mean, na.rm = T),
    mean_avgPM10 = mean(PM10_mean, na.rm = T),
    max_avgPM10 = max(PM10_mean, na.rm = T)
  ) %>%
  ungroup()

data_by_type %>%
```

```
  pivot_longer(cols = 3:last_col(), names_to = "Pollutant", values_to = "
    Concentrations") %>%
  ggplot(mapping = aes(x = Date, col = Pollutant)) +
  geom_line(mapping = aes(y = Concentrations)) +
  facet_wrap(~ ARPA_stat_type) +
  labs(x = "", y = TeX("$\\mu g/m^3$"),
       title = TeX("Weekly average PM$_{10}$ concentrations ($\\mu g/m^3$) by
    stations type")) +
  theme_bw() +
```

Interestingly, the charts show that average weekly concentrations exhibit different characteristics depending on the type of station considered. For instance, the weekly average PM_{10} concentrations observed at Urban-Traffic (UT) stations frequently exceed $75\mu g/m^3$ and are very compact during summer but more volatile in winter. In fact, the vertical distance between the maximum value (red series) and the minimum value (blue series) taken by the average is about $20\mu g/m^3$ during summer and doubles in winter. Similar considerations can be deduced for Suburban-Background (SB) stations. In contrast, at Rural-Background (RB) stations, average concentrations very rarely exceed $75\mu g/m^3$, but vary greatly in the winter months (note how the vertical distance between the minimum and maximum of the average in January 2020, 2021 and 2022, is about $50\mu g/m^3$).

Similarly to what we did for the type of stations, we now proceed by aggregating the weekly PM_{10} maxima by zone. To more clearly interpret the time series outputs, we start by downloading the shapefile of the zones through the command `get_ARPA_Lombardia_zoning`. The zones are then represented in a map using `ggplot2`. After that, the maximal concentrations are aggregated by calculating their maximum, minimum and mean value, and represented as time series. The series are divided into seven subpanels (one for each zone) in which the weekly minimum concentration (blue), weekly average concentration (green), and weekly maximum concentration (red) are depicted for each zone. Finally, the two graphs are placed side by side via `ggarrange` (Figure S2).

```
### Download ARPA Lombardia zoning shapefile and mapping
zoning <-  get_ARPA_Lombardia_zoning(plot_map = FALSE)
p_map <- 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))

### Compute weekly descriptive statistics for maximum PM10 concentrations by zoning
# Weekly minimum of maximum PM10 by zoning
# Weekly mean of maximum PM10 by zoning
# Weekly maximum of maximum PM10 by zoning
Data_by_zone <- Data_AQ %>%
  group_by(Date,ARPA_zone) %>%
  summarise(
    min_maxPM10 = min(PM10_max, na.rm = T),
    mean_maxPM10 = mean(PM10_max, na.rm = T),
    max_maxPM10 = max(PM10_max, na.rm = T)
  ) %>%
  ungroup()
Data_by_zone <- left_join(x = Data_by_zone, y = zoning %>% select(Cod_Zone,Zone),
                          by = c("ARPA_zone" = "Cod_Zone"))

p_ts <- Data_by_zone %>%
  pivot_longer(cols = contains("PM10"), names_to = "Pollutant", values_to = "
    Concentrations") %>%
```

Figure S1 Weekly PM_{10} minimum (blue), average (green), and maximum (red) concentrations observed from January 2018 to December 2021 by stations type.

```
  ggplot(mapping = aes(x = Date, col = Pollutant)) +
  geom_line(mapping = aes(y = Concentrations)) +
  facet_wrap(~ Zone) +
  labs(x = "", y = TeX("$\\mu g/m^3$"),
       title = TeX("Weekly maximum PM$_{10}$ concentrations ($\\mu g/m^3$) by ARPA
    zoning")) +
  theme_bw() +
  theme(legend.position = "")

# Combining plots
p_comb <- ggarrange(p_ts,p_map, ncol = 2, nrow = 1, widths = c(2,1))
print(p_comb)
```

Time series suggest that the maximum PM_{10} values are particularly high in the Milan metropolitan area (socio-economic center of the region with a high degree of urbanization and vehicles) and in the plains (both urbanized and rural). In the latter case, the particulate matter could be attributable to the numerous presence of agricultural activities and livestock farms, which, through the spreading of slurry and

animal manure, are responsible for the emission of large amounts of secondary particulate matter and ammonia (Fassó et al, 2023; Marongiu et al, 2023). The Alps to the north and the valleys have significantly lower maximum values. The other two urban agglomerations (Bergamo and Brescia) fall in an intermediate situation where the weekly PM_{10} maximum values are quite steady and less volatile. Stations located in the northern Alpine or pre-Alpine areas register significantly lower maximum values.

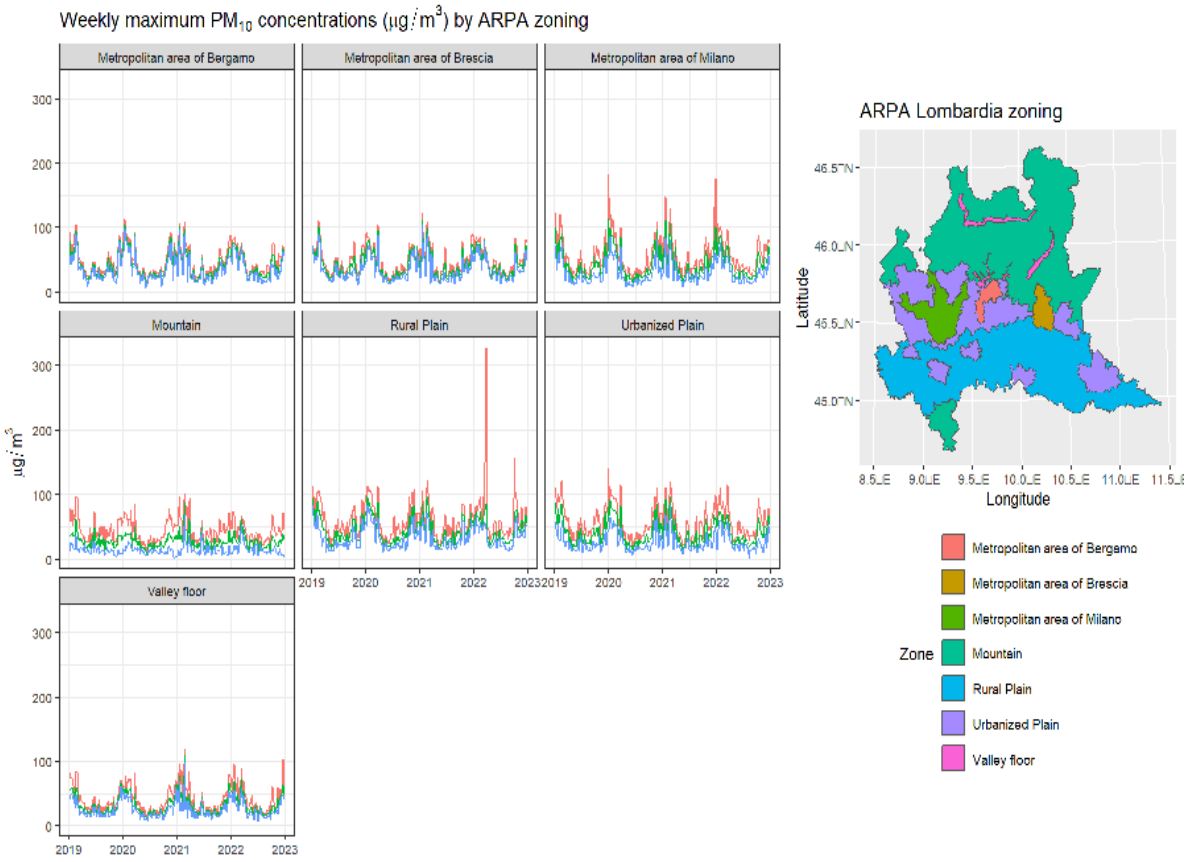

Figure S2 Weekly PM_{10} minimum (blue), average (green), and maximum (red) concentrations observed from January 2018 to December 2021 by ARPA zone.

Case study 2: air quality conditions during COVID-19 pandemic at municipal level in Lombardy

In the second example, we use `ARPALData` to investigate how the air quality improved across the region due to the shutdown imposed during the COVID-19 pandemic in

Spring 2020. Our goal is to compare the average values observed across the 1516 municipalities of Lombardy during the lockdown period (8^{th} March - 18^{th} May) with the concentrations observed in the same subperiod in 2018, 2019, and 2021.

As first step, we download daily (`Frequency = "daily"`) estimates of average NO_2 concentrations (`Var_vec = c("NO2_mean")` and `Fns_vec = c("mean")`) at the municipal level (`reg_AQ`) from January 2018 to December 2021 (`Date_begin = "2018-01-01"` and `Date_end = "2021-12-31"`). Retrieved data are store in a `data.frame` object called `Data_mun`.

