Monitoring PM2.5 Pollution In The North End Of Hartford, CT

Jocelyn Phung
jocelyn.phung@uconn.edu

Kristina Wagstrom

Follow this and additional works at: https://opencommons.uconn.edu/srhonors_theses

Part of the Chemical Engineering Commons

Recommended Citation
Phung, Jocelyn and Wagstrom, Kristina, "Monitoring PM2.5 Pollution In The North End Of Hartford, CT" (2023). Honors Scholar Theses. 959.
https://opencommons.uconn.edu/srhonors_theses/959
Monitoring PM$_{2.5}$ Pollution In The North End Of Hartford, CT

Jocelyn Phung, Dr. Kristina Wagstrom
Department of Chemical Engineering
Computational Atmospheric Chemistry and Exposure Lab
University of Connecticut
May 2023
# Table of Contents

Abstract........................................................................................................................................... 3  
**Background.................................................................................................................................... 3**  
**Methodology................................................................................................................................... 6**  
  PurpleAir Monitors: Features, Costs, Networks................................................................. 6  
  Co-location and Initial Testing............................................................................................. 7  
  Site assessment criteria.......................................................................................................... 9  
  Community engagement......................................................................................................... 9  
  Deployment of sensors............................................................................................................ 10  
  Data collection....................................................................................................................... 11  
  Data Processing: Daily Averages.......................................................................................... 12  
  Data Processing: AQI............................................................................................................. 13  
**Results, Analysis and Discussion.............................................................................................. 14**  
  Colocation Results.............................................................................................................. 14  
  Current network in North End............................................................................................. 20  
  Hartford Data......................................................................................................................... 21  
**Future Work................................................................................................................................ 26**  
  Community Engagement - Lessons Learned................................................................. 26  
  Communicating Findings - Quarterly Reports..................................................................... 27  
**Conclusion.................................................................................................................................... 27**  
**Acknowledgments.................................................................................................................... 28**  
**References.................................................................................................................................. 28**  
**Appendix.................................................................................................................................... 30**  
  Appendix 1: Removing_Duplicates_and_Data_Completeness.mlx......................................... 30  
  Appendix 2: Colocating_Plot.mlx.......................................................................................... 32  
  Appendix 3: Monitor_Comparison.mlx.................................................................................. 35  
  Appendix 4:Sensor_Comparison1.mlx.................................................................................. 39  
  Appendix 5: preprocess_purpleair.py................................................................................. 41  
  Appendix 6: PmtoAQI.m........................................................................................................ 46
Abstract

Particulate matter ($PM$) or particle pollution is one of the six criteria air pollutants that can cause harm to human health and the environment; yet, there is a lack of data in many areas of the United States. Particulate matter is a mixture of solid particles and liquid droplets suspended in the air (EPA). Exposure to $PM_{2.5}$ has been linked to respiratory and cardiovascular conditions.

People who live in urban areas are more likely to be exposed to particulate matter as many urban areas are known to have poor air quality. Our goal is to determine how particulate matter levels in the North End of Hartford, CT impact residents’ pollution exposure using a network of low-cost sensors. First, the team developed programming in MATLAB to process data, which included outlier methodology, eliminating duplicate data and plotting hourly or daily averages. Next, we reached out to community centers and concerned citizens to discuss installing monitors on their property. We then accessed the site, deployed PurpleAir PA-II-SD monitors across North Hartford, CT and collected $PM_{2.5}$ concentrations data over the course of several months. To communicate our findings to the community, we issue quarterly reports on their $PM_{2.5}$ pollution exposure. Equipped with concrete data, residents of North Hartford can be informed about their air and be empowered to advocate for improvements in air quality in the area.

Background

Even for high-income countries, there is a 50% chance that the air exceeds World Health Organization guidelines for air pollution (UNEP). Good health and well-being is one of the 17 goals of the United Nations Sustainable Development Goals. In October 2021, the UN Human
Rights Council declared access to a clean, healthy and sustainable environment as a human right. (UN News). In July 2022, the UN General Assembly recognized this right as well. Quantifying air pollution exposure is crucial to fulfilling the right to a healthy and sustainable environment.

Particulate matter (PM) or particle pollution is one of the six criteria air pollutants that can cause harm to human health and the environment (Clean Air Act). It is a mixture of solid particles and liquid droplets suspended in the air (EPA). PM is categorized by the size of the particle. PM$_{10}$, PM$_{2.5}$ and PM$_{1}$ have aerodynamic diameters of 10 micrometers or less, 2.5 micrometers or less and 1 micrometers or less respectively (Masoud et al., 2021). PM spans a variety of chemical composition and emission sources, for instance black carbon and oil smoke. PM$_{10}$ consists of dust, pollen or mold (Masoud et al., 2021). PM 2.5, also known as fine particle pollution, encompasses dust, fly ash, oil smoke and $H_2SO_4$ mist (Lipfert, 2020). PM can be directly emitted from a source, including construction sites, unpaved roads and fires, or formed in the environment from complex reactions of chemicals (EPA). Sulfur dioxide and nitrogen oxides emitted from power plants, automobiles and industrial operations can react in the atmosphere and form PM (EPA).

PM has negative effects on both the environment and people. In the environment, PM in the atmosphere coupled with stagnant meteorological conditions leads to haze in urban areas, which results in reduced visibility and clarity (Masoud et al., 2021). When PM settles in the environment, it can make lakes or streams acidic, deplete nutrients in soils, impact nutrient balance in coastal waters and large river basins, damage sensitive forests or crops, affect ecosystem diversity and contribute to acid rain effects (EPA, Masoud et al., 2021). Because PM...
is inhalable and can enter human lungs and bloodstream, exposure to $PM_{2.5}$ has an array of adverse effects on human health. $PM_{2.5}$ exposure alone is the fifth risk factor for global mortality and is linked to 230,000 deaths in adults per year (Jbaily et al., 2022, EPA). $PM_{2.5}$ exposure is also correlated with premature death in people with heart or lung disease, nonfatal heart attacks, irregular heartbeats, chronic obstructive pulmonary disease, aggravated asthma, decline in lung function and increased respiratory symptoms (Masoud et al., 2021, Awokola et al., 2020).

Evidence of air pollution is essential in increasing public awareness and policy advocacy to reduce adverse health effects (Awokola et al., 2020). In 2023, the Environmental Protection Agency (EPA) strengthened its health-based standard of $PM_{2.5}$ to between 9 to 10 $\mu g/m^3$ annually. Data is needed to assess whether or not communities are meeting the standard. Air pollution data is historically collected using expensive regulatory equipment. Because of their cost, a small number of the monitors are deployed. However, due to the spatial variability of air pollution, one or two regulatory monitors for an entire city is not enough to capture localized air pollution and identify emission sources. Fortunately, low-cost air quality monitors can be utilized by citizen scientists to complement regulatory evidence of air pollution and monitor air pollution at higher spatial resolutions.

