Environmental Modelling and Software 134 (2020) 104832

Contents lists available at ScienceDirect

Environmental Modelling and Software

journal homepage: http://www.elsevier.com/locate/envsoft

The AirSensor open-source R-package and DataViewer web application for interpreting community data collected by low-cost sensor networks

Brandon Feenstra [a,b,c,*], Ashley Collier-Oxandale [a], Vasileios Papapostolou [a], David Cocker [b,c], Andrea Polidori [a]

[a] South Coast Air Quality Management District, Air Quality Sensor Performance Evaluation Center (AQ-SPEC), Diamond Bar, CA, 91765, USA
[b] University of California - Riverside, Department of Chemical & Environmental Engineering, Riverside, CA, 92521, USA
[c] University of California - Riverside, Bourns College of Engineering, Center for Environmental Research and Technology (CE-CERT), Riverside, CA, 92521, USA

ARTICLE INFO

Keywords:
Community air monitoring
Citizen scientist
Low-cost air quality sensor
Open-source R package
Particulate matter PM2.5
Data interpretation

ABSTRACT

While large-scale low-cost sensor networks are now recording air pollutant concentrations at finer spatial and temporal scales than previously measured, the large environmental data sets generated by these sensor networks can become overwhelming when considering the scientific skills required to analyze the data and generate interpretable results. This paper summarizes the development of an open-source R package (*AirSensor)* and interactive web application (*DataViewer*) designed to address the environmental data science challenges of visualizing and understanding local air quality conditions with community networks of low-cost air quality sensors. *AirSensor* allows users to access historical data, add spatial metadata, and create maps and plots for viewing community monitoring data. The *DataViewer* application was developed to incorporate the functionality and plotting functions of the R package into a user-friendly web experience that would serve as the primary source for data communication for community-based organizations and citizen scientists.

Software availability

The *AirSensor* R-package version 0.5 was developed by Mazama Science and South Coast AQMD. *AirSensor* is Free and Open Source Software available through the GitHub repository [https://github.com/MazamaScience/AirSensor/tree/version-0.5]. Mazama Science maintains the package as part of its ongoing relationships with federal, state and local air quality agencies. AirSensor version 0.5 was first released in 2019 under General Public License v3.0 (GPL-3.0) and runs on Windows, Unix, and Macintosh operating systems. AirSensor was written in R and program files are less than 5 Mbytes. AirSensor is designed to be used with R ($\geq$ 3.3) and RStudio.

The *DataViewer* Shiny application was developed by Mazama Science and South Coast AQMD. *DataViewer* is Free and Open Source Software available through the GitHub repository [https://github.com/MazamaScience/AirSensorShiny]. The DataViewer was first released in 2019 under General Public License v3.0 (GPL-3.0) and runs on Windows, Unix, and Macintosh operating systems. DataViewer was written in R and program files are less than 7 Mbytes. The DataViewer requires Git, Apache, Docker, R, and R Shiny Server.

1. Introduction

A paradigm shift in air quality monitoring is occurring with citizen scientists able to develop hyper-local community monitoring networks to supplement the established regulatory monitoring networks that are designed for regional monitoring (Snyder et al., 2013). These environmental monitoring networks are increasing in complexity, size, and resolution (both spatial and temporal) due to technological advances and cost reductions for environmental monitoring hardware, connected Internet of Things (IoT) devices, and cloud computing. Citizen scientists can take an active role in monitoring air quality at the neighborhood level by installing low-cost air quality sensors (LCS) that collect and report air pollutant data. Particulate matter (PM) is an air pollutant that is categorized based on size with fine particulate matter ($PM_{2.5}$) defined as particles with aerodynamic diameter less than 2.5 μm. The ability to record and visualize hyper-local data in an intuitive and informative interface will likely spawn an increase in interest and interaction with environmental data sets due to the locally relevant nature of the information. On the other hand, non-intuitive or limited user interfaces and confusing user experiences may discourage citizen scientists from

* Corresponding author. 21865 Copley Dr. Diamond Bar, CA, 91765, USA.
E-mail address: bfeenstra@aqmd.gov (B. Feenstra).

https://doi.org/10.1016/j.envsoft.2020.104832
Accepted 6 August 2020
Available online 25 August 2020

interacting with the collected data. The increasing complexity, size, and resolution of today's environmental monitoring networks have created big data challenges leading to the emergence of a new field of study: Environmental Data Science (Gibert et al., 2018). Data science combines computer programming skills, math and statistical knowledge, and subject matter expertise (Conway, 2013). Free Open Source Software (FOSS) platforms play a vital role in the progress of research towards developing new methods for addressing environmental data science challenges. The R-environment and Python are two FOSS programing languages that are often used in environmental data science applications (Kadiyala and Kumar 2017a, 2017b). Open access to environmental data sets and related tools is foundational for environmental data science to thrive and develop. Environmental monitoring data can be considered open access when the data is available through a stable and consistent Application Programming Interface (API) that allows software and application developers to build applications to display and report that data in transparent and meaningful ways.

Environmental data scientists can access regulatory data via open API's (e.g., AirNow API, OpenAQ API) to create custom web applications for displaying air monitoring (AM) data (AirNow, 2020; OpenAQ, 2020). These AM data viewing websites are useful and provide information to the public at varying granularity spatially and temporally. Two examples of data viewing websites include the OpenAQ map and the World's Air Pollution: Real-time AQI (WAQI) map which both display international air quality monitoring data (OpenAQ, 2020; World Air Quality Index Project, 2020). OpenAQ uses a color scale (Fig. S1 in the Supplemental Information (SI)) that deviates from the common Air Quality Index (AQI) color scale to display air pollution concentrations. A special feature in the WAQI website is their use of calendar plots to display AM information. Data viewing websites that display modeled or interpolated air pollutant or AQI values are also available (BreezoMeter, 2020; IQAir, 2020; Plume Labs, 2020). When displaying data from both regulatory-grade instruments and LCS, the source and type of data displayed should be readily apparent. A lack of differentiating and identifying data sources may cause confusion for the end-user, especially if the LCS do not agree with nearby regulatory-grade instrumentation. With interpolated or modeled maps, often the user is not readily aware of the input parameters used to model air quality data. When viewing modeled air pollution information, the viewer should be cautious especially when data sources are not readily apparent and input parameters, whether defendable or questionable, for the data model are unknown to the end-user/viewer (Hagler et al., 2018). Broadly, the available sensor data viewing platforms are map-centric with point values or interpolated modeled data displayed with options for viewing recent time series data.

Resources for accessing and displaying data collected from networks of LCS are available, though they vary in terms of software (FOSS or proprietary), what they provide, and whether they are provided by the manufacturer, a project team, or through a citizen science model. While many sensor manufacturers have software and platforms in place for ingesting, storing, and analyzing data that is generated from their respective sensors, these are often proprietary and offered as a Software as a Service (SaaS) or Platform as a Service (PaaS) requiring accounts with monthly or annual subscriptions costs. In contrast to the SaaS and PaaS business model, several sensor resources are available for open-access viewing of data collected from LCS networks. These platforms include but are not limited to the HabitatMap AirCasting map, Air Quality Egg Portal, Luft Daten project map, PurpleAir Map, Smart Citizen Kit Map, and the uRADMonitor Network map (Air Quality Egg, 2020; HabitatMap, 2020; Luftdaten, 2020; PurpleAir, 2020; Smart Citizen Kit, 2020; uRADMonitor, 2020). PurpleAir provides open access to the data collected by the PurpleAir network of sensors through an API and provides open viewing and downloading of sensor data through the PurpleAir map. The Luft Daten project is a citizen science project with LCS reporting to a map and invites programmers to collaborate in this FOSS development through GitHub (OK Lab Stuttgart, 2020). When selecting a sensor in either the PurpleAir or Luftdaten GUI, the user is currently limited to viewing only the last seven days of data in a time series plot and current data on the map (accessed January 2020). To gain an understanding of the historical local AM data, the user is required to download, process, and visualize the data from these networks on their own, which may be a limiting factor to those without the environmental data science skills needed to perform such analysis. These sensor-specific online resources for viewing sensor data often do not include the regulatory AM data that may be publicly available through the AirNow or OpenAQ API and often do not indicate what, if any, quality control (QC) measures are taking place on the collected data before displaying publicly.

For community members to understand local air pollution trends, a more in-depth analysis of historical data is required. While map-centric GUIs work well for viewing real-time data, communities that monitor air quality in long-term deployments need additional plotting and viewing capabilities to access and understand their local historical AM data. A data dashboard for viewing and analyzing historical data would provide citizen scientist with a better understanding of local air pollution levels, particularly spatial and temporal air pollution trends. For those with varying levels of technical data science programming skills, several software resources are available that support individual data analysis of air quality data. If data can be organized and loaded into a software system, then a more in-depth analysis can occur, and custom visualizations can be produced. FOSS software packages have been developed in the R and Python environments specifically for accessing and visualizing freely available AM data. These include the R packages *openair*, *PWFSLSmoke*, *ropenaq*, and *raqdm*. *OpenAir* provides a useful package for developing visualizations from collected AM data with functions to create calendar plots, scatter plots, and time variation plots along with wind roses, pollution roses, and bivariate polar plots if wind speed and direction data is available (Carslaw and Ropkins, 2012; Carslaw and Beevers, 2013).