```
### Download daily NO2 concentrations at municipal level from 2018 to 2020
Data_mun <-  get_ARPA_Lombardia_AQ_municipal_data(
  Date_begin = "2018-01-01",
  Date_end = "2021-12-31",
  Frequency = "daily",
  Var_vec = c("NO2_mean"),
  Fns_vec = c("mean"),
  verbose = TRUE
)
```

Then, the full dataset is filtered in order to preserve only the values estimated at each municipality during the lockdown subperiod (March 8^{th} to May 18^{th}) of 2018, 2019, 2020, and 2021 (the output dataset is `Data_spring`). The average concentrations of the period are then computed by aggregating the daily values using the function `Time_aggregate` with argument `Frequency = "yearly"`.

```
### Filter observations with respect to the date: only measures from 8th March to
    18th May of each year
Data_spring <- Data_mun %>%
  filter(Date >= "2021-03-08" & Date <= "2021-05-18" |
           Date >= "2020-03-08" & Date <= "2020-05-18" |
           Date >= "2019-03-08" & Date <= "2019-05-18" |
           Date >= "2018-03-08" & Date <= "2018-05-18")

### Computing period averages (8th March - 18th May) of NO2 concentrations
Data_y <-  Time_aggregate(
  Dataset = Data_spring,
  Frequency = "yearly"
)
```

Eventually, the average NO_2 concentrations observed at each municipality during the lockdown subperiod (March 8^{th} to May 18^{th}) of each year are represented on a map by employing the command `ARPALdf_Summary_map`. Each map adopt a color scale ranging from green (low concentrations), to yellow (mid concentrations) to red (high concentrations). To make the color scale uniform throughout maps, we set a common central value by imposing the middle point equal to the regional average concentrations over the period of interest in 2018 (i.e., `val_midpoint = mid_conc_2018`). The four maps produced by `ARPALdf_Summary_map` are then combined into a single graph with common title and legend through the `ggpubr` package.

```
### Computing average concentration of NO2 in 2018
mid_conc_2018 <- Data_y %>%
  filter(lubridate::year(Date) == 2018) %>%
  summarise(mean(NO2_mean,na.rm = T)) %>%
  pull()

# Map of NO2 concentrations in 2018 by municipality
map_18 <-  ARPALdf_Summary_map(
  Data = Data_y %>% filter(lubridate::year(Date) == 2018),
  Title_main =TeX("March 8$^{th}$ to May 18$^{th}$, 2018"),
```

```
  Variable = "NO2_mean",
  val_midpoint = mid_conc_2018
)
# Map of NO2 concentrations in 2019 by municipality
map_19 <-  ARPALdf_Summary_map(
  Data = Data_y %>% filter(lubridate::year(Date) == 2019),
  Title_main = TeX("March 8$^{th}$ to May 18$^{th}$, 2019"),
  Variable = "NO2_mean",
  val_midpoint = mid_conc_2018
)
# Map of NO2 concentrations in 2020 by municipality
map_20 <-  ARPALdf_Summary_map(
  Data = Data_y %>% filter(lubridate::year(Date) == 2020),
  Title_main = TeX("March 8$^{th}$ to May 18$^{th}$, 2020"),
  Variable = "NO2_mean",
  val_midpoint = mid_conc_2018
)
# Map of NO2 concentrations in 2021 by municipality
map_21 <-  ARPALdf_Summary_map(
  Data = Data_y %>% filter(lubridate::year(Date) == 2021),
  Title_main = TeX("March 8$^{th}$ to May 18$^{th}$, 2021"),
  Variable = "NO2_mean",
  val_midpoint = mid_conc_2018
)
# Combining maps
fig_comb <- ggarrange(map_18,map_19,map_20,map_21,
                      ncol = 2, nrow = 2, common.legend = T, legend = "bottom")
fig_comb <- annotate_figure(p = fig_comb,
                            top = text_grob(TeX("Average NO$_{2}$ concentrations by
     municipality during spring time"),
                                            col="blue",
                                            face = "bold",
                                            size = 14))
print(fig_comb)
```

Figure S3 reports the combined maps of NO_2 concentrations. The first noticeable result is that during Spring 2020 the whole region shifted from a yellow-orange color (concentrations well above the 2018 average) to more greenish colors (concentrations below the 2018 average). In particular, one can notice that the alpine belt at north and the Appennines at south-west experienced remarkable improvements. The situation in the highly industrialized and urbanized central belt still remains critical. Indeed, it should be noted that the four main cities in the region (Milan, Monza, Bergamo and Brescia) are connected by an orange stripe, perfectly overlapping with the route of the main highway in Northern Italy, i.e. the A4 Turin-Trieste highway, length around 220 km and travelled in the Lombardy segment by around 153 thousand vehicles per day (AISCAT, 2017). Among others, supportive findings were found by Fioravanti et al (2022), which revealed a statistically significant decrease in NO_2 concentrations across the whole Po Valley during March and April 2020 (corresponding to the hard lockdown phase) as compared to 2019.

Example 3: characterization of meteorological phenomena across Lombardy

In the third example, we use `ARPALData` to retrieve weather data for Lombardy aiming at depicting the main characteristics of local meteorology across Lombardy and to study the heterogeneity due to the morphology of the territory. Also, we show the usage of data quality check functions provided in `ARPALData` to assess the actual

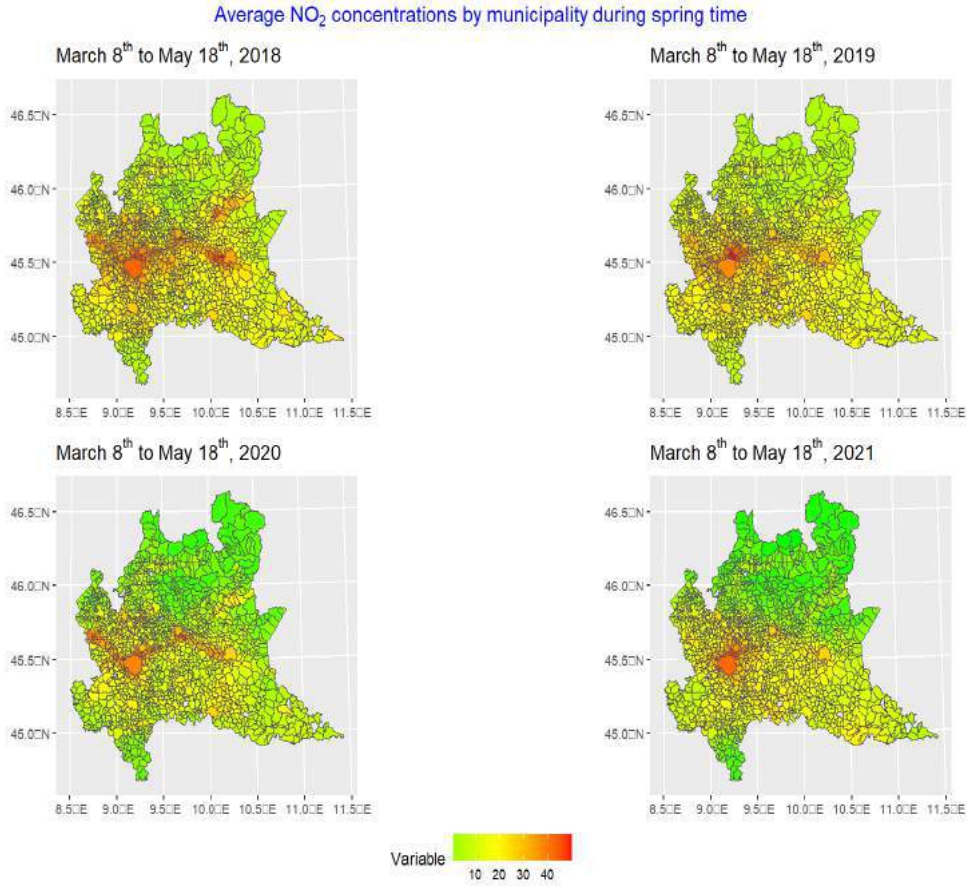

Figure S3 Average NO_2 concentrations observed in 2018, 2019, 2020 and 2021 during the subperiod 8^{th} March to 18^{th} May. Data regarding eleven small municipalities are not available (grey area on the maps).

statistical properties of available data. In particular, as described in the main paper, `ARPALData` contains functions that allow the statistical exploration of a given input dataset, including analytical tool for outlier detection and identification of reccurent patterns in the missing values.

We start by download weather data for the whole ground monitoring network (`ID_station = NULL`) in 2021 (`Date_begin = "2021-01-01"` and `Date_end = "2021-12-31"`) using the parallel setup (`parallel = TRUE`). In particular, we are interested in characterizing monthly (`Frequency = "monthly"`) measurements of cumulated rainfall, average temperature, and average wind speed and direction (`Var_vec = c("Rainfall","Temperature","Wind_speed","Wind_direction")` and `Fns_vec = c("sum","mean","mean","mean")`).