Intuitively, residents of urban areas are at increased risk of air pollution exposure. The Clean Air Act was established in 1970 by the US Congress, with major revisions made in 1977 and 1990 (EPA). The impetus for passing the Clean Air Act was dense, visible smog in many cities and industrial areas across the country (EPA). Although the U.S. has made significant progress in regards to air pollution, it is still an issue in overburdened communities like Hartford, CT. Not
only is Hartford, CT an urban area, it is also a predominately Black and Hispanic city, with 36.4% of residents identifying as Black or African American and 45.5% identifying as Hispanic or Latino (U.S. Census Bureau). The median household income in Hartford from 2017-2021 is around $37,000 (U.S. Census Bureau). It is well documented that communities of color are at increased risk to air pollution exposure due to environmental injustice (Masoud et al., 2021; Jbaily et al., 2022). Children, older adults and low-income communities are considered vulnerable populations as well (EPA, 2023; Jbaily et al., 2022). The North End of Hartford is federally designated as a Promise Zone to centralize urban renewal efforts, advance neighborhood revitalization and build a prosperous future for the community (City of Hartford). One of the North Hartford Promise Zone work groups focuses on health and aims to “improve the emotional and physical development of high-risk children and families” (City of Hartford). Thus, our research focuses on monitoring $PM_{2.5}$ pollution in North Hartford specifically.

**Methodology**

*PurpleAir Monitors: Features, Costs, Networks*

For this project, we used PurpleAir Classic Air Quality Monitors (PurpleAir PA-II SD). This model measures real-time $PM_{2.5}$ concentrations. The data is stored in the cloud and transmitted to the real-time PurpleAir Map when there is WiFi connectivity. This allows for public access to $PM_{2.5}$ data and the air quality index (AQI). When the monitor is offline, data is stored in the SD card. The monitors are intended for both outdoor and indoor use. They also require very little power to operate (0.18A continuous).
Inside the monitors, there are two Plantower PMS-5003 laser particle counters referred to as sensors. They can measure particulate matter ranging from 0.3 to 10 um. The two sensors take measurements at alternate 10s intervals, and provide a two-minute average as raw data (Barkjohn et al., 2021). The dual sensors allow us to validate the sensor’s readings to ensure proper monitor functioning. PurpleAir PA-II SD monitors also include pressure, temperature and humidity sensors (BOSCH BME280). The relative humidity data is used in this project to apply EPA’s correction factor.

PurpleAir PA-II SD monitors are considered low-cost monitors and retail for $229.00. Though cost is a great advantage for widespread community air quality monitoring, the accuracy of air pollution data is not up to standard. PurpleAir monitors have high precision but low accuracy (Barkjohn et al., 2021). The correction factor developed by Karoline Barkjohn, Brett Gantt and Andrea Clements solves this problem, and the formula is used by the EPA and AirNow.gov. The correction factor accounts for relative humidity, the parameter that affects concentration readings most. Corrected PM2.5 values are computed using Equation (1):

\[
PM_{2.5} = 0.524 \times PA_{cf_1} - 0.0862 \times RH + 5.75, \tag{1}
\]

where \( PA_{cf_1} \) is the average of the two sensors from the higher correction factor (cf_1), and RH is relative humidity in percentage.

Co-location and Initial Testing
Before the PurpleAir monitors were deployed in the North End of Hartford, CT, we co-located the monitors to ensure proper functioning. All of the monitors were installed in pairs at the Spring Valley Student Farm using plant stakes and zip-ties. After the monitors had been secured, we connected them to power. For consistency, all of the monitors were installed at approximately the same height. The monitors were deployed at the farm for several weeks.
After the co-location period, the data collected at the farm was accessed through PurpleAir’s now archived sensor data download tool. The first step of data pre-processing is removing duplicates. At the beginning of our project, we discovered that the data files from PurpleAir sometimes contained duplicates of the same data. Two simple double for-loops were used to identify duplicated data points and remove them from the dataset.

The next steps of data pre-processing include converting time zones, omitting data collected before midnight, and evaluating data completeness. The default time zone is UTC, which is 5 hours ahead of the local time EST. Data collected before midnight was omitted for plotting daily averages. Lastly, data completeness was calculated. Using the start and end time of the dataset, the program calculated how much time has passed and how many data entries are expected. The yield was then calculated. This is done using the Removing_Duplicates_and_Data_Completeness.mlx code (Appendix 1).

We used time series plots and one-to-one monitor plots to access the performance of co-located PurpleAir monitors. The time series plots generate the average hourly concentration vs time for two co-located monitors (Appendix 2). They allow us to visualize the $PM_{2.5}$ concentration data and identify issues like time lagging. One-to-one monitor plots are used to easily identify monitors that recorded different values than the other monitor (Appendix 3). We plotted the hourly concentrations of monitor 1 and 2 on the x-axis and y-axis respectively. We expect to see data points lined up at $y = x$ if the monitors are functioning properly.
For monitors that are not functioning as expected based on the time series plots and one-to-one monitor plots, we used one-to-one sensor plots to further investigate the issue (Appendix 4). We plotted hourly concentrations from sensor A and sensor B on the x-axis and y-axis respectively. Again, the data points should line up at \( y = x \) if both sensors are working properly.

**Site assessment criteria**

Prior to community engagement efforts, we developed a site assessment protocol. Initially, there were four technical must-haves for a site to be considered: existence of an outdoor outlet, consistent wifi, being away from obstructions (e.g. bushes, shrubs), and a place to zip-tie or screw the monitor onto. Later in the project, the requirement to have consistent wifi was relaxed because SD card data can be accessed. This increased the number of sites that met our requirements. In addition to the technical requirements, there are two physical criteria. The first was allowing free air flow by installing the monitor 1-2 m above ground and ideally 1 m away from the wall or building surface. The second was avoiding direct sources, including direct exhausts, bus stations, parking garages and major highways, in order to study background levels of particulate matter pollution.

**Community engagement**

After we have ensured that our PurpleAir monitors are ready for deployment, we reached out to community members in the North End of Hartford, CT. First, the team compiled a list of community centers we can work with and gathered their contact information. We drafted an
email template including brief information about the health impacts of $PM_{2.5}$ pollution, increased risk in urban areas and what is required of a monitor host. We also used this email template as a guide when we call community centers. Our outreach strategies include emailing, phone calls, visiting in person and distributing flyers.

*Deployment of sensors*

Once we determine a site for data collection, we install the PurpleAir monitors using zip ties. The most important factor is having an outdoor outlet. The team searches for places to zip tie the monitor to while still within reach of an outlet. We have installed monitors on porches, balconies, fences and handrails. We also ensure that the installation location is away from any direct sources. For example, installing a monitor directly next to a busy parking lot of a community center is not ideal. **Figure 1** shows one of our deployed monitors in the North End.
Figure 1: A PurpleAir monitor installed on the handrails of a residential home.