If we use advanced analytical tools and access AM data directly, then we can facilitate more organized, robust, systematic, and repeatable data processing, analysis, and visualization of LCS data. Furthermore, using FOSS tools allows for increased iteration and development. An example of this workflow would be the *PWFSLSmoke* R package and the associated $PM_{2.5}$ AM web application developed as part of the AirFire tools by the U.S. Forest Service (USFS) Wildland Fire Air Quality Response Program (WFAQRP) (Callahan et al., 2019; Air Fire Tools, 2020). These tools were developed to access regulatory grade AM data via the AirNow API and display that data graphically to assist the USFS Air Resource Advisors to gather air quality data and create air quality reports during wildfire smoke events. The *PWFSLSmoke* R package provides functions to download, parse, and plot AM data and provides the back-end software necessary to generate plots for displaying on the front-end web application. A similar model in which an R-package is used for accessing and processing LCS data would save users time and would allow the development of custom functions for different approaches to QC and more complex historical data analysis, which are gaps we see in the current offerings. Additionally, the R-package could provide the back-end software to support a front-end web application to display historical AM data to provide communities with more useful analysis and visualizations of historical data. This web application would allow community members to answer questions about their local environment, which are not readily answered with the current offerings of real-time maps with limited historical data analysis.

The objectives of the software development associated with this project were to build an FOSS R package and data viewer web application that would address the challenges identified with the data management and visualization of LCS networks deployed within the U.S. EPA Science To Achieve Results (STAR) grant project. This paper summarizes the development of an R package and web application designed to address the environmental data science challenges created by deploying 400+ LCS in 14 different communities. We wanted an open source R package that would allow users to download sensor data, add

spatial metadata, perform data fusion with other relevant data sets, and create maps and plots for viewing data collected by AM sensors. We also wanted the package designed with functions so that minimal coding would be required to complete tasks. Understanding that many would prefer to interact with an online web application, we wanted to build an application that would provide an interactive data experience allowing users to make selections and explore the community AM data sets by generating pre-defined data visuals based on their user input selections. The South Coast Air Quality Management District (South Coast AQMD) collaborated with Mazama Science to develop the R package *AirSensor* and web application *AirSensor DataViewer* (*DataViewer*) to meet these software development aims.

2. Methods (Software design and characteristics)

2.1. Community engagement

In 2016, South Coast AQMD was awarded a U.S. EPA STAR grant, titled "Engage, Educate and Empower California Communities on the Use and Applications of 'Low-cost' Air Monitoring Sensors" under Assistance Agreement No. R836184. South Coast AQMD has engaged 14 California communities through a series of workshops to introduce the project, provide technical guidance on sensor technology and deployment (siting, installation, configuration, and registration) of air quality sensors, review deployment progress and examine community data sets, and provide software tools and resources for citizen scientists to engage with collected data sets and create informative data visualizations. Roughly 400 PurpleAir PA-II sensors (PurpleAir LLC, USA) were distributed to community members. The on-going engagement with the STAR Grant Sensor Communities (SGSC) has provided the motivation to develop software tools to enhance the community members' ability to interact with historical data and extract meaningful information about their local environment. Participants were not engaging with the data that often (as is supported by the survey data, which is most respondents only check their air quality data "sometimes" - 36% as opposed to "often" - 17% and "everyday" - 5%). In person discussions provided useful context to help us understand this by (1) reporting that data was difficult to access and download (especially. historic data), and (2) sharing what they wished to do with the data. For example, after displaying a static time of day bar chart showing the diurnal $PM_{2.5}$ trends during a community workshop, one community group leader asked, "How do I generate that plot on a regular basis and share with my community members?" In one SGSC, a sensor host wanted to know the best time of day to walk their dog to reduce their exposure to particulate pollution. Additionally, multiple participants from different communities shared their difficulty downloading and analyzing the publicly accessible PA-II data especially with regards to the time/date reformatting required for plotting in Microsoft Excel. The survey responses along with the discussions with community members on the data science challenges provided the motivation to build additional software tools to address the difficulty and challenges posed by analyzing these large community AM data sets. Increasing the number of data-sharing events with effective data visualizations should provide participants with a better understanding of the principles of air quality, their local air pollution, and the proper use and application of LCS (Sandhaus et al., 2019). Table S1 in the SI provides a summary of the environmental data science challenges that are addressed in this project.

2.2. Software tools (R environment, RStudio, R packages, and Shiny)

The R environment is an integrated suite of software facilities that is designed on a simple yet effective computer programming language, R. The R environment provides tools and functions for data processing, storage, calculation, and graphical display. Since R is designed essentially on a computer programming language, users are able to add further functionality to existing packages by defining new functions and developing packages of functions (The R environment 2019). RStudio, a public benefit corporation, provides a FOSS version of an Integrated Development Environment (IDE) for R which supports code execution, debugging, and workspace management (RStudio, 2019; Allaire, 2020). Instructions for installing R and RStudio can be found on the web and in the literature (Kadiyala and Kumar 2017b). The fundamental unit of shareable code in R is a package. Packages bundle together R code, data, documentation, and tests. Packages are sharable on the Comprehensive R Archive Network (CRAN), which is the public clearing house for R packages. CRAN hosts a wide variety of FOSS packages that allow researchers to collaborate and build upon already developed R code. The development of *AirSensor* built upon R packages available on CRAN; most notably *MazamaSpatialUtils, openair*, *PWFSLSmoke, and worldmet*. *AirSensor* is designed to be used with R version $\geq$3.3. This paper describes version 0.5 of the *AirSensor* package which is available on GitHub. The latest or master branch of *AirSensor* is also available on GitHub. The *AirSensor* package can be installed using the *devtools* package within R using the following code:

```
devtools::install_github("MazamaScience/AirSensor@version-0.5")
devtools::install_github("MazamaScience/AirSensor", ref = "master")
```

Shiny is a FOSS R package that provides a framework for building interactive web applications. Shiny allows the user to turn R derived analysis and plots into interactive web applications without requiring HTML, CSS, or JavaScript programming. Shiny allows for the development of a web application for viewing and sharing data analytics. Since not all users would be comfortable using the R environment which does require coding, R Shiny was used to develop the *DataViewer* web application to provide an interactive data experience for community members that would prefer to interact with the sensor data in a web application rather than in the R programming environment.

2.3. AirSensor - R package

Rather than describing each individual function in *AirSensor*, the following examples will showcase the three primary data objects available through the package, how to apply quality control measures on the imported data, and how to generate plots for each of the data objects. A complete guide to *AirSensor* functions and operations can be found within the R-environment after the package has been loaded. Helpful R vignettes are also available within the package to provide the user with code examples for using the *AirSensor* functions and working with the sensor data.

2.3.1. Data access, extraction, and data objects overview

AirSensor currently accesses data generated by PurpleAir sensors by collecting real-time data from www.purpleair.com/json and historical data from a ThingSpeak Representational State Transfer (REST) API. Extracted data is enhanced with spatial metadata and transformed into efficient data objects for downstream analytics. The three primary data objects are the Purple Air Synoptic (PAS), Purple Air Timeseries (PAT), and AirSensor (*sensor*) data objects. Functions exist for creating or loading data objects as well as manipulating and visualizing them. An overview of the *AirSensor* R package data access, data objects, and functions is provided in Fig. 1. After installing or loading the package, a data archive repository can be set to access archived data. Data archives can be created that for specific sensor networks (e.g. SGSC) or for a specific geographic area (e.g., Southern California) so that the R user can access and load historical data more efficiently from an archive rather than the ThingSpeak REST API. The data archives developed for the SGSC are kept current with cron jobs (cron jobs are time-based jobs that can run commands at specific time intervals) that are scheduled to run every hour to pull and add the most recent data to the archive. The data archive for the SGSC is accessible at http://smoke.mazamascience.com/data/PurpleAir and includes historical data starting from October 01,