```
### Downloading weather measurements for 2021 at monthly frequency
### Cumulated rainfall, average temperature and average wind speed and direction
```

```
Data_W <-  get_ARPA_Lombardia_W_data(
  ID_station = NULL,
  Date_begin = "2021-01-01",Date_end = "2021-12-31",
  Frequency = "monthly",
  Var_vec = c("Rainfall","Temperature","Wind_speed","Wind_direction"),
  Fns_vec = c("sum","mean","mean","mean"),
  parallel = TRUE
)
```

After downloading the dataset of interest, we employ the `ARPALdf_Summary` function to explore the quality of available data.

```
### Summary statistics and plots for the dataset:
# Descriptive statistics for the whole sample
# Descriptive statistics for each station and for each year
# Gap length (missing data) analysis
# Outlier detection
# Correlation analysis for each station
# Density plots for each variable
summ_data <-  ARPALdf_Summary(
  Data = Data_W,
  by_IDStat = TRUE,
  by_Year = TRUE,
  gap_length = TRUE,
  outlier = TRUE,
  correlation = TRUE,
  density = TRUE,
  histogram = FALSE,
  verbose = FALSE
)

### Analyzing summary statistics: Hampel filter for outlier detection
summ_data$Hampel %>% View()
```

Variable	Lower_bound	Obs_below_low_count	Obs_below_low_perc	Upper_bound	Obs_above_upp_count	Obs_above_upp_perc
Wind_speed	0.00	0.00	0.00	1.42	84.00	3.15
Wind_direction	0.00	0.00	0.00	360.00	0.00	0.00
Temperature	-18.17	3.00	0.11	27.68	0.00	0.00
Rainfall	0.00	0.00	0.00	269.89	51.00	1.91

Table S1 Outlier analysis for each variable (Hampel filter)

The Hampel filter output (Table S1) shows that at aggregate level, all the considered meteorological parameters are mainly affected by slight positive skewness and that potential outliers are abundant. For example, considering wind speed, one can notice that almost 3% of the observations (84 values) are above the upper bound (column `Obs_above_upp_count`), while there are no values below the lower bound. Similar considerations can apply to rainfall. However, we must consider that the upper bound value of the wind speed is just above 1.42 m/s, that is a fairly moderate value. The table also suggests that three potential negative outliers measurements (column `Obs_below_low_count`) can be detected for the average temperature. In other words, there are three months in 2021 at some locations in which the recorded temperatures are particularly lower than the rest of the sample.

```
head(summ_data$Gap_length$Wind_speed,5) %>% View()
```

Also, by inspecting the output of the gap length (missing values) analysis by station (`summ_data$Gap_length`), we can notice that none of the stations presents missing

IDStation	NameStation	Min_gap	q25_gap	Mean_gap	Median_gap	q75_gap	Max_gap	SD_gap
58	Carona Lago Fregabolgia	1.00	1.00	1.00	1.00	1.00	1.00	0.00
60	Villa Di Chiavenna	1.00	1.00	1.00	1.00	1.00	1.00	0.00
100	Milano Lambrate	1.00	1.00	1.00	1.00	1.00	1.00	0.00
102	San Colombano Al Lambro	1.00	1.00	1.00	1.00	1.00	1.00	0.00
106	Varzi Nivione	1.00	1.00	1.00	1.00	1.00	1.00	0.00

Table S2 Gap length (missing data) analysis for wind speed measurements at five stations. Since observations have monthly frequency, gap lengths are measured as months.

values as the gap statistics are always equal to 1 month. In Table S2, we report the gap statistics for five exemplified stations among those available.

As described in the main paper, descriptive statistics at station or municipal levels, as well as time-specific measurements, can be graphically represented through maps using the `ARPALdf_Summary_map` function. In example 2, we used such command to show the average observed values of NO_2 at the municipal level for several years. Similarly, here we want to the average temperature (i.e., `Variable = "Temperature"`) observed at each monitoring station. This can be done by passing to `ARPALdf_Summary_map` as main argument the site-specific descriptive statistics computed through the function `ARPALdf_Summary`. To clearly show how the temperature is strongly affected by seasonality, geography, and local conditions, we represent the average temperature across the whole Lombardy on two non-overlapping sub-periods, that is October to March 2021 (winter) and April to September 2021 (summer).

Starting from the full dataset (`Data_W`), we split the sample into the winter and the summer subperiods (i.e, `Data_spring_summer` and `Data_autumn_winter`). We then compute the summary descriptive statistics by stations for both the two datasets (i.e., `summ_data_spring_summer` and `summ_data_autumn_winter`). Eventually, we plot the average seasonal temperature (`Variable = "Temperature_mean"`) for each station using the function `ARPALdf_Summary_map` by setting the argument `Data = summ_data_spring_summer$Descr_by_IDStat$Mean_by_stat`, where `Descr_by_IDStat$Mean_by_stat` table contained in the output of `ARPALdf_Summary`. To allow comparabiity of the maps, we setup a common reference scale for the colors by imposing the average color being equal to the yearly grand mean across the region. (i.e., `mean_temp <- mean(data$Temperature, na.rm = TRUE)`). The two maps are shown in Figure S4.