Data collection

Data collected by the PurpleAir monitors were accessed using PurpleAir’s data download tool, API or SD cards. PurpleAir monitors collect $PM_{1}, PM_{2.5}, PM_{10}$ concentrations, as well as temperature, humidity, pressure and particle counts. Measurements are taken every two minutes. For the purposes of this study, $PM_{2.5}$ real-time concentrations and humidity were used.

Accessing data using the API is a tedious process for historical data because only two days of
data can be downloaded at a time. Using the API also involves some coding and obtaining read and write API keys. SD card data is stored by day, so it is time consuming to compile all the data files into one if we are interested in a longer time period. The data download tool is the simplest from a user standpoint. We requested read and write API keys from PurpleAir. Then, we input the start and end time, the sensor index, and the fields that we are interested in ($PM_{2.5}$ cf.1 for a and b, relative humidity).

**Data Processing: Daily Averages**

Data processing to compute daily averaged $PM_{2.5}$ concentration is achieved by the preprocess_purpleair.py code written by Britney Russell (Appendix 5). Data preprocessing includes removing duplicates, converting to local time zones, inserting placeholders for missing values and calculating data completeness. The methodology is the same as the one used for colocated data.

Data processing to plot $PM_{2.5}$ concentration involves averaging between concentration data from sensors A and B, applying a correction factor, calculating daily averages, and plotting. If both sensors A and B recorded values of zero, the value is deemed as a true zero. If one of the sensors reported a value of zero, the non-zero value is used.

If both sensors report concentration values below 15 $\mu g/m^3$, the absolute difference is calculated. If the absolute difference between two sensors is below 3 $\mu g/m^3$, we take the
average. If the absolute difference is greater than 3 \( \mu g/m^3 \), the data point is considered an outlier and is removed from the dataset.

If both sensors report concentration values above 15 \( \mu g/m^3 \), the percentage difference is calculated. The threshold is a 20% percentage difference. If the percentage difference is below 20%, the average is calculated. If not, the data point is treated as an outlier and is removed.

After averaging between sensors, we apply the correction factor developed by Barkjohn et al. Then, if there is more than 50% data within a day, we take the daily average. If not, the data point is omitted. The code outputs an excel file with a date column as well as daily averaged concentration columns for each location.

**Data Processing: AQI**

AQI is calculated with the PmtoAQI.m code written by Erica Hindle (Appendix 6). We follow the equation used by regulatory agencies, where the air quality index \( I_p \) is calculated by Equation (2):

\[
I_p = \frac{I_{Hi} - I_{Lo}}{BP_{Hi} - BP_{Lo}} \left( C_p - BP_{Lo} \right) + I_{Lo},
\]

where \( C_p \) is the truncated concentration of pollutant \( p \), \( BP_{Hi} \) is the concentration breakpoint that is greater than or equal to \( C_p \), \( BP_{Lo} \) is the concentration breakpoint that is less than or equal to \( C_p \), \( I_{Hi} \) is the AQI value corresponding to \( BP_{Hi} \), and \( I_{Lo} \) is the AQI value corresponding to \( BP_{Lo} \). The AQI breakpoints are ranges given by AirNow.gov.
Results, Analysis and Discussion

Colocation Results

Hourly averaged time series plots are produced for two colocated monitors. If the monitors are functioning properly, we expect to see similar hourly concentrations. Figure 2 shows $PM_{2.5}$ hourly averaged concentrations for monitor AC86 and 8135 for a five-day period. The trends and concentrations from the two monitors align for the most part. Monitor AC86 recorded slightly higher readings for the peak from 50-60 hours.
**Figure 2:** Time series plot of two colocated monitors, AC86 and 8135, at the Spring Valley Student Farm from September 12 to September 16, 2022. This figure was generated using Colocating_Plot.mlx.

Additionally, the team produces one-to-one monitor plots as another way to validate that monitors are functioning properly. **Figure 3** shows concentration values from monitor AC86 plotted on the y-axis and values from monitor 8135 plotted on the x-axis. We expect the data
points to be in line with $y = x$. The data points that diverge slightly from the $y = x$ line are the same data points where monitor AC86 recorded higher values in the figure above.

Figure 3: One-to-one plot of two colocated monitors, AC86 and 8135, at the Spring Valley Student Farm from September 12 to September 16, 2022. This figure was generated using Monitor_Comparison.mlx.

One-to-one monitor plots are useful to identify monitors that are not functioning. In Figure 4, all of the data points congregate on the y-axis, which represents concentration values for monitor
758A. This means that monitor E3E3 either recorded values of zero or outliers that were removed by the algorithm throughout the entire time period.

**Figure 4:** One-to-one plot of two collocated monitors, 758A and E3E3, at the Spring Valley Student Farm from October 8 to October 18, 2022. This figure was generated using Monitor_Comparison.mlx.

In cases where the time series plots or the one-to-one monitor plots indicate an issue, we use the one-to-one sensor plots to determine if an individual sensor is malfunctioning. **Figure 5** below shows a one-to-one sensor plot of sensor A of monitor E3E3 on the y-axis and sensor B on the
x-axis. All of the data points were congregated on the x-axis, which reveals that sensor A is recording zeros or outliers.

**Figure 5**: One-to-one plot of sensors A and B of E3E3 at the Spring Valley Student Farm from October 8 to October 18, 2022. This figure was generated using Sensor_Comparison1.mlx.

Plotting sensor A and B separately against time in **Figure 6** and **Figure 7** shows that sensor B is functioning properly. We determined that sensor A is broken since it recording concentration values as high as almost 6,000 $\mu g/m^3$. The team sprayed compressed air into the sensor A channel, which resolved the issue.
Figure 6: Time series plot of sensor A of monitor E3E3 at the Spring Valley Student Farm from October 8 to October 18, 2022. This figure was generated using Comparing_Sensors.mltx.
Figure 7: Time series plot of sensor B of monitor E3E3 at the Spring Valley Student Farm from October 8 to October 18, 2022. This figure was generated using Comparing_Sensors.mlx.

Current network in North End

The map in Figure 8 shows our current network of PurpleAir monitors in the North End of Hartford, CT. We have four monitors deployed at the Park House of Keney Park Sustainability Project, a residential home at Sunset Street, Habitat for Humanity and Hartford City Mission.
Our goal is to deploy at least 11 monitors that are dispersed throughout the North End to better monitor spatial distribution of $PM_{2.5}$.

*Figure 8*: Map of the North End of Hartford, CT. The red dotted line indicates the perimeter of the region. The red dots are locations with a PurpleAir monitor deployed.

**Hartford Data**

The monitor at Sunset Street has been deployed since May 5, 2022. The monitor at Sunset Street is located close to the freeway. Between May 6 2022 and April 26, 2023, $PM_{2.5}$ concentrations
ranged from 5.78 to 25.49 μg/m³. From Figure 9 below, we can see that most of the values fall within the 5 to 20 range, with some higher peaks every so often.