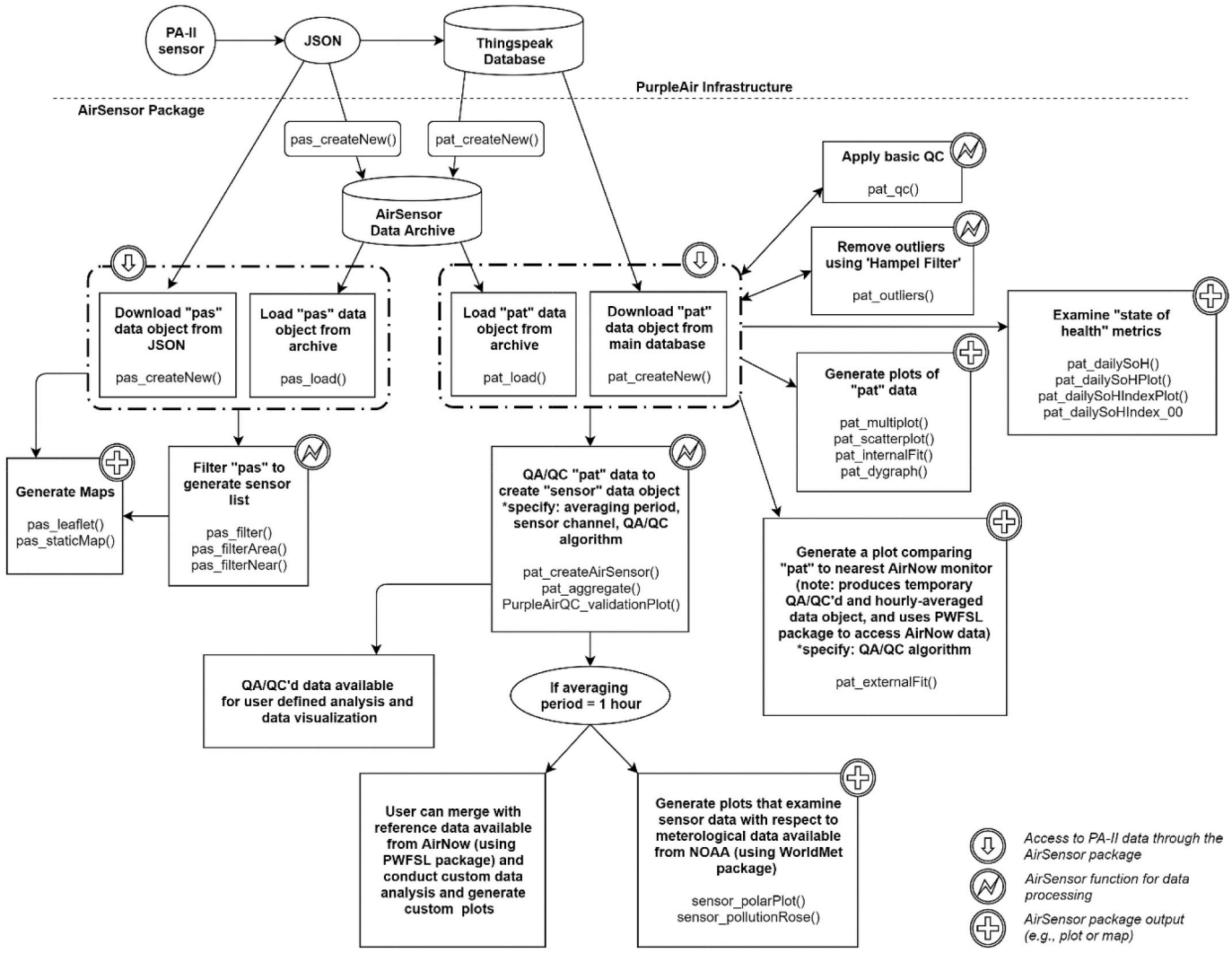


Fig. 1. Flow Chart for data flow and functionality of the *AirSensor* R package.

2017. A base archive can be set in *AirSensor* by the following code:

```
setArchiveBaseUrl("http://smoke.mazamascience.com/data/PurpleAir")
```

2.3.2. Purple Air Synoptic - Data object

The Purple Air Synoptic (PAS) data object provides an instantaneous view of the measured values from a network of sensors. A PAS can be created from the JSON data available at www.purpleair.com/json or can be loaded by accessing a data archive (Fig. 1). At the time of this writing, the time resolution of the PA-II sensors is 120 s and therefore a new PAS data object would be available roughly every 120 s. The available functions for manipulating PAS data object include pas_filter(), pas_filterArea(), and pas_filterNear(). The PAS data can be plotted on a map to display the instantaneous data collected by the sensor network with the pas_leaflet() and pas_staticMap() functions. Fig. 2 shows a PAS data object displayed on an interactive map using the pas_leaflet() function which maps sensor locations and colors the locations according to AQI. The map is interactive in that the user can select an individual sensor and view the values recorded at that location for the time the PAS object was created. If a user is interested in loading specific states or air districts, the user can apply filters when generating the PAS data object. The leaflet map can be modified with options for map tiles, parameter displayed, and what type of sensors to display (i.e. inside or outside sensors). Fig. 2 was produced by the following two lines of code:

2.3.3. Data fusion enhancements

Data fusion with other relevant data sources provides benefits for custom analytics, for performing data quality checks, and for providing information on local weather conditions. Data fusion provides the ability to tell a more complete story about local air pollution by fusing collected sensor data with other publicly available data sets. *AirSensor* has been integrated with the *PWFSLSmoke* R package for access to regulatory AM data via the AirNow API and integrated to the *worldmet* R package for access to the U.S. National Oceanic and Atmospheric Administration (NOAA) Integrated Surface Database for meteorological data (Callahan et al., 2019; Carslaw, 2019). These data fusion enhancements provide the ability to generate comparison plots between a LCS and the nearest regulatory-grade instrument and allow for sensor data to be joined with nearest meteorological data so that wind roses, pollution roses, and bivariate polar plots can be generated to provide insights into local air pollution trends. Data fusion enhancements are performed on both the PAT and *sensor* data objects.

2.3.4. Purple Air Timeseries (PAT) - Data object and quality control functions

The PAT timeseries data object provides timeseries data on a per-sensor basis. Data manipulation functions for the PAT data object include filtering, sampling, and joining. A PAT can be loaded from a data archive using the pat_load() function or can be created from the PurpleAir ThingSpeak API with the pat_createNew() function. The code

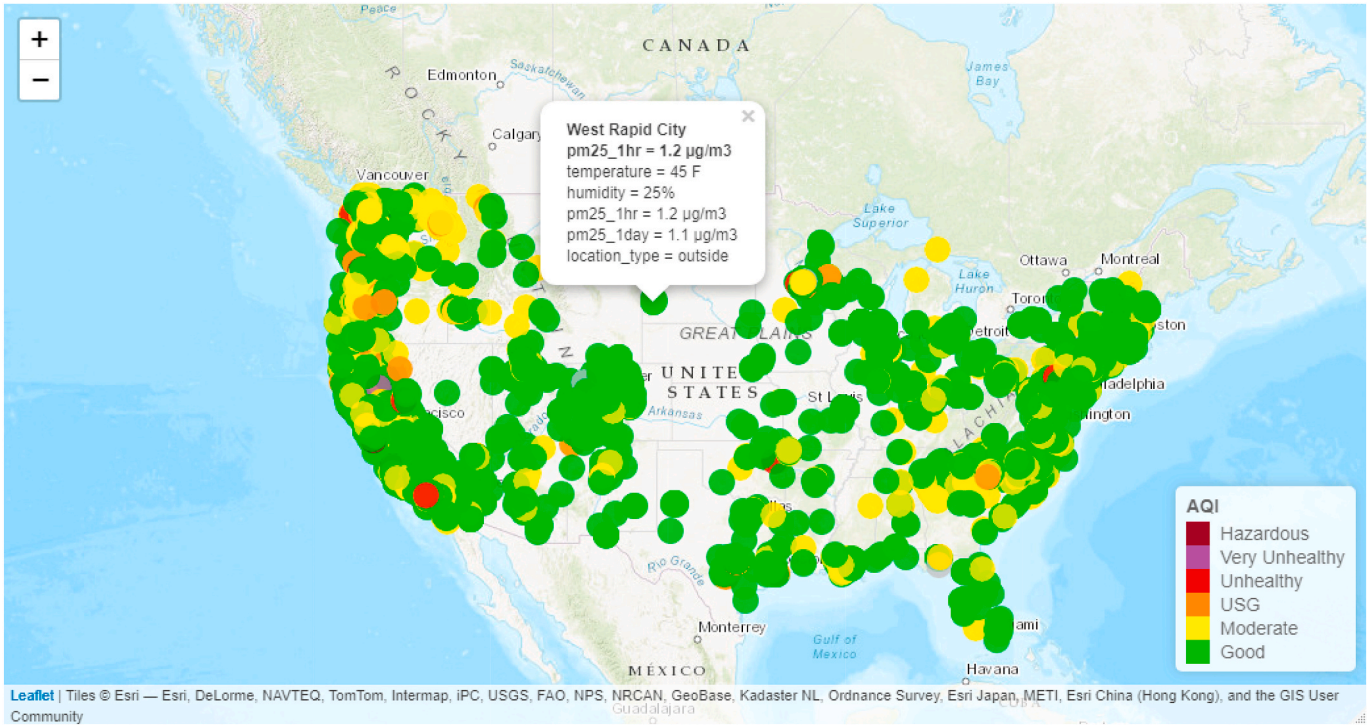


Fig. 2. Interactive leaflet map created from a PAS data object.

```
pas_example <- pas_load()        # Load synoptic PurpleAir Data for U.S.
pas_leaflet(pas_example)         # Create an interactive map of synoptic data for U.S.
```

example below loads a PAT data object for a sensor in Seal Beach, CA that was deployed as part of the SGSC deployments. The PAT data object includes data from January 01 to December 31, 2018. Subsequent example code and plots displaying the *AirSensor* functions will be performed on this PAT data object or a filtered PAT data object created from the SCSB_20 sensor. The PAT data object can be loaded into the R environment and filtered by date with the following code:

```
pat_example <- pat_load(
  label = "SCSB_20",
  startdate = 20180101,
  enddate = 20181231,
  timezone = "America/Los_Angeles",
)
# Create a PAT filtered for June/July 2018
pat_JuneJuly <- pat_filterDate(
  pat_example,
  startdate = "20180627",
  enddate = "20180708"
)
```