```
### Subsampling the observations according to the climatic season:
# spring/summer data (April to September) and autumn/winter data (October to March)
Data_spring_summer <- Data_W %>%
  filter(lubridate::month(Date) %in% c(4,5,6,7,8,9))
Data_autumn_winter <- Data_W %>%
  filter(lubridate::month(Date) %in% c(10,11,12,1,2,3))

### Summary statistics afor each subsample
summ_data_spring_summer <-  ARPALdf_Summary(Data = Data_spring_summer,
                                            gap_length = FALSE,
                                            outlier = FALSE,
                                            correlation = TRUE,
                                            density = FALSE)

summ_data_autumn_winter <-  ARPALdf_Summary(Data = Data_autumn_winter,
                                            gap_length = FALSE,
                                            outlier = FALSE,
                                            correlation = TRUE,
                                            density = FALSE)
```

```
### Plotting average temperature by station and season
# Computing average temperature in 2021 as central reference value
mean_temp <- mean(data$Temperature, na.rm = T)
# Map for spring/summer period
map_T_summer <-  ARPALdf_Summary_map(
                    Data = summ_data_spring_summer$Descr_by_IDStat$Mean_by_stat,
                    Variable = "Temperature",
                    Title_main = "Average temperature in Spring/Summmer 2021",
                    col_scale = c("#FFFF00","#FF9933","#FF0000"),
                    val_midpoint = mean_temp)
# Map for autumn/winter period
map_T_winter <-  ARPALdf_Summary_map(
                    Data = summ_data_autumn_winter$Descr_by_IDStat$Mean_by_stat,
                    Variable = "Temperature",
                    Title_main = "Average temperature in Autumn/Winter 2021",
                    col_scale = c("#FFFF00","#FF9933","#FF0000"),
                    val_midpoint = mean_temp)
# Combining plots
fig_comb <- ggarrange(map_T_summer,map_T_winter,ncol = 2, nrow = 1)
print(fig_comb)
```

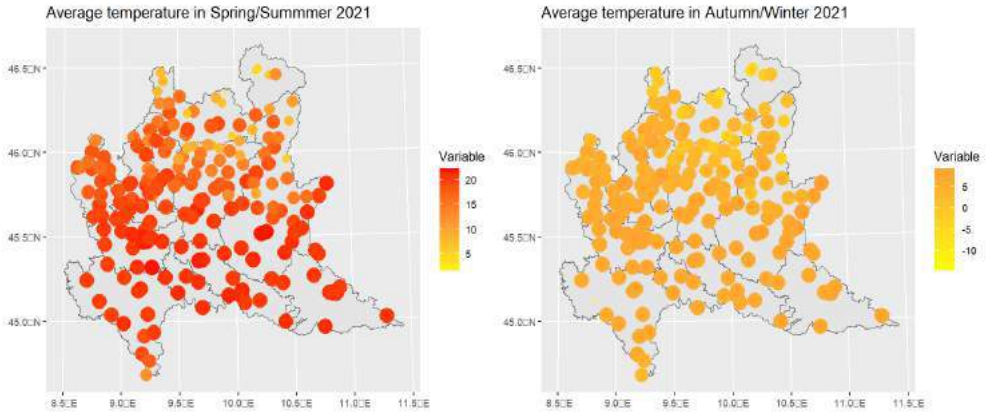

Figure S4 Average temperature observed in the subperiod April to September 2021 for each station (left panel) and average temperature observed in the subperiod October to March 2021 for each station (right panel). The central value of the colors scales is the grand mean temperature observed in 2021.

Observing the left map, it is immediately clear that the average summer temperature recorded in the Alps is significantly lower than temperature in the Po Valley at

south, whereas the thermal excursion may depend on the yearly subperiod. In fact, the summer range is around 30 Celsius degrees (from 0 to 25○), while the winter range is from -10○ to 5○.

Gap length and outlier statistics by station can be well summarized either on a tabular form or on a map. However, linear correlations are suitable for graphical representation and mapping, particularly with the number of stations is large. In particular, we propose to represent the map of correlations between the observed average temperature and rainfall using the coordinates of the stations as points and the average of the correlations as the corresponding value. This plot can be obtain employing the `ARPALdf_Summary_map` filled with the argument `Data = summ_data_spring_summer$Cor_matrix`, where `Cor_matrix` is the linear correlation matrix computed using the `ARPALdf_Summary` function on the subsample. By repeating this operation on both the winter and the summer subsamples, we can compare the dynamics of Pearsons' linear correlation index across climatic seasons. The correlation maps for the two sub-samples are shown in Figure S5.

```
### Plotting linear correlation among maximum temperature and rainfall by station
    and season
# Map for spring/summer period
map_corr1 <-  ARPALdf_Summary_map(
                Data = summ_data_spring_summer$Cor_matrix,
                Variable = "Temperature_max_Rainfall",
                Title_main = "Correlation among max temperature and rainfall in
    Spring/Summmer 2021",
                val_midpoint = 0)

# Map for autumn/winter period
map_corr2 <-  ARPALdf_Summary_map(
                Data = summ_data_autumn_winter$Cor_matrix,
                Variable = "Temperature_max_Rainfall",
                Title_main = "Correlation among max temperature and rainfall in
    Autumn/Winter 2021",
                val_midpoint = 0)