![Figure 9: Time series plot of daily averaged PM$_{2.5}$ concentrations at Sunset Street from May 6, 2022 to April 26, 2023. The daily averages were calculated using preprocess_purpleair.py and the figure was generated in Excel.](image)

Converting PM$_{2.5}$ concentration to AQI puts our results into context. 0-50 AQI is considered satisfactory air quality, where air pollution poses little to no risk to human health. 51-100 AQI translates to acceptable air quality, but there may be a risk particularly for people who are unusually sensitive to air pollution; the level of concern is moderate. The AQI at Sunset Street from May 6 2022 to April 26, 2023 ranged from 24 to 80, as shown in Figure 10. This falls within the good to moderate range. The air quality at this location can pose health risks to
vulnerable populations, like children, older adults, people with heart or lung disease, and people of low socioeconomic status.

**Figure 10**: Time series plot of daily averaged AQI at Sunset Street from May 6, 2022 to April 26, 2023. The AQI values were calculated and plotted using PmtoAQI.m.

Monitors were deployed at the Park House and Habitat for Humanity on October 28, 2022. Park House is relatively close to the freeway, as shown in the map in **Figure 8**, but is a part of Keney
Park. A monitor was deployed at Hartford City Mission on December 18, 2022. Hartford City Mission is located in a residential area.

Daily averaged $PM_{2.5}$ concentrations are similar among the four monitors in the North End. As we can see from Figure 11, all four monitors peaked on the same days as well. $PM_{2.5}$ concentrations ranged from 5.87 to $26.7 \mu g/m^3$. The monitors at Park House, Habitat for Humanity and Hartford City Mission had missing data for about 15 days in March 2023. We suspect that there were wifi connectivity issues or power issues during this time.

Figure 11: Time series plot of daily averaged $PM_{2.5}$ concentrations at Habitat for Humanity, Hartford City Mission, Park House and Sunset Street from October 28, 2022 to April 26, 2023. The daily averages were calculated using preprocess_purpleair.py and the figure was generated in Excel.
The AQI for the monitors at Habitat for Humanity, Hartford City Mission, Park House and Sunset Street are shown below in Figure 12. The AQI ranged from 25-81, which falls in the good to moderate range. Similar to the AQI range at Sunset Street, the air quality can pose health risks to vulnerable populations.

**Figure 12**: Time series plot of daily averaged AQI at Habitat for Humanity, Hartford City Mission, Park House and Sunset Street from October 28, 2022 to April 26, 2023. The AQI values were calculated and plotted using PM_to_AQI.m.
Our findings show that there is room for improvement for air quality in the North End of Hartford, CT to protect residents and workers in the area. Ideally, AQI should be under 50 year-round to ensure protect residents from air pollution. Deployed monitors should be kept to collect more data. Once we have a full year’s data, we might be able to identify seasonal differences in $PM_{2.5}$.

**Future Work**

**Community Engagement - Lessons Learned**

For this project, the team utilized different community engagement methods and found some to be more successful than others. For starters, attempting to gather interest for air quality research was more challenging because of the COVID-19 pandemic. During our first two semesters of community outreach (Fall 2021- Spring 2022), our modes of communication were email and phone calls. Community centers are usually already busy and/or understaffed, and most did not reply to our outreach emails. We also suspect that our emails might have gone to spam. Phone calls were slightly more successful, but most of our voicemails were not returned. Some community centers asked for information to be emailed when we chatted over the phone, but did not respond to further information.

Visiting sites in person and speaking to staff members were the most effective community outreach strategy. After explaining that we are a team of undergraduate students working with the community to study air pollution exposure, community members were usually willing to help
host a monitor and/or spread the word about the project. In some instances, community members have expressed personal concerns about their air pollution concerns, like aggravated asthma. In person visits were limited during Fall 2021- Spring 2022 as most places were still following strict COVID-19 protocols, but are recommended for future work.

**Communicating Findings - Quarterly Reports**

After collecting and analyzing air pollution data from the North End, it is important that we communicate our findings to the community. In the future, teams should issue quarterly reports with weekly or monthly concentration averages of $PM_{2.5}$, as well as their respective AQI. It might be useful to compare a site to the regional average, so that residents can understand how their air quality is compared to the rest of the North End. This semester, my team developed a template for the quarterly reports, but did not get to generate them with our latest data.

**Conclusion**

Particulate matter pollution can be dangerous to human health. Monitoring particulate matter levels is important to understand air quality concerns in communities. Low-cost monitors coupled with a correction factor is a way to increase community monitoring of air pollution. Future work should focus on expanding the monitoring network in the North End of Hartford, and communicating our results with the community.
Acknowledgments

I would like to express my sincere gratitude to Dr. Kristina Wagstrom, the principal investigator of the Hartford Air Pollution Monitoring Project and my faculty mentor for this honors thesis. I would also like to thank Britney Russell, who is a graduate student in Chemical Engineering, for her continued guidance and contributions to this project. Additionally, thank you to my undergraduate peers – Erica Hindle, Mackenzie Sonnenberg, Christine Troller, Margaret Ducasse and Natalia Holovyn – for working alongside me and contributing to the work outlined in this paper. Thank you to the Spring Valley Student Farm for allowing us to use their space to conduct preliminary testing on our monitors. Lastly, my deepest gratitude to our community partners in Hartford, including the Keney Park Sustainability Project, Hartford Habitat for Humanity, Hartford City Mission and Hartford resident Helen Nixon.

References


**Appendix**

*Appendix 1: Removing_Duplicates_and_Data_Completeness.mlx*

```matlab
%Import data from excel (change file name accordingly)
data_A = xlsread("TEST_A.xlsx");

%Part 1: Remove duplicated data
k = length(data_A);
r = 0; d = 0;

%For loop 1: identify the number of duplicated data points
%freezes one row (outer loop) and iterates over the rest of the excel
%sheet (inner loop) to check for duplicates
for i = 1:(k-1)
    for j = 1:k
        if i ~= j
            if (data_A(i,1) == data_A(j, 1)) && (data_A(i,2) == data_A(j,2))
                %date and time of two different rows are the same
                r = r+1;
            end
        end
    end
end
r = r/2; %number of duplicates
```
%For loop 2: remove duplicated rows
%Iterates from 1 to the length of the excel file minus the number of
duplicates as duplicated rows are being deleted
for i = 1:(k-r-1)
    for j = 1:(k-r)
        if i ~= j

            if (data_A(i,1) == data_A(j, 1)) && (data_A(i,2) == data_A(j,2))

                data_A(j,:) = [];

            end

        end

    end
end

%Part 2: Convert UTC to EST. EST is 5 hours behind = +0.208333335
%Rewrite data table such that the first entry is after 00:00
k = length(data_A);
for i = 1:k
    data_A(i,2) = data_A(i,2) + 0.208333335;
    if data_A(i,2) > 1
        data_A(i,1) = data_A(i,1) + 1;
        data_A(i,2) = data_A(i,2) -1;
    end
end