PAT data objects can be processed for time averaging, QC algorithms, and outlier detection for removal or replacement. The user can create their own framework for applying QC functions depending on their project requirements. The pat_aggregate() function returns a data frame with aggregate statistics which are helpful for building out QC algorithms. The aggregate statistics include the mean, median, standard deviation, minimum, maximum, and count for the aggregate time period chosen. Note that the PA-II sensor node is manufactured with two identical OEM (original equipment manufacturer) PM sensors (model PMS 5003, Plantower, China) that report the same types and amounts of data and for reference purposes are labeled as channel A and channel B, respectively. For the paired channel A and B $PM_{2.5}$ data columns, the pat_aggregate() function also returns the *t*-test statistic (based on an unpaired, two-sample student's t-test), p-value, and degrees of freedom. Several built-in QC algorithms are available in *AirSensor* and are labeled as pat_qc, hourly_AB_01, and hourly_AB_02. The pat_qc function allows the user to perform a first-level QC check for values that are considered "out-of-spec" with regards to the manufacturer defined specifications for the acceptable ranges for $PM_{2.5}$, temperature, and humidity. The PurpleAirQC_hourly_AB_00() function allows the user to perform an hourly average of the A and B sensor channels when sufficient sub-hourly data exists for both channels within an hour. The default min-count for sub-hourly data is set to 20 data points; requiring a data recovery for A and B channels >66% for the current time-resolution at 120-s. No further QC is applied with this function. Note that the PA-II's time resolution has changed with firmware updates over time. As firmware updates have not been performed across the board simultaneously for all sensors in the PurpleAir network, the following dates are estimates for firmware releases and data resolution. Time resolution for data prior to February 2017 is 20 s, from February 2017 to March 2017 is 40 s, from March 2017 to May 2017 is 70 s, from May 2017 to May 2019 is 80 s, and data recorded after May 2019 is 120 s. The function PurpleAirQC_hourly_AB_01 allows the user to perform an hourly average of the A and B sensor when sufficient sub-hourly data exists and when data is considered statistically similar. Data is invalidated when (1) minimum count < 20 values, (2) when both the means of channels A and B are not statistically the same (two-sample *t*-test p-value $< 1e^{-4}$) and the mean difference between channels A and B is greater than 10 $\mu g/m^3$, and (3) when the mean difference between A and B is greater than 20 $\mu g/m^3$ for $PM_{2.5}$ values less than 100 $\mu g/m^3$. These conditions assume that the air entering the channel A and B sensors is the same and therefore the means of the two channel measurements should be statistically similar. When measurements from these sensor channels agree, the user can have

higher confidence in the LCS air quality measurements and the subsequent data averaging of the two -OEM sensors into one value. The two-sample *t*-test is a statistical technique to determine whether the difference between two means is significant. The default settings of these QC checks can be modified to adjust the QC check to individual project requirements. Additionally, new QC functions can be created and *AirSensor* users are encouraged to create their own custom QC functions and submit these functions to be added to the *AirSensor* package through GitHub. The PurpleAirQC_validationPlot() function creates a series of timeseries plots for channel A and B, the difference between channel A and B, *t*-test p-value, min count, and the hourly averaged final output (Fig. 3).

The *AirSensor* pat_outlier() function provides an outlier detection function that allows the user to apply a rolling Hampel filter to identify points that may be outliers, and if desired, replace those identified outliers with a rolling median value. The Hampel Filter is an outlier detection technique that uses the Median Absolute Deviation (MAD). For each data point, a median and standard deviation are calculated using neighborhood values within a sample window size. If the MAD of a single data point is a specified number of standard deviations (threshold minimum) from the median value for the sample window, then the data point is flagged as an outlier. The default values for the pat_outlier() function set the sample window = 23 and the threshold minimum = 8. Adjusting the default parameters on the function for identifying outliers would adjust the number of points detected as outliers. Fig. 4 provides an example of the pat_outlier function with the potential outliers identified as red asterisks. In this example, a date filter was applied to the pat_example previously generated to only include the June 27 to July 8, 2018 time period that would be impacted by a special event: 4th of July fireworks. The outlier detection function appears to identify many of the one-off high values as outliers but does not consider the elevated $PM_{2.5}$ concentrations due to the fireworks to be outliers. This function allows *AirSensor* data users to quickly implement an outlier detection technique and visualize the results of their outlier detection function. Fig. 4 was produced by the following R-code:

```
pat_outliers(pat = pat_JuneJuly)    # generate the outlier plot
```

Data visualization functions for the PAT include plotting raw data time series, interactive time series, multiplot time series (A, B, Temp, RH), comparison plot for channel A vs. B, and a comparison plot with regard to the nearest regulatory $PM_{2.5}$ monitor. The channel A and B $PM_{2.5}$ timeseries data can be compared using the pat_interalFit() function as shown in Fig. 5. For SCSB_20, the A and B sensors agree with each other with an $R^2 > 0.98$, slope of 1.05, and an intercept of −0.8. Since the two sensors perform similarly for 2018-time frame, the blue timeseries points representing the B sensor are plotted over top of the red points representing the A sensor. The code to generate the plot is:

The pat_scatterplot function provides a multi-panel scatterplot for variables in the PAT data object with an example of the plot shown in Fig. 6. This plot allows the researcher to determine if there is a lack of correlation between the A and B sensor channels or if there are higher than expected correlations between $PM_{2.5}$ concentrations and weather

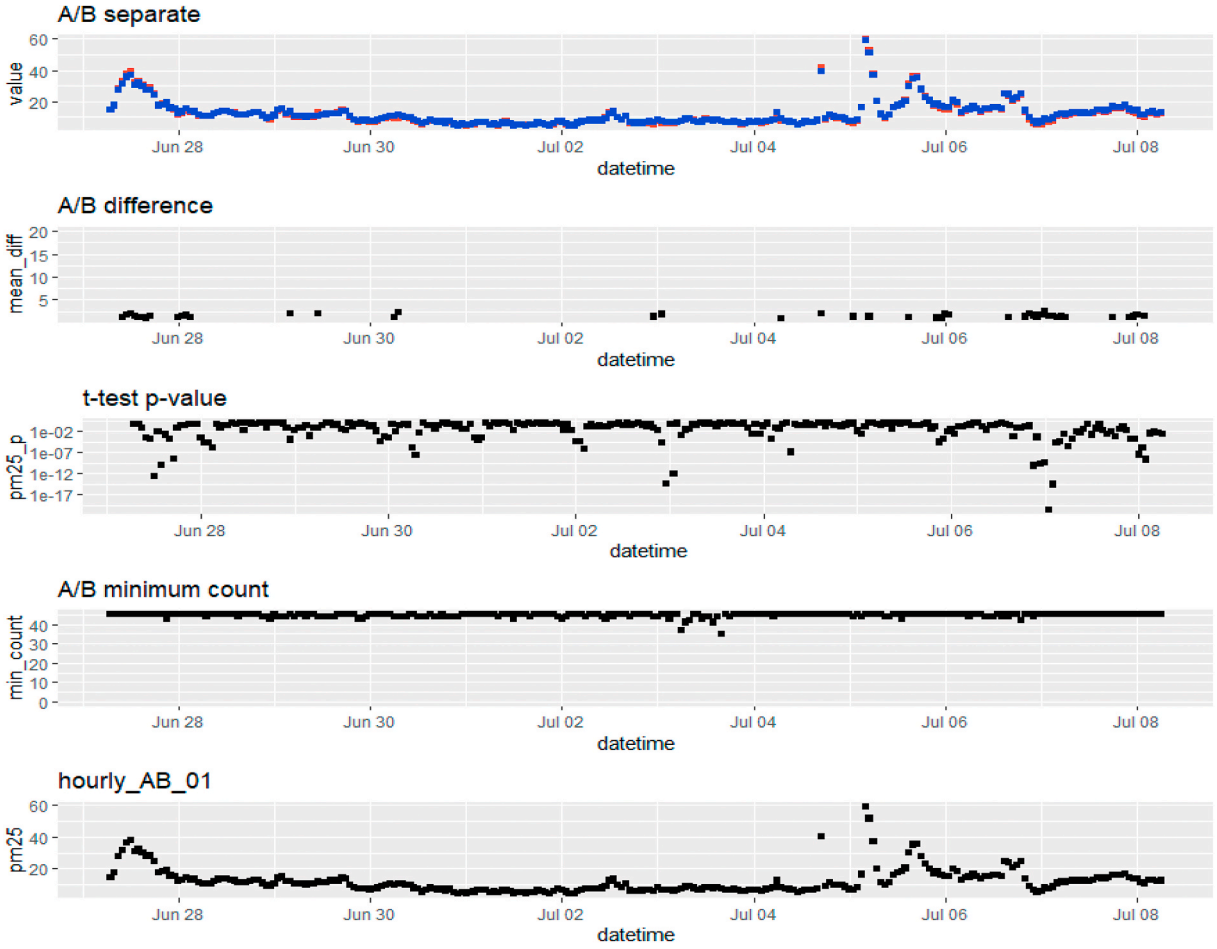


Fig. 3. Plot generated to visualize the QC_01 algorithm for SCSB_20 located in Seal Beach, CA. A/B separate provides a timeseries plot for channel A and B; A/B difference provides a timeseries of the mean difference between the A and B channels; *t*-test p-value provides a timeseries of the p-value statistic between the two channels; A/B minimum count provides the minimum count of data points in 1-hr for the A and B channels; and hourly_AB_01 provides the 1-hr quality controlled average data processed with the hourly_AB_01 function.