# Combining plots
fig_comb <- ggarrange(map_corr1,map_corr2,ncol = 2, nrow = 1)
```

The arranged maps show how the correlation between rainfall and temperature depends strongly on both morphology and weather season. In particular, the mountainous area (Alps) in the north exhibits strong positive correlations in both summer and winter (dots are red or orange), while the Po Valley at south shows strongly negative correlations (green points). In the former case, the positive correlation could indicate that at higher altitudes the lower temperature is also accompanied by low rainfall during the summer (or that for higher temperatures there is more rainfall). In contrast, the plains are characterized by scarcity of rainfall and high temperatures, especially during summer. The correlations shown in these two maps are in line with the negative monthly rainfall anomalies that characterized the year 2022 in the entire Lombardy region, as well as the trends in the maximum cumulative rainfall values across the 24-hours day. An extensive discussion about rainfall and temperature anomalies registered across Lombardy during the considered period can be found in the report "Climate, natural risks and availability water supply in Lombardy in 2022" (ARPA Lombardia, 2023) produced by ARPA Lombardia. In particular, the

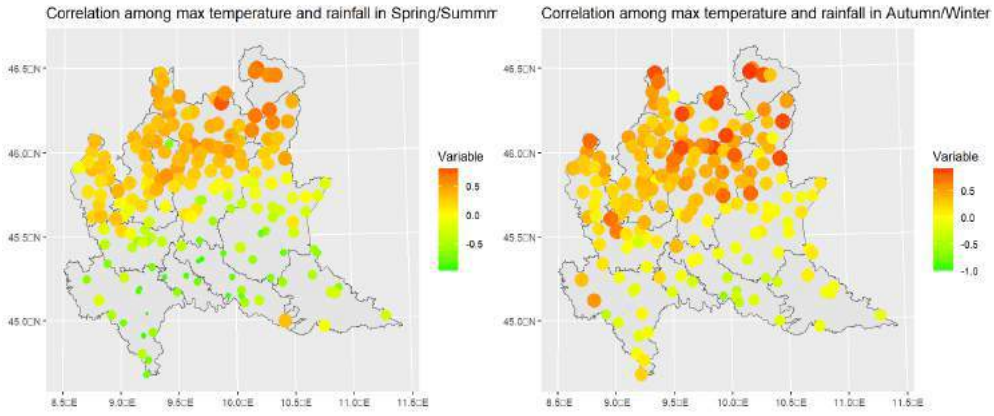

Figure S5 Linear correlation among maximum temperature and rainfall observed in the subperiod April to September 2021 for each station (left panel) and linear correlation among average temperature and rainfall observed in the subperiod October to March 2021 for each station (right panel). Green points are associated with negative values of linear correlations, while orange points are associated with positive values. Yellow points mean uncorrelated data.

report states that the climate in 2022 was classified as "hot-dry" with negative rainfall anomalies distributed for most of the year in the plain areas while the Alpine and pre-Alpine areas enjoyed positive anomalies only in the first autumn months.

Example 4: maps of air quality and weather monitoring networks in Lombardy

In this section, we aim at reproducing the two maps reported in Appendix C1 of the main paper. The maps provide information on either the currently active or historical air quality and ground weather monitoring networks. Notice that the two networks are heterotopic and only in a few cases they do overlap (i.e., few sites monitor both pollution and weather).

We start by downloading the metadata associate with the air quality ground monitoring network through `get_ARPA_Lombardia_AQ_registry`, which does not require any argument. Then, we create a second `data.frame` object containing only the currently active stations. This is done by imposing a filter that select on those stations

with missing stop date (i.e., `is.na(DateStop)`). Eventually, the two AQ networks are mapped through `map_Lombardia_stations` and stored with a graph-specific name.

```
### Air quality network
# Retrieve the full AQ network metadata
aq <-  get_ARPA_Lombardia_AQ_registry()
# Filter currently active stations
aq_red <- aq %>%
  filter(is.na(DateStop))
# Map of the historical network
f1 <-  ARPALData::map_Lombardia_stations(
        data = aq,
        title = "Historical ARPA Lombardia AQ stations (since 1968)"
        )
# Map of the actual network
f2 <-  ARPALData::map_Lombardia_stations(
        data = aq_red,
        title = "Active ARPA Lombardia AQ stations (Dec. 2023)"
        )
```

The operations computed above on the AQ network are now repeated on the weather ground monitoring network. To distinguish the two networks, we adopt a red colored mapping for weather (i.e., `col_points = "red"`).

```
### Weather network
# Retrieve the full W network metadata
w <- get_ARPA_Lombardia_W_registry()
# Filter currently active stations
w_red <- w %>%
  filter(is.na(DateStop))
# Map of the historical network
f3 <-  ARPALData::map_Lombardia_stations(
        data = w,
        title = "Historical ARPA Lombardia W stations (since 1989)",
        col_points = "red"
        )
# Map of the actual network
f4 <-  ARPALData::map_Lombardia_stations(
        data = w_red,
        title = "Active ARPA Lombardia W stations (Dec. 2023)",
        col_points = "red"
        )
```

Eventually, the four plots (i.e., `f1`, `f2`, `f3`, and `f4`) are combined in a single graph with common legend and exported (externally saved) on a .png file. The combined plot is reported in Figure S6.

```
### Combine plots and export
f <- ggarrange(f1,f2,f3,f4, ncol = 2, nrow = 2, common.legend = T)
ggpubr::ggexport(f,width = 1800, height = 1200, res = 150,
        filename = "ARPALData_Stats.png")
```

Example 5: maps of air quality and weather sensors across Lombardy

In this section, we aim at reproducing the two maps reported in Appendix D2 and D3 of the main paper. The maps provide information on the locations of relevant air quality (i.e., NO_2, PM_{10}, $PM_{2.5}$, and ammonia) and weather parameters (i.e., temperature, rainfall, snow depth, and wind) across the Lombardy region. Data refer to locations active on December 2023.

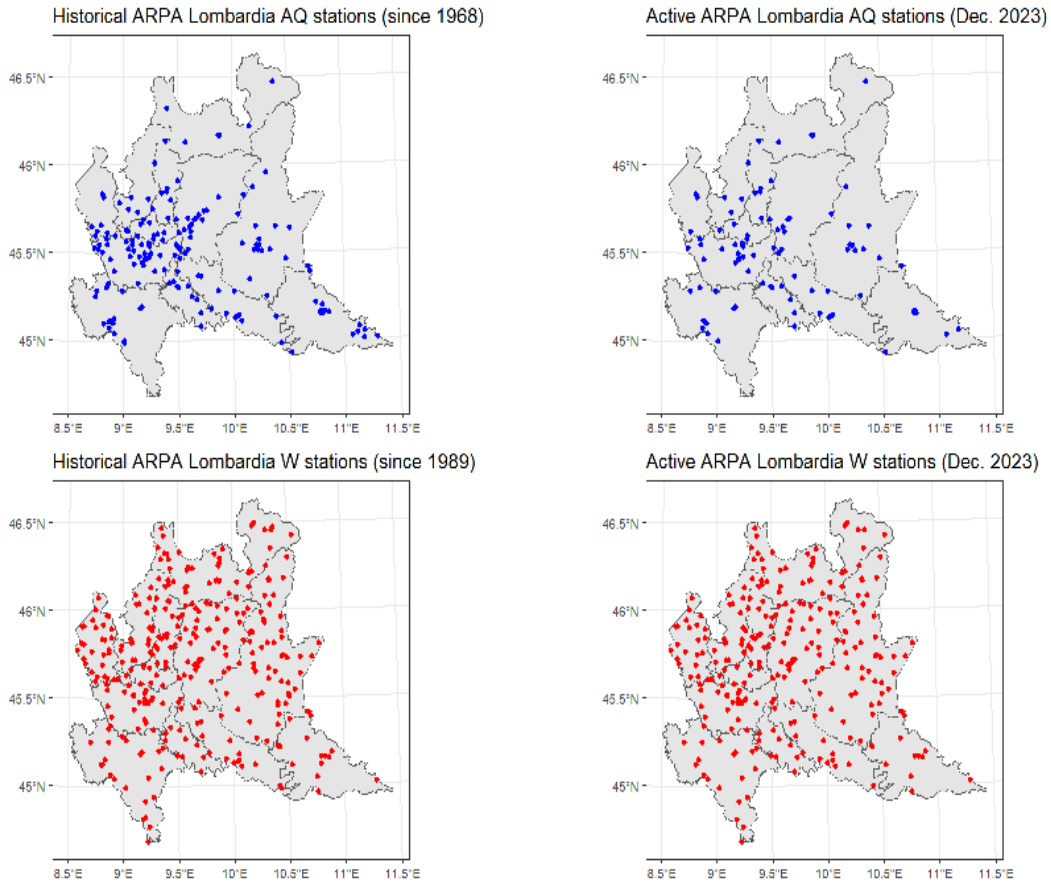

Figure S6 Historical (left panesl) and currently active on December 2023 (right panels) air quality (upper panels) and weather (lower panels) ground monitoring network of ARPA Lombardia.

We start by downloading the metadata associate with the air quality ground monitoring network through `get_ARPA_Lombardia_AQ_registry`, which does not require any argument. Then, we create a second `data.frame` object containing only the currently active stations. This is done by imposing a filter that select on those stations with missing stop date (i.e., `is.na(DateStop)`). Eventually, we keep the sensors associated with each of the considered airborne pollutants and generate the corresponding maps through `map_Lombardia_stations` and stored with a graph-specific name. To distinguish each pollutant, we adopt four colors to map the locations (i.e., `col_points = "color"`). Eventually, the four plots (i.e., `p1`, `p2`, `p3`, and `p4`) are combined in a single graph with common legend and exported (externally saved) on a .png file. The combined plot is reported in Figure S7.