%Part 3: Omit data before 00:00 so we start on a new day (0.002 for UTC, 0.21 for EST)
t=1;
while data_A(t,2) > 0.21
    t=t+1;
end
dataA = data_A(t:end,:);

%Part 3: Check for data completeness
start = datetime(2022,9,16,19,17,30); %enter start date and time (YYYY,M,DD,HOUR,MIN,SEC)
end1 = datetime(2023,1,30,15,51,51); %enter end date and time
time_passed = end1-start;
days_passed = days(time_passed) %should be x-axis when plotting
mins = minutes(time_passed)
expected_data = mins/2 %expected number of data points
%Calculate yield
numdataA = length(dataA) % actual number of data points
yield = 100*(numdataA/expected_data) %in percentage
%Export data table to new excel sheet(change file name accordingly)
writematrix(dataA, "TEST_A1")
Appendix 2: Colocating_Plot.mlx

%This code plots the hourly average of two monitors that are colocated.
data1A = xlsread("monitorAC86_A.xlsx");
data1B = xlsread("monitorAC86_B.xlsx");
data2A = xlsread("monitor8135_A.xlsx");
data2B = xlsread("monitor8135_B.xlsx");
%data3A = xlsread("E3E3_A.xlsx");
%data3B = xlsread("E3E3_B.xlsx");
[x1 y1] = average(data1A,data1B); %Monitor 1

Z = 4.5216
O = 1.5422
dc = 100

[x2 y2] = average(data2A,data2B); %Monitor 2

Z = 8.3945
O = 4.0923
dc = 100

figure
plot(x1,y1,x2,y2)
title('SVF PM 2.5 Concentrations: Sep 12,2022 - Sep 16,2022')
subtitle('AC86 & 8135')
xlabel('Time (hour)')
ylabel('PM2.5 Concentration (ug/m3)')
legend('Monitor AC86','Monitor 8135')
function [size1 c] = average(dataA, dataB)
    f = min(length(dataA), length(dataB));
    difference = []; 
    difference = dataA(1:f,2) - dataB(1:f,2); % time of sensor A - time of sensor B
    g = 0; a = 0; b = 0; % establish zero values before for loop
    % Time tolerance: 45 seconds = 0.00052
    for i = 1:(f-1)
        if abs(difference(i)) < 0.00052
            dataA(i, 2) = dataB(i, 2);
            g = g+1; % number of times the times matched
        elseif difference(i) >= 0.00052 && difference(i+1) >= 0.00052 % sensor A time > sensor B time = sensor A skipped a measurement
            a = a+1; % number of times sensor A skipped a reading
            dataA(end+1,:) = 0; % extends sensor A matrix by 1 row
            for e = 1:length(dataA)-i
                dataA(end-e+1,:) = dataA(end-e,:); % shift rows i:end of sensor A down 1 row
            end
        end
    dataA(i,2) = dataB(i,2); % set the sensor A time equal to time of sensor B
    dataA(1,5:7) = 0; % set PM values to 0
    f = min(length(dataA), length(dataB)); % recalculate f
    difference = dataA(1:f,2) - dataB(1:f,2); % recalculate difference between sensors
elseif difference(i) <= -0.00052 && difference(i+1) <= -0.00052 %sensor B skipped a measurement
b = b+1; %number of times sensor A skipped a reading
dataB(end+1,:) = 0; %extends sensor B matrix by 1 row
for e = 1:(length(dataB)-i)
dataB(end-e,:) = dataB(end-e,:); %shift rows i:end of sensor B down 1 row
end
dataB(i,2) = dataA(i,2); %set the sensor B time equal to time of sensor A
dataB(i,5:7) = 0; %set PM values to 0
f = min(length(dataA),length(dataB)); %recalculate f
difference = dataA(1:f,2) - dataB(1:f,2); %recalculate difference between sensors
end
end

% Assign the date and time columns in separate tables
dA = dataA(:,1); % date column in Sensor A
dB = dataB(:,1); % date column in Sensor B
tA = dataA(:,2); % time column in Sensor A
tB = dataB(:,2); % time column in Sensor B
APM25 = dataA(:,6); % PM 2.5 measurement from Sensor A
BPM25 = dataB(:,6); % PM 2.5 measurement from Sensor B
T = [dA tA]; % combine date and time columns of Sensor A
S = [dB tB]; % combine date and time columns of Sensor B
[~,index_T,index_S] = intersect(T,S,'rows'); % Match date and time columns with intersect
C = APM25(index_T); % Create a column with the matched times in Sensor A
D = BPM25(index_S); % Create a column with the matched times in Sensor B
p = length(index_T); % Calculate length of matrix to use in for loop
% establish zero values before for loop
k = 0; h = 0;m = 0;o = 0;z = 0;
for i = 1:p
if C(i) == 0 && D(i) == 0 %Both sensors had a value of 0
j(i) = 0;
z = z+1;
end
b(i) = abs(C(i) - D(i)); % calculate the absolute value of the difference between Sensor A and B measurements
if b(i) < 3 % if the difference between Sensor A and B is less than 3 ug/cm3, then take the average of the two
j(i) = (C(i)+D(i))/2;
end
if C(i) == 0 && D(i) ~= 0 % if Sensor A is equal to 0 and Sensor B is not, use Sensor B’s measurement
j(i) = D(i);
k = k+1;
end
if D(i) == 0 && C(i) ~= 0 % Same as above if statement but switched sensors
j(i) = C(i);
h = h+1;
end
if b(i)>3 \ \% \ \text{if the difference is above 3 \text{ug/cm}^3, then replace the value with}
\text{-10 (this will be omitted in the next for loop)}
j(i) = -10;
o = o+1;
end
end
Z = 100*z/length(j) \ \% \ \text{Percentage of zeros}
O = 100*o/length(j) \ \% \ \text{Percentage of outliers}
dc = 100*length(j)/f \ \% \ \text{Data completeness after time matching between sensors}
jj = [1:1:length(j)]; \ \% \ \text{establish matrix for x axis that is the length of j}
\text{%average the x-axis and calculate size to use for x-axis on graph}
S = numel(jj); \ \% \ \# \ \text{of elements in jj}
xx = reshape(jj(1:S-mod(S,30)),30,[]);
x_sum = sum(xx,1); \ \% \ \text{returns row vector with sum of each column}
%m = x_sum.'/720; \ \% \ \text{trailing rows are ignored to match data set sizing}
%sizx = size(m);
sizex1 = [1:1:length(x_sum)]'; \ \% \ \text{final x axis}
\text{%average the y-axis}
H = numel(j); \ \% \ \# \ \text{of elements in j}
ff = reshape(j(1:H-mod(H,30)),30,[]);
sum_y = sum(ff.*(ff>0),1); \ \% \ \text{sum each column but omit outliers: -10 values are}
\text{replaced by 0}
num_readings = 30-sum(ff.*(ff<0),1)/(-10); \ \% \ \text{subtract the numbers of outliers}
\text{omitted from 30}
\text{%Omit hourly average if there are less than 15 data points}
\text{%For future work: Interpolate from nearest neighbors to replace averages}
\text{%where there are not enough data points}
\text{for i=1:length(num_readings)}
if num_readings(i)<15
sum_y(i) = 0; \ \% \ \text{For now, the values set are simply placeholders}
num_readings(i)=1;
end
end
c = (sum_y./num_readings)'; \ \% \ \text{final y axis: hourly average}
end