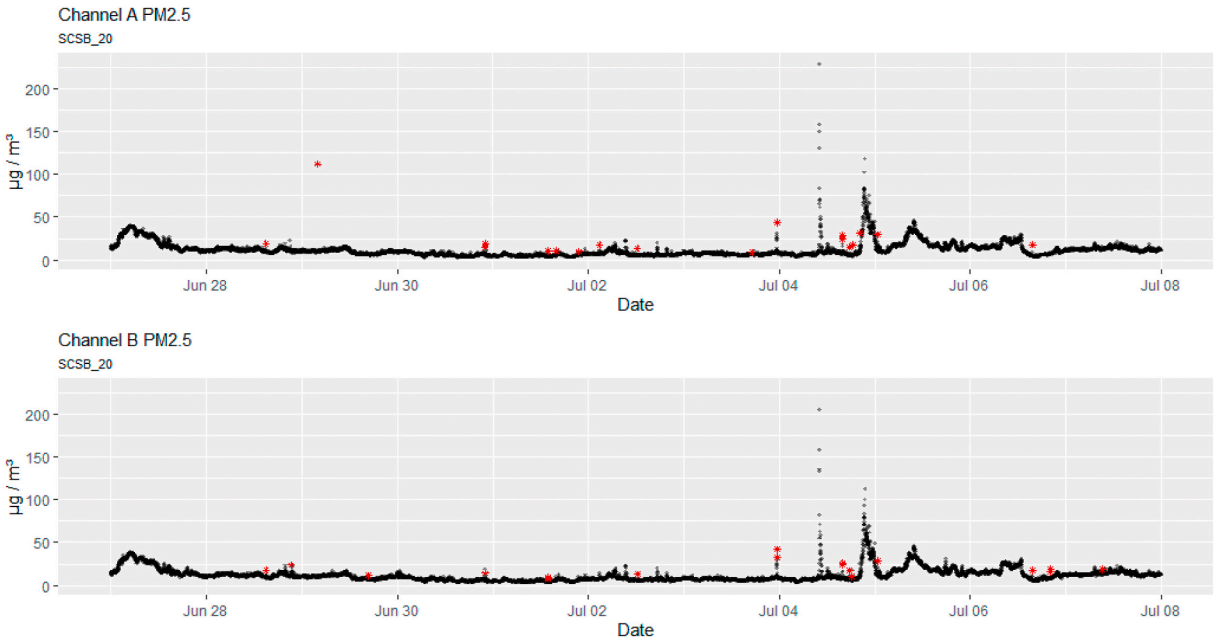


Fig. 4. Plot generated with the rolling Hampel filter identifying potential outliers in red asterisks for SCSB_20 from June 27 to July 08, 2018. (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

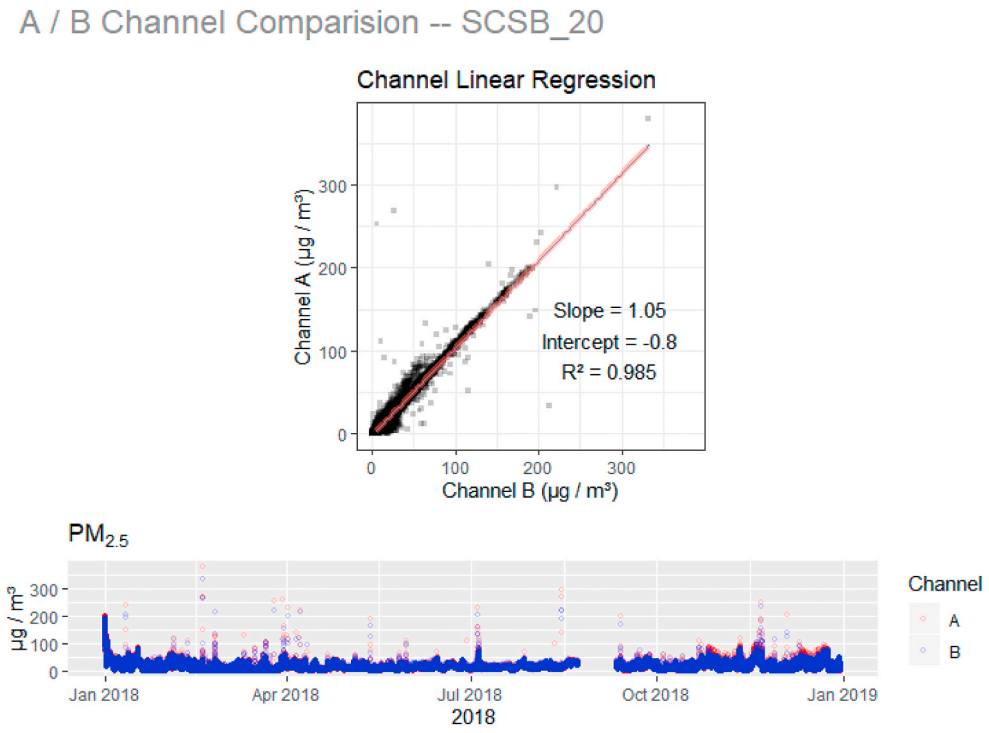


Fig. 5. Scatter plot and timeseries rendered using the pat_internalFit function to compare channel A and B within a single PA-II sensor, SCSB_20, for 2018.

```
pat_internalFit(pat_example) # Create timeseries and scatter plot for A/B
```

conditions (temperature and humidity). This plot also provides the timeseries and distribution of data points for PM, temperature and humidity. In Fig. 6, the distribution plots for the A and B sensor channels indicate $PM_{2.5}$ concentrations for this sensor are typically less than 25 $\mu g/m^3$. The datetime column provides an indication of periods of downtime with a noticeable downtime seen in August and September of 2018.

A sensor can also be compared to the nearest regulatory air monitoring station (AMS) with the pat_externalFit() function (Fig. 7). In this example, the sensor is 3.1 km away from the regulatory AMS equipped

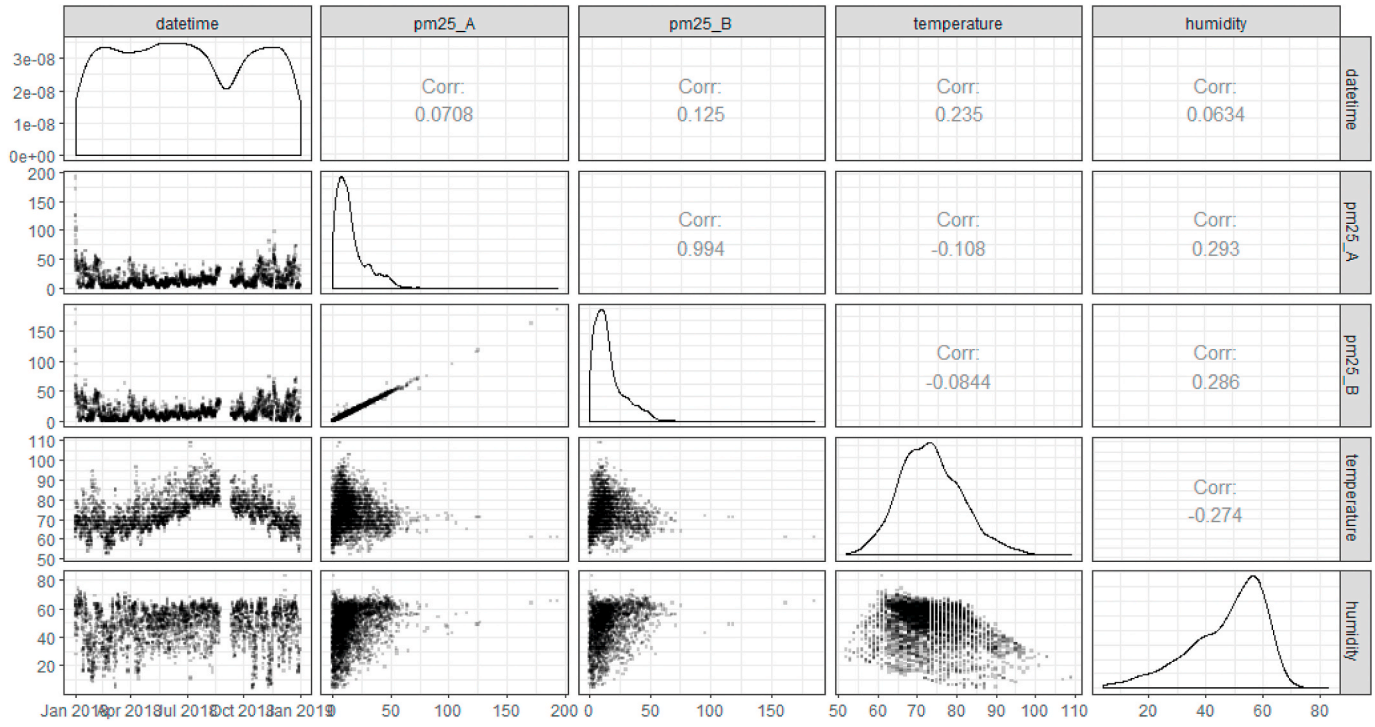


Fig. 6. Plot generated using the pat_scatterplot function to graphically view the variables in the PAT timeseries data object.

with a Met One Beta Attenuation Monitor (BAM), which is a U.S. EPA designated Class III FEM (EQPM-0308-170) for $PM_{2.5}$. The time resolution of the regulatory $PM_{2.5}$ data is hourly. To match LCS data with the regulatory data, this function uses the QC procedures previously described to hourly aggregate the sensor data. The user can specify which QC algorithm to apply or create custom QC functions. Fig. 7 indicates that while the sensor follows the typical daily $PM_{2.5}$ trends of the nearby regulatory-grade instrument for $PM_{2.5}$ with $R^2 > 0.73$, the sensor tends to estimate higher concentrations than the regulatory-grade instrument. This slope/intercept offset could be due to a local emission source impacting this particular sensor location or could be due to sensor measurement bias error that has been identified in prior publications (Feenstra et al., 2019; Magi et al., 2019). For the time-series in Fig. 7, the purple colored points represent the 1-hr PurpleAir sensor data