```
### Air quality network
# Retrieve the full AQ network metadata
regAQ <-  get_ARPA_Lombardia_AQ_registry()
```

```
# Filter currently active stations
regAQ_red <- regAQ %>%
  filter(is.na(DateStop))
# Map of the actual networks
p1 <- regAQ_red %>%
  filter(Pollutant == "NO2") %>%
  map_Lombardia_stations(title = "NO2 locations (December 2023)", col_points = "
    blue")
p2 <- regAQ_red %>%
  filter(Pollutant == "PM10") %>%
  map_Lombardia_stations(title = "PM10 locations (December 2023)", col_points = "
    red")
p3 <- regAQ_red %>%
  filter(Pollutant == "PM2.5") %>%
  map_Lombardia_stations(title = "PM2.5 locations (December 2023)", col_points = "
    forestgreen")
p4 <- regAQ_red %>%
  filter(Pollutant == "Ammonia") %>%
  map_Lombardia_stations(title = "Ammonia (NH3) locations (December 2023)",
    col_points = "orange")
# Combining plots
fig_comb <- ggarrange(p1,p2,p3,p4,ncol = 2, nrow = 2)
ggpubr::ggexport(fig_comb,width = 1800, height = 1200, res = 150,
filename = "ARPALData_MapPollutants.png")
```

The operations computed above on the AQ network are now repeated on the weather ground monitoring network. The four plots (i.e., `p5`, `p6`, `p7`, and `p8`) are combined in a single graph with common legend and exported (externally saved) on a .png file. The combined plot is reported in Figure S8.

```
### Weather network
# Retrieve the full W network metadata
regW <- get_ARPA_Lombardia_W_registry()
# Filter currently active stations
regW_red <- regW %>%
  filter(is.na(DateStop))
  # Map of the actual networks
p5 <- regW_red %>%
  filter(Measure == "Temperature") %>%
  map_Lombardia_stations(title = "Temperature locations (December 2023)",
    col_points = "blue")
p6 <- regW_red %>%
  filter(Measure == "Rainfall") %>%
  map_Lombardia_stations(title = "Rainfall locations (December 2023)", col_points =
    "red")
p7 <- regW_red %>%
  filter(Measure == "Snow_height") %>%
  map_Lombardia_stations(title = "Snow depth locations (December 2023)", col_points
    = "forestgreen")
p8 <- regW_red %>%
  filter(Measure == "Wind_direction") %>%
  map_Lombardia_stations(title = "Wind locations (December 2023)", col_points = "
    orange")
# Combining plots
fig_comb2 <- ggarrange(p5,p6,p7,p8,ncol = 2, nrow = 2)
ggpubr::ggexport(fig_comb2,width = 1800, height = 1200, res = 150, filename = "
    ARPALData_MapWeather.png")
```

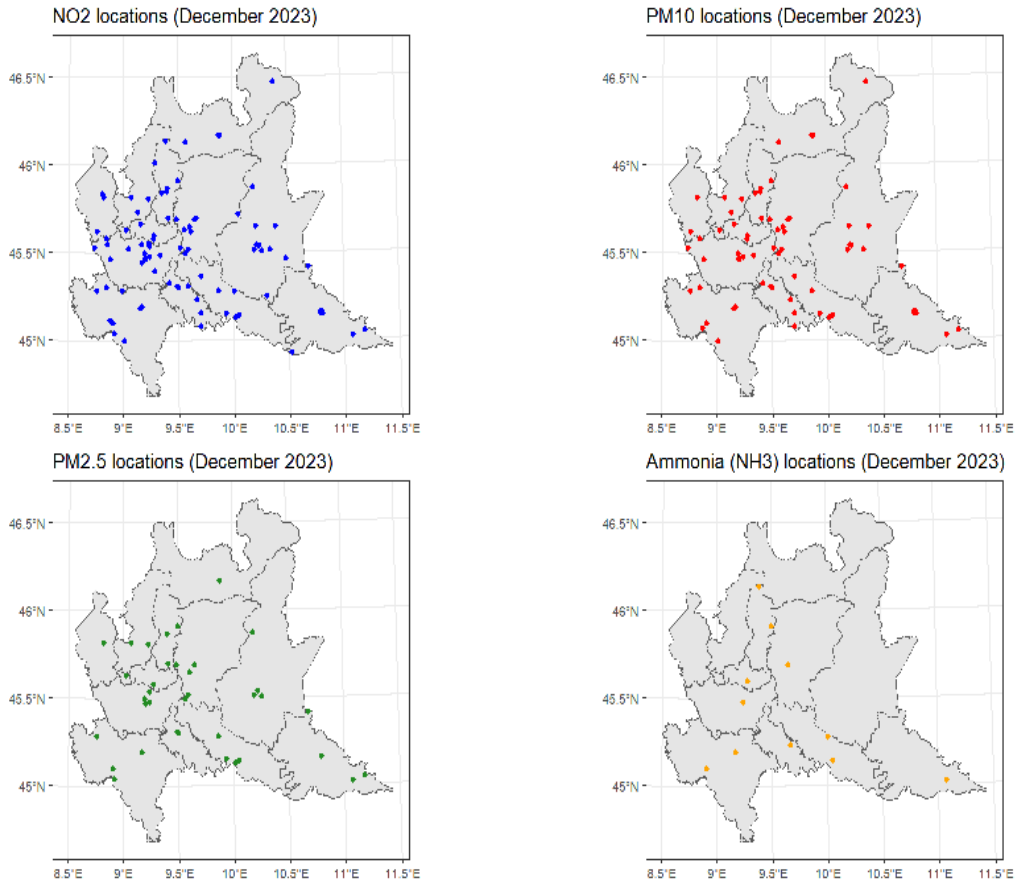

Figure S7 Location of NO_2, PM_{10}, $PM_{2.5}$, and ammonia (NH_3) sensors managed by ARPA Lombardia (sensors active on December 2023)

Example 6: Monte Carlo experiment to assess the computational benefits provided by parallel downloading strategies

In this section, we provide the code to reproduce the Monte Carlo experiment discussed at Section 4.1.1 of the paper. The experiment is designed to assess the computational benefits deriving from the parallel computation compared to the serial strategy. In particular, we compare the computing time required to download air quality data for several combinations of number of stations and number of months to be retrieved.

We start by retrieving the metadata associate with the air quality ground monitoring network through `get_ARPA_Lombardia_AQ_registry`. Then, we filter the dataset in order to preserve only the currently active stations (`is.na(DateStop)`) with installation date before January 1st, 2014 (i.e., `is.na(DateStop)`). This is done by imposing

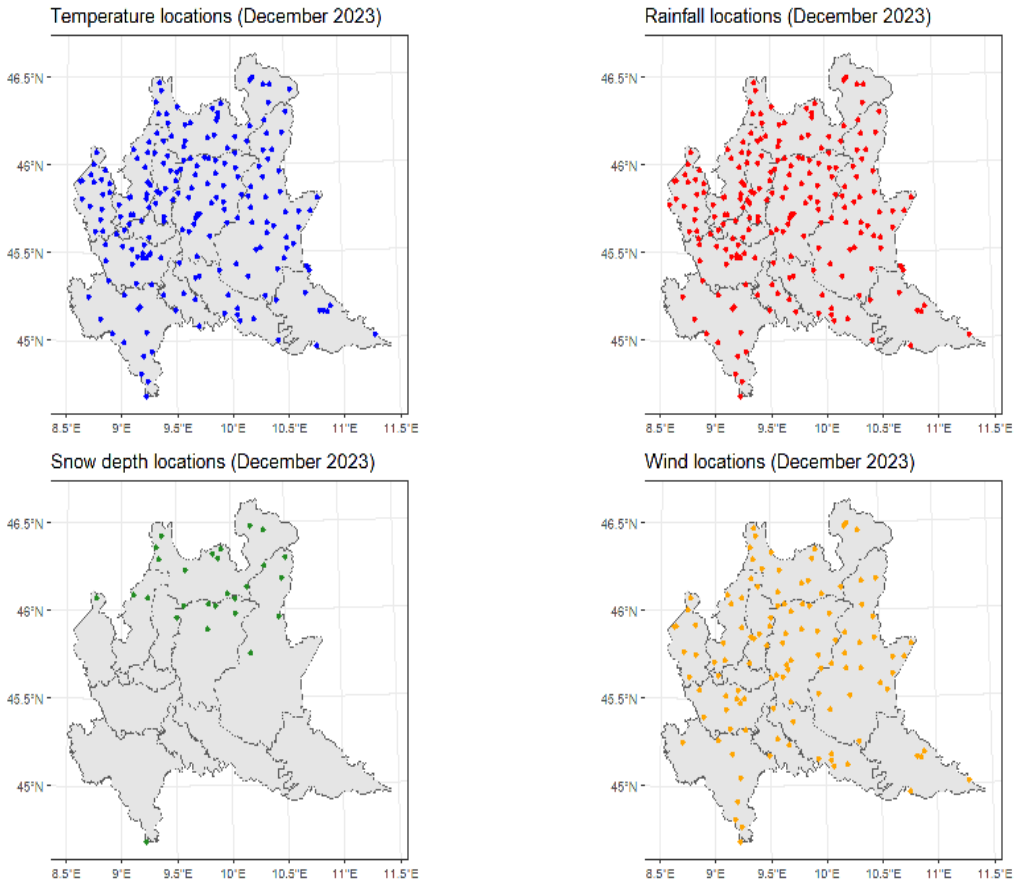

Figure S8 Location of temperature, rainfall, snow depth, and wind speed/direction sensors managed by ARPA Lombardia (sensors active on December 2023)

a filter that select on those stations with missing stop date (i.e., `DateStart <= "2014-01-01"`).