\textbf{Appendix 3: Monitor\textunderscore Comparison.mlx.}

\% \ \text{This code plots two monitors that are co-located against each other.}
data1A = xlsread("E3E3_A.xlsx");
data1B = xlsread("E3E3_B.xlsx");
data2A = xlsread("758A_A.xlsx");
data2B = xlsread("758A_B.xlsx");
[x1 y1] = average(data1A, data1B); %Monitor 1

Z = 0
O = 99.9115
dc = 98.7512

[x2 y2] = average(data2A, data2B); %Monitor 2

Z = 1.3291
O = 1.8147
dc = 98.8629

% Establish vector to plot x=y
m = max(max(y1), max(y2)); % Maximum value of the sensors
xaxis = [1:m];
yaxis = [1:m];
% Match length of y1 and y2 for scatter plot
len = min(length(y1), length(y2));
y1 = y1(1:len);
y2 = y2(1:len);
% Plot comparing Sensors A & B
figure
scatter(y1, y2) % Scatter plot of PM 2.5 values from Sensors A & B
hold on
plot(xaxis, yaxis) % Plot x=y
hold off
xlim([0 inf])
ylim([0 inf])
title('SVF PM 2.5 Concentrations: Oct 8, 2022 - Oct 18, 2022')
xlabel('E3E3')
ylabel('758A')
function [size1 c] = average(dataA,dataB)
    f = min(length(dataA),length(dataB));
    difference = [ ];
    difference = dataA(1:f,2) - dataB(1:f,2); % time of sensor A - time of sensor B
    g = 0; a = 0; b = 0; % establish zero values before for loop
    % Time tolerance: 45 seconds = 0.00052
    for i = 1:(f-1)
        if abs(difference(i)) < 0.00052
            dataA(i, 2) = dataB(i, 2);
            g = g+1; % number of times the times matched
        elseif difference(i) >= 0.00052 && difference(i+1) >= 0.00052 % sensor A time > sensor
            a = a+1; % number of times sensor A skipped a reading
            dataA(end+1,:) = 0; % extends sensor A matrix by 1 row
            for e = 1:(length(dataA)-i)
                dataA(end-e+1,:) = dataA(end-e,:); % shift rows i:end of sensor A down 1 row
            end
            dataA(i,2) = dataB(i,2); % set the sensor A time equal to time of sensor B
        end
    dataA(i,5:7) = 0; % set PM values to 0
    f = min(length(dataA),length(dataB)); % recalculate f
    difference = dataA(1:f,2) - dataB(1:f,2); % recalculate difference between sensors
```matlab
elseif difference(i) <= -0.00052 && difference(i+1) <= -0.00052 % sensor B skipped a reading
    b = b+1; % number of times sensor A skipped a reading
dataB(end+1,:) = 0; % extends sensor B matrix by 1 row
for e = 1:(length(dataB)-i)
dataB(end-e+1,:) = dataB(end-e,:); % shift rows i:end of sensor B down 1 row
end
dataB(i,2) = dataA(i,2); % set the sensor B time equal to time of sensor A
dataB(i,5:7) = 0; % set PM values to 0
f = min(length(dataA),length(dataB)); % recalculate f
difference = dataA(1:f,2) - dataB(1:f,2); % recalculate difference between sensors A and B
end
end

% function average(dataA, dataB)
% Assign the date and time columns in separate tables
dA = dataA(:,1); % date column in Sensor A
dB = dataB(:,1); % date column in Sensor B
tA = dataA(:,2); % time column in Sensor A
tB = dataB(:,2); % time column in Sensor B
APM25 = dataA(:,6); % PM 2.5 measurement from Sensor A
BPM25 = dataB(:,6); % PM 2.5 measurement from Sensor B
T = [dA tA]; % combine date and time columns of Sensor A
S = [dB tB]; % combine date and time columns of Sensor B
[~, index_T, index_S] = intersect(T,S,'rows'); % Match date and time columns with intersect
C = APM25(index_T); % Create a column with the matched times in Sensor A
D = BPM25(index_S); % Create a column with the matched times in Sensor B
p = length(index_T); % Calculate length of matrix to use in for loop
% establish zero values before for loop
k = 0; h = 0; m = 0; o = 0; z = 0;
for i = 1:p
    if C(i) == 0 && D(i) == 0 % Both sensors had a value of 0
        j(i) = 0;
        z = z+1;
    end
    b(i) = abs(C(i) - D(i)); % calculate the absolute value of the difference between Sensor A and B
    if b(i) < 3 % if the difference between Sensor A and B is less than 3 ug/cm3, then take the average
        j(i) = (C(i)+D(i))/2;
    end
    if C(i) == 0 && D(i) ~= 0 % if Sensor A is equal to 0 and Sensor B is not, use Sensor B’s measurement
        j(i) = D(i);
        k = k+1;
    end
    if D(i) == 0 && C(i) ~= 0 % Same as above if statement but switched sensors
        j(i) = C(i);
    end
end
```

h = h+1;
end
if b(i)>3  % if the difference is above 3 ug/cm3, then replace the value with
         -10 (this will be omitted in the next for loop)
j(i) = -10;
o = o+1;
end
Z = 100*z/length(j)  %Percentage of zeros
O = 100*o/length(j)  %Percentage of outliers
dc = 100*length(j)/f  %Data completeness after time matching between sensors
jj = [1:1:length(j)];  % establish matrix for x axis that is the length of j
%average the x-axis and calculate size to use for x-axis on graph
S = numel(jj);  % # of elements in jj
xx = reshape(jj(1:S-mod(S,30)),30,[]);
x_sum = sum(xx,1);  %returns row vector with sum of each column
%m = x_sum.'/720;  % trailing rows are ignored to match data set sizing
%sizx = size(m);
sizex1 = [1:1:length(x_sum)'];  % final x axis
%average the y-axis
H = numel(j);  % # of elements in j
ff = reshape(j(1:H-mod(H,30)),30,[]);
sum_y = sum(ff.*(ff>0),1);  % sum each column but omit outliers: -10 values are
                          %replaced by 0
num_readings = 30-sum(ff.*(ff<0),1)/(-10);  %subtract the numbers of outliers
                          %omitted from 30
%Omit hourly average if there are less than 15 data points
%For future work: Interpolate from nearest neighbors to replace averages
%where there are not enough data points
for i=1:length(num_readings)
    if num_readings(i)<15
        sum_y(i) = 0;  %For now, the values set are simply placeholders
        num_readings(i)=1;
    end
end
c = (sum_y./num_readings)';  % final y axis: hourly average
end