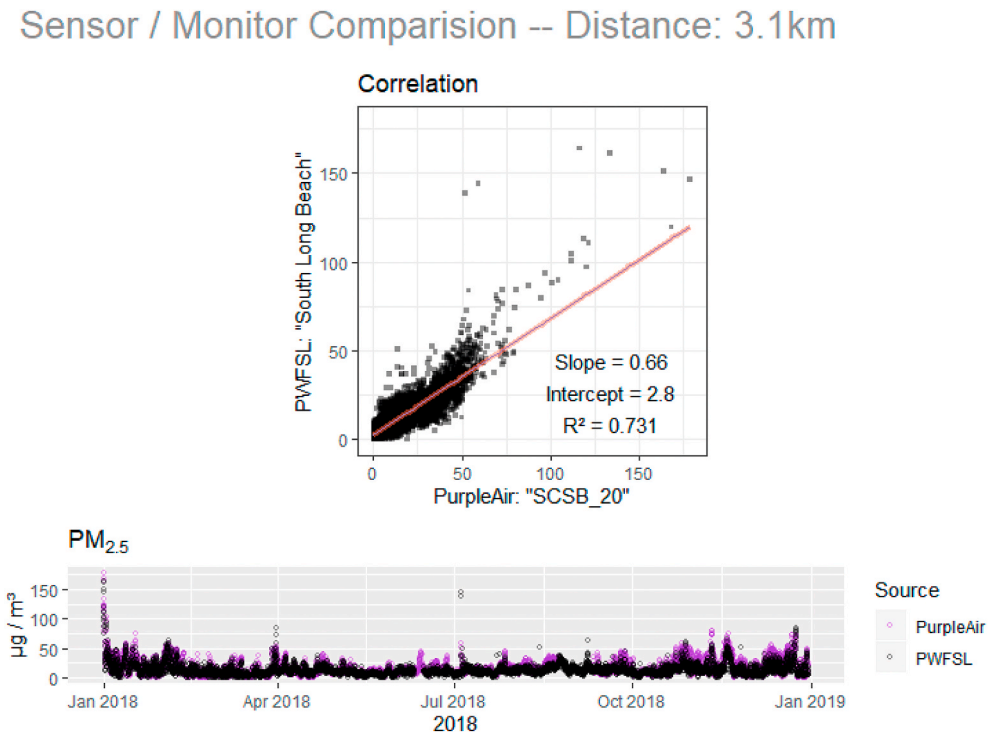


Fig. 7. Scatter plot and timeseries plot rendered using the pat_externalFit() function which compares the PA-II sensor, SCSB_20, in Seal Beach, CA to a nearby regulatory-grade $PM_{2.5}$ instrument in Long Beach, CA.

and the black colored points represent the regulatory-grade instrument data. If the two agree closely within an hour, the black point would be plotted on top of the purple point for that hour. The plot in Fig. 7 is created with the following:

```
pat_externalFit(pat_example)
```

The pat_dygraph function returns an interactive time-series plot for both channel A and B allowing the user to zoom in/out and investigate date/times when $PM_{2.5}$ concentrations may be higher than normal (Fig. 8). Using the interactive time-slider located below the plot allows the user to quickly zoom in to further investigate dates and times with particle pollution events. With a small amount of code, the dygraph provides a versatile, interactive plot, where the user can explore a large amount of data at customizable levels with the time slider and zoom in/out features. Fig. 8 is created with the following:

```
pat_dygraph(pat_example)
```

2.3.5. Hourly QC data object (sensor)

The *sensor* data object is generated on a per sensor basis from a PAT data object with the pat_createAirSensor() function. The user will need to specify a PAT data object, time averaging period, parameter, channel, QC algorithm, and minimum count. The QC algorithms applied in creating the *sensor* data object are described earlier in Section 2.2.4 with regards to the QC functions that can be applied to a PAT timeseries data object. The functions for *sensor* data objects begin with "sensor_". An example creating a *sensor* data object is shown in the code below:

```
AirSensor_example <- pat_createAirSensor(
      pat=pat_JuneJuly,
      period = "1 hour",
      parameter = "pm25",
      channel = "ab",
      qc_algorithm = "hourly_AB_01",
      min_count = 20

)
```

Plots available for the *sensor* data object within the *AirSensor* package include a bivariate polar plot and pollution rose, which wrap functions from the openair R package. The meteorological data used to generate these plots is retrieved from the NOAA *worldmet* R package. The bivariate polar plot and pollution rose, which are shown in Fig. 9 and Fig. 10 respectively, provide the user with the ability to couple wind direction and wind speed with $PM_{2.5}$ pollutant data to determine whether pollution events can be attributed to specific meteorological conditions and potentially identify pollution sources. A more in-depth analysis of these plots and their application in analyzing AM datasets is accessible within the published literature on the 'open-air' R package development (Carslaw and Ropkins, 2012) and use of bivariate polar plots (Carslaw and Beevers, 2013; Grange et al., 2016). The pollution rose and polar plot are generated by the following code:

2.3.6. Timestamp and time averaging for AirSensor data objects and functions

AirSensor and *AirSensor* functions have been designed to appropriately handle timestamps and various time zones of potential users. Users should understand how time stamps are stored and visualized within *AirSensor* and take appropriate steps when creating and visualizing *AirSensor* data objects; especially if using plotting functions outside of the *AirSensor* package to visualize data. The PurpleAir API provides access to data stored in Coordinated Universal Time (UTC). The *AirSensor* data objects (PAS, PAT, and sensor) all store data with a UTC timestamp. When creating or loading either a PAT or a *sensor* data objects, the user can specify the local time zone of the sensor selected. If a time zone is not specified when creating a data object for a single day, the date/time parameters will be passed as UTC, which for a sensor located in the Pacific Time Zone (+8h UTC) would return a data object with data from 08:00 AM to 08:00 AM local time of the following day. In *AirSensor,* time stamps are labeled and time averages are coded as "time beginning". For example, a 1-hr time average with a timestamp of 14:00 would be an average of the data collected between 14:00 and 14:59. This holds true even with the 2-min time-matched channel A and B sensor data available in *AirSensor*. The PurpleAir PA-II channel A and B sensors report at different times within a 120-sec time interval. In *AirSensor*, the seconds are dropped and data from the A and B sensor are assigned to a 2-min time beginning time stamp for matching purposes between the two OEM sensors within a PA-II sensor. Since data is stored as UTC, the plotting functions within *AirSensor are* coded to appropriately apply time shifts based on the sensor's location (time zone) so that data will be plotted and displayed in the local time of that sensor's location.

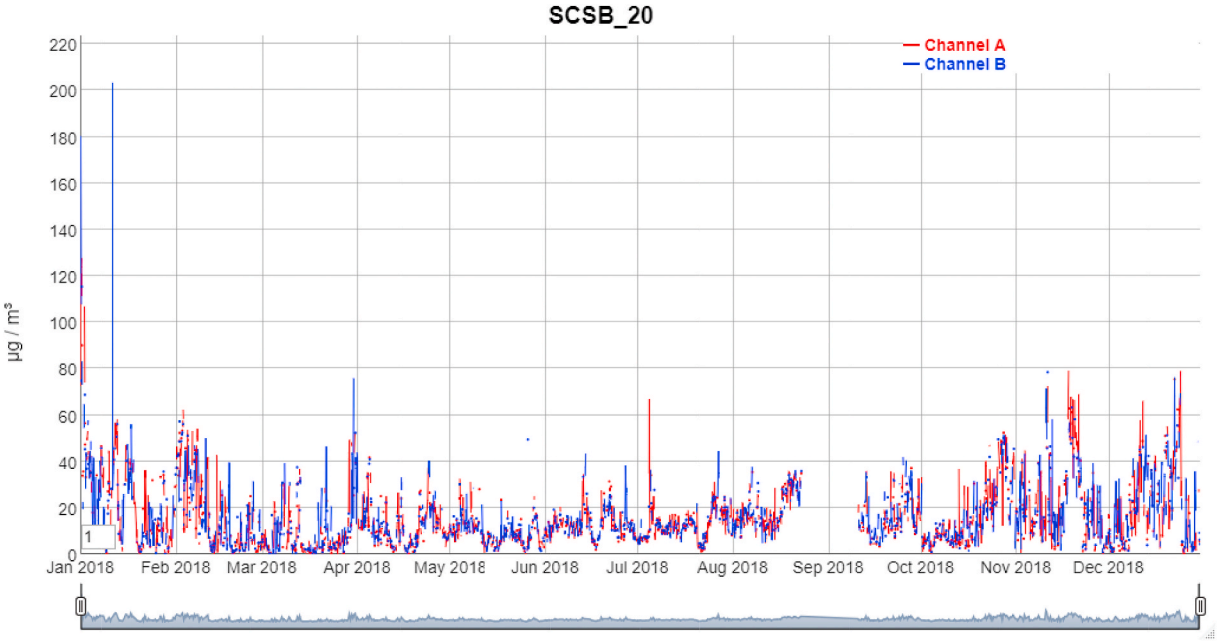


Fig. 8. Dygraph plot with interactive time-slider generated by the pat_dygraph function.