```
##### Define the list of monitoring stations for the experiment: currently active
    and activated before 2014
reg <-  get_ARPA_Lombardia_AQ_registry()
reg_red <- reg %>%
  filter(is.na(DateStop),
       DateStart <= "2014-01-01")
```

Now, we define the sequence of dates among which to sample the starting date of each download experiment. In particular, we consider only monthly dates (always the first day of the month) from January 1st, 2014 to may 1st, 2017 (i.e., the sequence has to be long `length = 40` months from January 2014).

```
##### Define the sequence of dates for the experiment
Date_seq <- character(length = 40)
for (t in 1:length(Date_seq)) {
```

```
  Date_seq[t] <- as.character(as_datetime("2014-01-01") + months(t))
}
```

Then, we define the simulations parameters. For each combination of number of stations (`NumStats`) and number of months (`NumMonths`) to be downloaded, we replicate the download $N = 50$ times both using both parallel and serial settings. In particular, we consider an increasing number of stations,i.e., 10, 20, 50 and all the stations (`NumStats = 0`) and an uneven growing number of months from 1 to 36. Randomness is included in the experiment by randomly selecting the stations to be included in the sample and randomly selecting the period to be downloaded.

```
##### Simulations parameters
iterations <- 50
NumStats <- c(10,20,50,0)
NumMonths <- c(1,3,6,12,36)
```

Eventually, the Monte Carlo experiment is run using a triple for loop (one for the number of stations, one for the number of months, and one for the number of iterations). Notice that, at each iteration, the computing times using serial and parallel strategies are calculated on the same set of data (i.e., a common seed is generated at each iteration) in order to ensure full comparability of the procedures. Computing times of each iteration are store in a `data.frame` object called `res`.

```
##### Stations for loop
for (s in 1:length(NumStats)) {
  ##### Months for loop
  for (m in 1:length(NumMonths)) {
    num_stats_down <- NumStats[s]
    months_down <- NumMonths[m]
    res <- matrix(data = NA, nrow = iterations, ncol = 12)
    colnames(res) <- c("NumStats", "Months", "Iter", "Seed", "Date_begin",
                       "Date_end", "Begin_np", "End_np", "Time_np",
                       "Begin_p", "End_p", "Time_p")
    ##### Iterations for loop
    for (iter in 1:iterations) {
      cat(paste0("NumStats: ",num_stats_down," & NumMonths: ",months_down, " -
    Iteration ",iter))
      seed <- iter + 1000
      set.seed(seed)
      ##### Random sampling station IDs
      if (num_stats_down != 0) {
        stats <- sample(x = reg_red$IDStation, size = num_stats_down, replace = F)
      } else {
        stats <- NULL
      }
      ##### Random sampling date sequence
      Date_begin <- sample(x = Date_seq, size = 1, replace = F)
      Date_end <- as.character(as_datetime(Date_begin) +  months(months_down))
      stats
      Date_begin
      Date_end
      ##### Non parallel download
      b_download_np <- Sys.time()
      d <-  get_ARPA_Lombardia_AQ_data(ID_station = stats,
                                       Date_begin = Date_begin,
                                       Date_end = Date_end,
                                       parallel = FALSE)
      e_download_np <- Sys.time()
      t_download_np <- e_download_np - b_download_np
      ##### Parallel download
      b_download_p <- Sys.time()
      d <-  get_ARPA_Lombardia_AQ_data(ID_station = stats,
                                       Date_begin = Date_begin,
```

```
                                    Date_end = Date_end,
                                    parallel = TRUE)
      e_download_p <- Sys.time()
      t_download_p <- e_download_p - b_download_p

      res[iter,] <- c(num_stats_down, months_down, iter, seed,
                      Date_begin, Date_end,
                      as.character(b_download_np),
                      as.character(e_download_np), t_download_np,
                      as.character(b_download_p),
                      as.character(e_download_p), t_download_p)
    }
    #### Transform to df
    res <- data.frame(res)
    res <- res %>%
      mutate(across(c("NumStats", "Months", "Iter", "Seed",
                "Time_np", "Time_p"), as.numeric),
             across(c("Date_begin", "Date_end","Begin_np", "End_np",
                  "Begin_p", "End_p"), lubridate::as_datetime))
    save(res, file = paste0("SimsRes_NumStats",num_stats_down,
                  "_NumMonths",months_down,".Rdata"))
    gc()
  }
}
```

The next chunk simply allows the user to load in the `R` environment the results from an external `.RData` file.