Appendix 4:Sensor_Comparison1.mlx
%This code compares data between Sensors A and B in a monitor
%Plots Sensor A on x-axis and Sensor B on y-axis
%When running the code, change Lines 6-7 and 105-108 accordingly
%Import data from Excel
dataA = xlsread("E3E3_A");
dataB = xlsread("E3E3_B");
f = min(length(dataA),length(dataB));  %Minimum length between the 2 sensors
difference = [];
difference = dataA(1:f,2) - dataB(1:f,2);  %time of sensor A - time of
sensor B

g = 0; a = 0; b = 0; % establish zero values before for loop

% Time tolerance: 45 seconds = 0.00052

for i = 1:(f-1)
    if abs(difference(i)) < 0.00052
        dataA(i, 2) = dataB(i, 2); % Set timestamp equal to each other
        g = g+1; % number of times the times matched
    elseif difference(i) >= 0.00052 && difference(i+1) >= 0.00052 % sensor A time > sensor B time = sensor A skipped a measurement
        a = a+1; % number of times sensor A skipped a reading
        dataA(end+1,:) = 0; % extends sensor A matrix by 1 row
        for e = 1:(length(dataA)-i)
            dataA(end-e+1,:) = dataA(end-e,:); % shift rows i:end of sensor A down 1 row
        end
    end
    dataA(i,2) = dataB(i,2); % set the sensor A time equal to time of sensor B
    dataA(1:5:7) = 0; % set PM values to 0
    f = min(length(dataA),length(dataB)); % recalculate f
    difference = dataA(1:f,2) - dataB(1:f,2); % recalculate difference between sensors A and B
    elseif difference(i) <= -0.00052 && difference(i+1) <= -0.00052 % sensor B skipped a measurement
        b = b+1; % number of times sensor A skipped a reading
        dataB(end+1,:) = 0; % extends sensor B matrix by 1 row
        for e = 1:(length(dataB)-i)
            dataB(end-e+1,:) = dataB(end-e,:); % shift rows i:end of sensor B down 1 row
        end
    end
    dataB(i,2) = dataA(i,2); % set the sensor B time equal to time of sensor A
    dataB(1:5:7) = 0; % set PM values to 0
    f = min(length(dataA),length(dataB)); % recalculate f
    difference = dataA(1:f,2) - dataB(1:f,2); % recalculate difference between sensors A and B
end

times_matched = g/f*100

% Assign the date and time columns in separate tables

x = dataA(:,1); % date column in Sensor A
y = dataB(:,1); % date column in Sensor B
z = dataA(:,2); % time column in Sensor A
w = dataB(:,2); % time column in Sensor B

APM25 = dataA(:,6); % PM 2.5 measurement from Sensor
BPM25 = dataB(:,6); % PM 2.5 measurement from Sensor

% Exclude PM 2.5 values greater than 200 ug/cm^3

eA = 0; eB = 0;
for i=1:length(APM25)
    if APM25(i)>200
        APM25(i)=-10;
        eA = eA + 1;
    end
end
for i=1:length(BPM25)
  if BPM25(i)>200
    BPM25(i)=-10;
    eB = eB + 1;
  end
end
%Show how many points we removed
excludedA = eA/length(APM25)*100 %Percentage of erroneous data
excludedB = eB/length(BPM25)*100
T = [x z]; % combine date and time columns of Sensor A
S = [y w]; % combine date and time columns of Sensor B
[~,index_T,index_S] = intersect(T,S,'rows'); % Match date and time columns with intersect function and record index of matches
C = APM25(index_T); % Create a column with the matched times in Sensor A
D = BPM25(index_S); % Create a column with the matched times in Sensor B
%Establish vector to plot x=y
m = max(max(C),max(D)); %Maximum value of the sensors
xaxis = [1:m];
yaxis = [1:m];
%Plot comparing Sensors A & B
figure
scatter(C,D) %Scatter plot of PM 2.5 values from Sensors A & B
hold on
plot(xaxis,yaxis) %Plot x=y
hold off
xlim([0 inf]) %Exclude erroneous values that were set to -10
ylim([0 inf])
title('SVF PM 2.5 Concentrations: Oct 8 to Oct 18, 2022')
subtitle('E3E3')
ylabel('Sensor A')
xlabel('Sensor B')
Published with MATLAB® R2020b

Appendix 5: preprocess_purpleair.py

import pandas as pd
import numpy as np
# import matplotlib.pyplot as plt
import os

# FILEPATHS
*******************************************************************
csv_inputs = "./dat/csv" # path to csv input files
out = "./out" # path to output directory

# FUNCTIONS
*******************************************************************
# function that creates a new pm2.5 column using conditions
def calculate_pm2_5(row):
    a = row['pm2.5_cf_1_a']
    b = row['pm2.5_cf_1_b']

    # use the zero value if both sensors report a value of zero
    if a == 0 and b == 0:
        return 0

    # use the nonzero value if one of the sensors report a value of zero
    elif a == 0 or b == 0:
        return max(a, b)

    # if both sensors report values lower than 15, check the absolute difference.
    # abs diff < 3 ug m-3
    elif a < 15 and b < 15:
        avg = 0
        if abs(a-b) < 3:
            avg = (a + b) / 2
        return avg if avg > 0 else np.nan

    # if both sensors report values greater than 15, check the percent difference.
    # percent diff < 20%
    elif a > 15 and b > 15:
        avg = 0
        if abs(a-b) / (a + b) / 2 < 0.2:
            avg = (a + b) / 2
        return avg if avg > 0 else np.nan

    # return NaN as a default
    else:
        return np.nan

# PROCESSING
******************************************************************
# create an empty dataframe
averages = pd.DataFrame()

# create an empty list to hold monitor locations
locations = []

for file in os.listdir(csv_inputs):
    location = file.split('-')[0]
    location = location[0 : len(location) - 5]
locations.append(location)
print(f"processing monitoring data for {location}\")

... load data columns of the csv file and load the data as strings ...
df = pd.read_csv(csv_inputs + "/" + file, dtype = "str")

... change the datatype of specific columns ...
# change PM data columns and humidity to float
df["pm2.5_cf_1_a"] = df["pm2.5_cf_1_a"].astype(float)
df["pm2.5_cf_1_b"] = df["pm2.5_cf_1_b"].astype(float)
df["humidity"] = df["humidity"].astype(float)