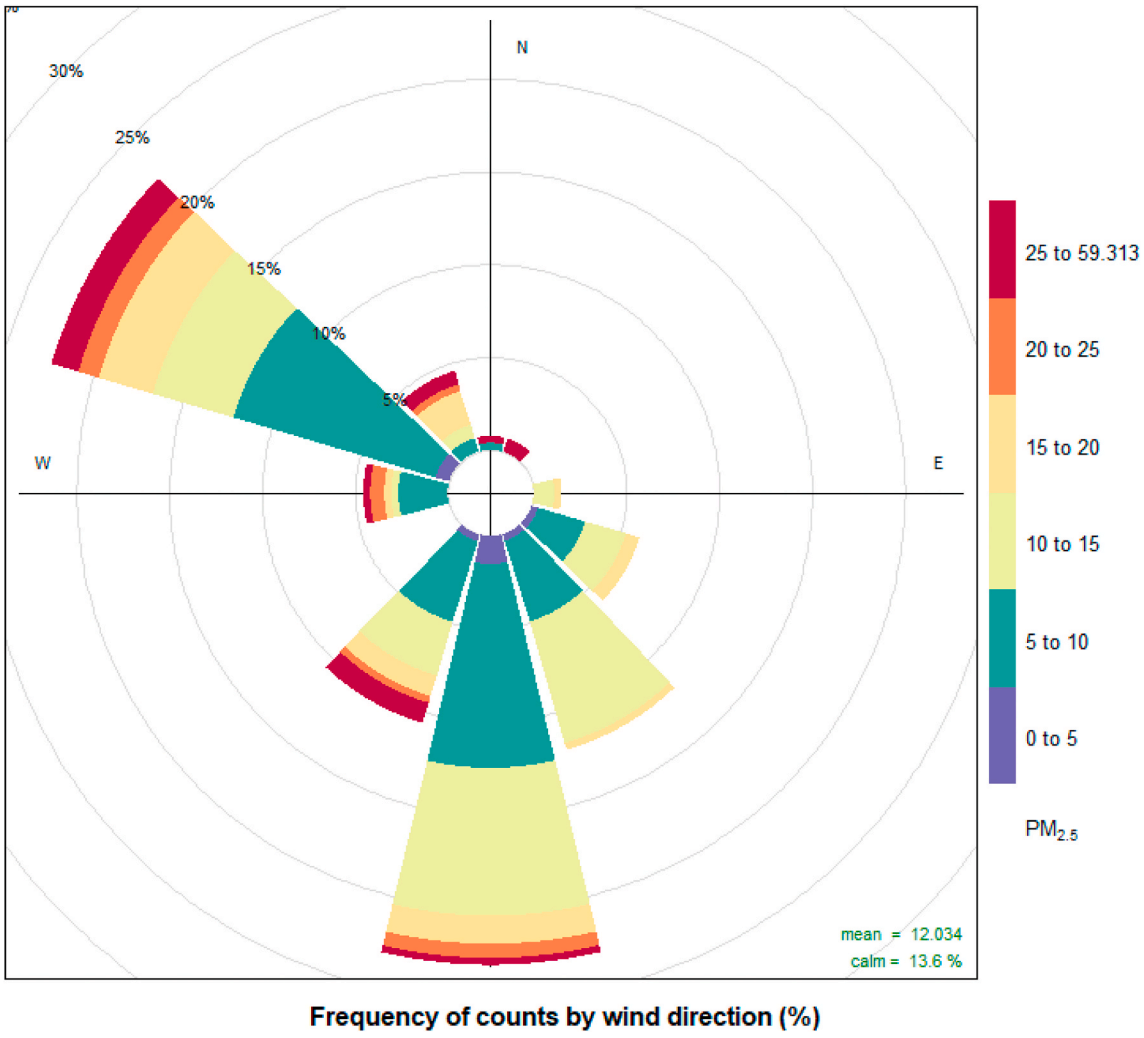


Fig. 9. $PM_{2.5}$ Pollution Rose generated by the sensorpollutionRose() function for SCSB_20 located in Seal Beach, CA for June 27 to July 08, 2018.

2.4. AirSensor DataViewer web application

2.4.1. AirSensor DataViewer overview

The *DataViewer* application was developed to provide an online interactive data experience for the SGSC networks. These communities and sensor names are listed in SI Table S2. This interactive web application provides access to the functionality of the *AirSensor* R package. Citizen scientists that would not be able to download R and run code or scripts to access, process, and visualize community data are now able to visualize their community data through the DataViewer. While the infrastructure to generate the types of plots that had resonated with community group members during the workshops was developed in the *AirSensor* package, the ability for community group members to use that infrastructure and generate visualizations in an interactive web application without writing a single piece of code is provided in the *DataViewer* application. Plots that generated the most interest with community group members, including calendar plots, concentration maps, community time-lapse videos, and sensor performance plots between the A and B internal sensors and between the sensor and nearest reference $PM_{2.5}$ monitor, were prioritized for incorporation in the *DataViewer*. The following sections will provide an overview of the back-end infrastructure required for the *DataViewer* application and the methodology for the *DataViewer* color scale and timelapse videos. The front-end of the *DataViewer*, which is the online web application and the primary point of interaction for community members, is highlighted in the results section.

2.4.2. Cloud computing resources

Cloud computing provides computing services over the internet using a pay-as-you-go pricing model. Computing services typically include computing power, storage, networking, and analytics. Cloud computing can provide benefits by allowing programmers to focus on building new and innovative applications rather than acquiring and maintaining the infrastructure required for their computational needs. The cloud can provide benefits with cost reductions for IT infrastructure and can increase the scalability, elasticity, reliability, and security of computational services in comparison to computation services provisioned locally or on-premise. Azure, which is Microsoft's public cloud computing platform, was used to support the computational requirements of the *DataViewer* application. The application could also be run on another public cloud platform or on premise if desired. The computation services required include running scheduled tasks (cron jobs) for creating data objects, storing data in structured data directories, and hosting the *DataViewer* application. The data archive consists of a set of flat files defined by a simple directory and naming protocol with the data ingest scripts written in the R programming language. A virtual machine (VM) was configured on Azure with the structured directories for the data directories along with required software (i.e., Git, Apache, Docker, and R). A second VM was configured to host the *DataViewer* application. Fig. 11 provides a simplified system architecture for the *DataViewer* application.

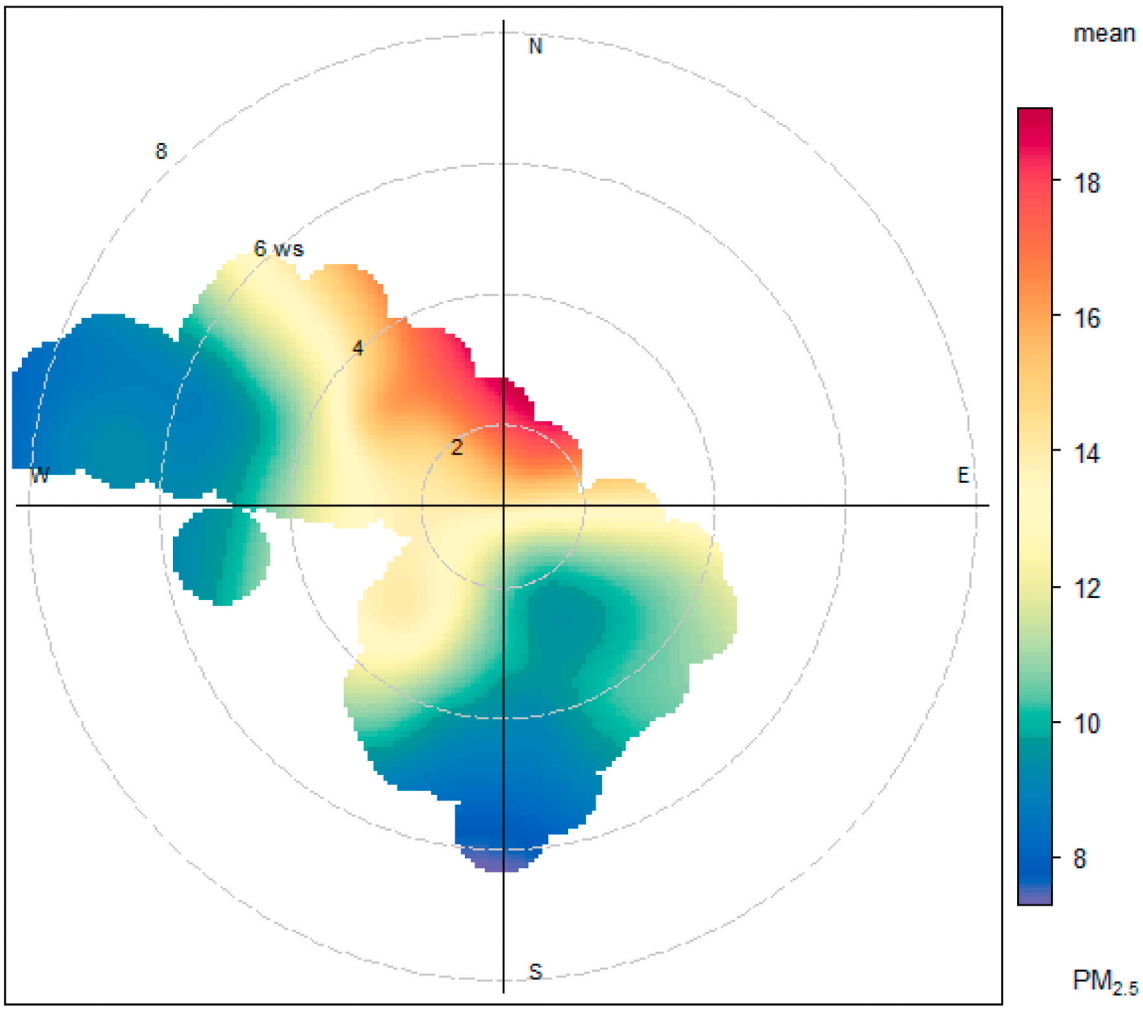


Fig. 10. $PM_{2.5}$ Bivariate Polar Plot generated by the sensorpolarPlot() function for SCSB_20 located in Seal Beach, CA for June 27 to July 08, 2018.

```
sensor_pollutionRose(AirSensor_example)     # Code for Pollution Rose
sensor_polarPlot(AirSensor_example)         # Code for Polar plot
```