```
##### Output analysis
NumMonths <- c(1,3,6,12,36)
NumStats <- c(0,10,20,50)
combinations <- expand.grid(NumMonths,NumStats)
ResSims <- vector(mode = "list", length = dim(combinations)[1])
for (i in 1:dim(combinations)[1]) {
  file_name <- paste0("~/R/Sims_timing/SimsRes_NumStats",combinations[i,2],"
    _NumMonths",combinations[i,1],".RData")
  if (file.exists(file_name)) {
    load(file_name)
    ResSims[[i]] <- res
  } else {
    ResSims[[i]] <- NULL
  }
}
ResSims <- bind_rows(ResSims)
```

Eventually, serial and parallel computing time (both averages and standard deviations) are stored into tabular objects.

```
##### Non-parallel: average computing time (seconds)
ResSims %>%
  mutate(Time_np = End_np - Begin_np,
         Time_p = End_p - Begin_p,
         NumStats = factor(NumStats,levels = c("10","20","50","0"), ordered = T),
         Months = factor(Months,levels = c("1","3","6","12","36"), ordered = T))
    %>%
  group_by(NumStats,Months) %>%
  summarise(m_Time_np = mean(Time_np)) %>%
  pivot_wider(names_from = Months, values_from = m_Time_np)

##### Non-parallel: average computing time (seconds)
ResSims %>%
  mutate(Time_np = End_np - Begin_np,
         Time_p = End_p - Begin_p,
         NumStats = factor(NumStats,levels = c("10","20","50","0"), ordered = T),
         Months = factor(Months,levels = c("1","3","6","12","36"), ordered = T))
    %>%
  group_by(NumStats,Months) %>%
```

```
  summarise(s_Time_np = sd(Time_np)) %>%
  pivot_wider(names_from = Months, values_from = s_Time_np)

##### Parallel: average computing time (seconds)
ResSims %>%
  mutate(Time_np = End_np - Begin_np,
         Time_p = End_p - Begin_p,
         NumStats = factor(NumStats,levels = c("10","20","50","0"), ordered = T),
         Months = factor(Months,levels = c("1","3","6","12","36"), ordered = T))
    %>%
  group_by(NumStats,Months) %>%
  summarise(m_Time_p = mean(Time_p)) %>%
  pivot_wider(names_from = Months, values_from = m_Time_p)

##### Parallel: average computing time (seconds)
ResSims %>%
  mutate(Time_np = End_np - Begin_np,
         Time_p = End_p - Begin_p,
         NumStats = factor(NumStats,levels = c("10","20","50","0"), ordered = T),
         Months = factor(Months,levels = c("1","3","6","12","36"), ordered = T))
    %>%
  group_by(NumStats,Months) %>%
  summarise(s_Time_p = sd(Time_p)) %>%
  pivot_wider(names_from = Months, values_from = s_Time_p)
```

Serial		Number of months				
		1	3	6	12	36
Number of stations	10	13.06 (1.73)	20.44 (2.52)	31.56 (3.89)	53.90 (6.00)	146.66 (14.4)
	20	15.22 (1.47)	33.86 (4.34)	55.94 (7.25)	104.82 (13.2)	273.92 (40.7)
	50	30.14 (3.50)	63.06 (6.81)	140.48 (22.2)	210.00 (35.7)	548.30 (81.4)
	All	48.72 (6.04)	130.72 (18.3)	253.18 (37.4)	538.62 (103)	1400.32 (222)

Table S3 Serial computation: average and standard deviation (under parentheses) computing time (in seconds) across the $N = 50$ Monte Carlo replications for several combinations of number of stations and number of months to be retrieved.

Parallel		Number of months				
		1	3	6	12	36
Number of stations	10	27.50 (1.40)	28.72 (0.858)	31.86 (1.43)	36.34 (2.98)	61.34 (4.62)
	20	27.62 (0.725)	31.40 (1.68)	37.70 (2.51)	49.52 (5.37)	100.90 (11.4)
	50	31.22 (1.20)	39.30 (2.82)	59.84 (7.27)	83.70 (11.9)	188.74 (14.2)
	All	37.60 (2.29)	64.30 (6.35)	106.12 (14.4)	202.52 (37.1)	503.20 (75.5)

Table S4 Parallel computation (`parworkers = 8` and `parfuturetype = "multisession"` options): average and standard deviation (under parentheses) computing time (in seconds) across the $N = 50$ Monte Carlo replications for several combinations of number of stations and number of months to be retrieved.

In Table S3, we report the average (and the standard deviation) computing times under serial strategy, whereas in Table S4, we report the timing under parallel strategy. Comparing the two tables we can infer the following: 1) parallel computation drastically reduce the download time when the number of stations considered is large (average times are reduced by 50% to 65% for 50 stations or for the whole network); 2) regardless of the number of stations, for periods of at least one year it is very convenient to use the parallel strategy (especially for a large number of years); 3) when both the number of stations of interest and the period of interest are small, the serial strategy seems preferable; 4) the computation time seems to be more affected by the length of the period of interest than by the number of stations (note, for example, that the average times referred to a single serial month triples when going from 10 to all stations, while the average times fixed the number of stations grows much faster); 5) the variability of the estimated times increases both as the number of stations and the number of months increase (this could be attributed to the need for a longer connection to the databases via the Socrata API, which in turn is dependent on the accesses and the number of users connected at the time of download).

References

AISCAT AISCAeT (2017) Traffico autostradale, veicoli teorici medi giornalieri e veicoli-km - autostrada (2015). Report, AISCAT, Associazione Italiana Società Concessionarie Autostrade e Trafori, URL https://www.asr-lombardia.it//asrlomb/it/14032traffico-autostradale-veicoli-teorici-medi-giornalieri-e-veicoli-km-autostrada?t=Tabella&restrictBy=CCAUTOSTRADE_E_1831249587=Milano-Brescia%7CBrescia-Milano%7CTorino-Milano,CCANNO_63889777=2015

ARPA Lombardia (2023) Clima, rischi naturali e disponibilità idrica in lombardia nel 2022. Report, ARPA Lombardia, URL https://www.arpalombardia.it/media/fd2je4v5/report_arpa_riscus_lombardia_2023.pdf

Fassó A, Rodeschini J, Moro AF, et al (2023) Agrimonia: a dataset on livestock, meteorology and air quality in the lombardy region, italy. Scientific Data 10(1):143. https://doi.org/10.1038/s41597-023-02034-0, URL https://doi.org/10.1038/s41597-023-02034-0

Fioravanti G, Cameletti M, Martino S, et al (2022) A spatiotemporal analysis of no2 concentrations during the italian 2020 covid-19 lockdown. Environmetrics 33(4):e2723. https://doi.org/https://doi.org/10.1002/env.2723, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/env.2723

Marongiu A, Collalto AG, Distefano GG, et al (2023) Application of machine learning to estimate ammonia atmospheric emissions. Preprints https://doi.org/10.20944/preprints202309.0607.v1, URL http://dx.doi.org/10.20944/preprints202309.0607.v1

Meschiari S (2022) latex2exp: Use LaTeX Expressions in Plots. URL https://CRAN.R-project.org/package=latex2exp, r package version 0.9.6

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