... add a datetime column to hold datetime objects in the local timezone and update the date and time columns ...
# create the datetime column
df["datetime"] = pd.to_datetime(df["date"].astype(float), unit = "d", origin = "1900-01-01") + pd.to_timedelta(df["time"].astype(float), unit = "d")

# round the minutes down to remove second increments
df["datetime"] = df["datetime"].round('min')

# make the datetime stamps timezone aware
df["datetime"] =
df["datetime"].dt.tz_localize('UTC').dt.tz_convert('EST')

# check if minute is odd and round down to even increment
df.loc[df["datetime"].dt.minute % 2 == 1, 'datetime'] -= pd.Timedelta(minutes = 1)

# update the date and time column with just the timezone adjusted dates
df["date"] = df["datetime"].dt.date
df["time"] = df["datetime"].dt.time

... drop timestamps with duplicate values, if any ...
# get a count of all records before duplicates are dropped
oldrec_count = len(df)

# drop duplicates
df.drop_duplicates(subset = "datetime", keep = "first", inplace = True)
# print statement
if len(df) != oldrec_count:
    print(f"{oldrec_count - len(df)} records dropped. reason: duplicates")

# reset index
df = df.reset_index(drop = True)

...  
add missing records for each date
...
# set the datetime column as the index
df.set_index("datetime", inplace = True)

# add the missing records and reset the index
df = df.resample('2 min').asfreq().reset_index()

...  
check for data completeness in the raw data per day and change the PM
data columns to
NaN if the number records for the day are less than the defined
threshold
...
threshold = 0.5
counter = 0 # counter used to count the number days with the number of
records being less than the threshold

for date, rec in df.groupby("date"):
    # check if the number of records for the day is less than the
defined threshold
    # if so, change all PM records associated with the date to NaN
    if len(rec) / 720 < threshold:
        print(f"date {date} has less than {threshold * 100}%
missing values")
        df.loc[df["date"] == date, ["pm2.5_cf_1_a", "pm2.5_cf_1_b"]] = np.nan

    counter = counter + 1 # update counter

...  
combine the data from both sensors into a single PM2.5 column and drop
sensor data columns
...
# combine the data from both sensors
df["pm2.5"] = df.apply(calculate_pm2_5, axis = 1)

# drop columns that are not needed
df.drop(columns = ["pm2.5_cf_1_a", "pm2.5_cf_1_b"], inplace = True)
check for data completeness for the entire dataset

# get all non NaN pm2.5 records
allnonNaN_records = df.loc[~df['pm2.5'].isna()]
completeness_score = ((len(allnonNaN_records)) / (len(df) - counter * 720)) * 100
print(f"data completeness percentage for the dataset: 
{completeness_score}%.

# correct pm2.5 values
# correction found in the article:
# Barkjohn KK, Gantt B, Clements AL.
# Development and Application of a United States wide correction for
# PM2.5 data collected with the PurpleAir sensor.

df["pm_corrected"] = 0.524 * df["pm2.5"] - 0.0862 * df["humidity"] / 100 + 5.75

calculate daily averages of the corrected values

# remove rows with NaN values in the "pm2.5" column
df_dropna = df.dropna(subset = ["pm_corrected"])

daily_avg = df_dropna.groupby("date")["pm_corrected"]).mean().reset_index()
	daily_avg.columns = ["date", f"pm2.5_{location.lower().replace(' ', '_')}"
	daily_avg.columns = ["date", location]

add to master dataframe

if os.listdir(csv_inputs).index(file) == 0:
    averages = pd.concat([averages, daily_avg], ignore_index = True)
else:
    averages = pd.merge(averages, daily_avg, on = "date", how =
"outer")

# EXPORT DATAFRAME TO CSV
averages.fillna(-10).to_csv(out + "/" + "daily_averages.csv", index = False)

# PLOTTING
# # set the 'date' column as the index
# averages.set_index("date", inplace = True)
# # plot the PM columns vs the date
# fig, ax = plt.subplots(figsize=(20, 8))
# averages.plot(ax = ax, y = averages.columns.tolist(), color=['red', 'green', 'blue'])
# plt.xlabel('Date')
# plt.ylabel('PM2.5 (ug/m3)')
# plt.show()

Appendix 6: PmtoAQI.m

datatable = xlsread('daily_averages-2.xlsx');
date = datatable(:,1);
PM1 = datatable(:,2);
PM2 = datatable(:,3);
PM3 = datatable(:,4);
PM4 = datatable(:,5);
AQI1 = PM_to_AQI(PM1);
AQI2 = PM_to_AQI(PM2);
AQI3 = PM_to_AQI(PM3);
AQI4 = PM_to_AQI(PM4);
xaxis= datetime(date, "ConvertFrom", "Excel");
AQI_table = [xaxis,AQI1',AQI2',AQI3',AQI4'];
xlswrite("Hartford_AQI_xlsx",AQI_table)
figure
plot(xaxis, AQI4)
title('Sunset Street AQI')
subtitle('May 6, 2022 to April 26, 2023')
xlabel('Date')
ylabel('AQI')
figure
plot(xaxis(176:end), AQI1(176:end), xaxis(176:end), AQI2(176:end),
    xaxis(176:end), AQI3(176:end), xaxis(176:end), AQI4(176:end))
title('North End Monitors AQI')
subtitle('October 28, 2022 to April 26, 2023')
xlabel('Date')
ylabel('AQI')
legend('Habitat For Humanity','Hartford City Mission','Park
function out = PM_to_AQI(PM)
for i = 1:length(PM)
    if PM(i) < 12
        Bh(i) = 12;
        Bl(i) = 0;
        Ih(i) = 50;
        Il(i) = 0;
    elseif PM(i) < 35.4
        Bh(i) = 35.4;
        Bl(i) = 12.1;
        Ih(i) = 100;
        Il(i) = 51;
    elseif PM(i) < 55.4
        Bh(i) = 55.4;
        Bl(i) = 35.5;
        Ih(i) = 150;
        Il(i) = 101;
    elseif PM(i) < 150.4
        Bh(i) = 150.4;
        Bl(i) = 55.5;
        Ih(i) = 200;
        Il(i) = 151;
    elseif PM(i) < 250.4
        Bh(i) = 250.4;
        Bl(i) = 150.5;
        Ih(i) = 300;
        Il(i) = 201;
    elseif PM(i) < 350.4
        Bh(i) = 350.4;
        Bl(i) = 250.5;
        Ih(i) = 400;
        Il(i) = 301;
    else
        Bh(i) = 500.4;
        Bl(i) = 350.5;
        Ih(i) = 500;
        Il(i) = 401;
    end
    AQI(i) = (Ih(i)-Il(i))/(Bh(i)-Bl(i))*(PM(i)-Bl(i)) + Il(i);
end
Data_new(i) = AQI(i);
end
out = Data_new;
end
Published with MATLAB® R2020b