2.4.3. DataViewer color scale

Determining an appropriate color scale for pollutant concentrations generated by LCS is challenging. Historically, air quality has been colored according to the AQI with values ranging from 0 to 500 with six distinct color categories; good (green), moderate (yellow), unhealthy for sensitive groups (orange), unhealthy (red), very unhealthy (purple), and hazardous (maroon). Historically, AQI has been calculated at 24-h averages due to the scientific information about air pollution exposure and public health. In 2013, the U.S. EPA released a new AQI calculation method (NowCast Reff method) for $PM_{2.5}$ that calculates AQI hourly based on the previous 12 h with the most recent hourly pollutant concentrations given larger weighting when air quality is changing rapidly (Mintz et al., 2013). The U.S. EPA in the Air Sensor Toolbox suggested a new pilot version color/concentration scale that could be used for 1-min high time resolution data from LCS (U.S. Environmental Protection Agency, 2019). This scale uses four shades of blue for low, medium, high, and very high $PM_{2.5}$ concentrations and is shown in SI Fig. S2. The scale from the AirSensor Toolbox was created for 1-min sensor data in contrast to this work in which LCS data is processed with QC algorithms and time-averaged to 1-hr concentrations prior to being displayed in the *DataViewer* application. Furthermore, the authors wanted to provide users with a clearer differentiation among the higher pollutant levels sometimes indicated by the sensors. Hence, a new color scheme was developed for the *DataViewer* that includes 5 concentration categories represented by two colors (blue and purple) with variations in the hue and luminance as shown in Table 1.

2.4.4. FF MPEG and digital stills and video stills creation

One of the desires of the community groups was to view historical time-lapse concentration maps to view past air quality events in their communities. To accomplish this task, cron jobs run hourly to create video still images for each of the 14 SGSC. These images are stored in the structured data directory in sequence and converted into mp4 video files using FFmpeg, which is a FOSS (Dawes, 2019; FFmpeg, 2019).

3. Results

3.1. AirSensor package

The *AirSensor* R package meets the community needs for those desiring to work with PurpleAir LCS data programmatically in the R environment. Through the *AirSensor* package, real-time and historical

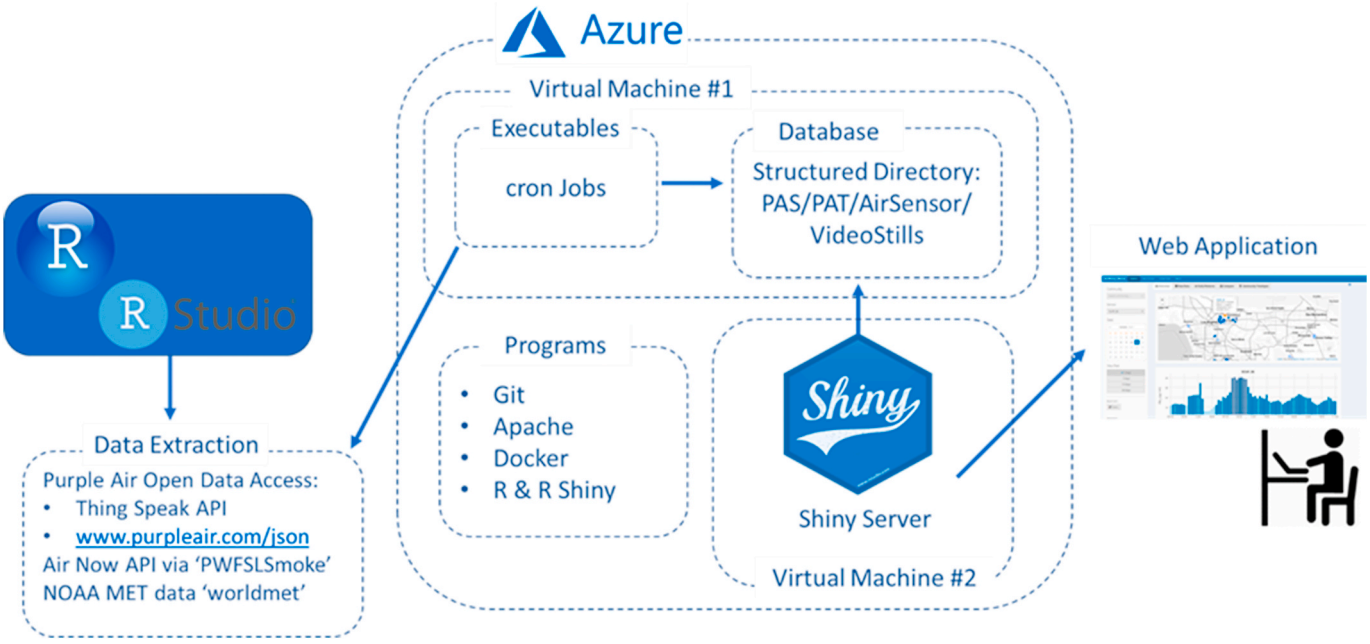


Fig. 11. System architecture for the *DataViewer* application.

Table 1
DataViewer color/concentration scale for 1-Hr $PM_{2.5}$ concentrations.

Color Hex # (RGB)	$PM_{2.5}$ Concentration ($\mu g/m^3$)
#ABE3F4 (171,227,244)	$PM_{2.5} \leq 12$
#118CBA (17,140,186)	$12 < PM_{2.5} \leq 35$
#286096 (40,96,150)	$35 < PM_{2.5} \leq 55$
#8659A5 (134,89,165)	$55 < PM_{2.5} \leq 75$
#6A367A (106,54,122)	$PM_{2.5} > 75$

data from the SGSC (listed in SI Table S2) can be accessed, loaded into R, and visualized used pre-built plotting functions. These plotting functions allow the user to create useful and interactive plots that can be shared within a community group and deliver actionable information for the community members to answer questions like "When is particle pollution highest in my community?" and "What time of day or day of week would be best to plan an outdoor activity (i.e. walk dog or golf game) to potentially reduce my particle pollution exposure?" *AirSensor* creates a data flow for the end-user to create data objects for synoptic data, time-series data, and QC hourly $PM_{2.5}$ data. With the functions of this R package highlighted in the methods section, the user can easily create informative plots for community members to understand their local historical air quality trends with minimal coding required. The *AirSensor* R package and associated functions provide the necessary back-end software analysis and plotting functions to create the front-end *DataViewer* web application. The DataViewer is usable and useful to a much broader segment of the public and is the primary point of interaction for community members to gain insights into their local air quality conditions. This solution provides an example of how these types of tools and solutions can enhance public engagement with data from LCS networks.

3.2. *DataViewer application*

3.2.1. *User interface: Tabular structure and plotting features*

The *DataViewer* application, version 0.9.7, has a hierarchical page and tab structure with 4 top-level pages: Explore, View Data, Latest Data, and About. The View Data page is for viewing tabled data and provides the ability to download data in 3 to 30-day intervals on a per sensor basis. For SGSC, historical data can be accessed back to the start of the SGSC deployments: October 01, 2017. The View Data page includes high resolution (2-min) time-matched PA-II $PM_{2.5}$ data from the A and B sensor channels, temperature (°F), and relative humidity (%). This data output provides the user with a clean time-matched data set for the A and B sensor. Creating a similar data set outside of the *AirSensor* R package or *DataViewer* application would likely be time consuming and difficult; especially if the user were not proficient with Microsoft Excel or data science environments. The Latest Data page provides visual access to the latest non-QC data on a per sensor basis with timeseries plots provided for sensor channel A, channel B, humidity, and temperature. The "About" page provides an overview of the *DataViewer*, its intended purpose, QC procedures, and a disclaimer message.

3.2.2. *Explore page: Tabs and functionality*

The Explore page has the most functionality for exploring and analyzing community AM data and includes six tabs: Overview, Calendar, Raw Data, Daily Patterns, Compare, and Timelapse. In the Overview tab, the user can select a community, a single sensor (sensor name), a date (end date), and view past data with options for viewing the prior 3, 7, 15, or 30 days to the selected end date. The Overview tab (Fig. 12) provides a map that displays the average $PM_{2.5}$ for all sensors within the selected community for the time period selected (3, 7, 15, or 30-day average) and a bar chart that displays hourly $PM_{2.5}$ concentrations for the selected sensor. This overview tab provides the user with access to historical pollutant concentrations for the user-selected timeframe for their community and individual sensor. By changing the date, the user can quickly identify spatial differences between locations since the map indicates an average $PM_{2.5}$ concentration for the entire timeframe

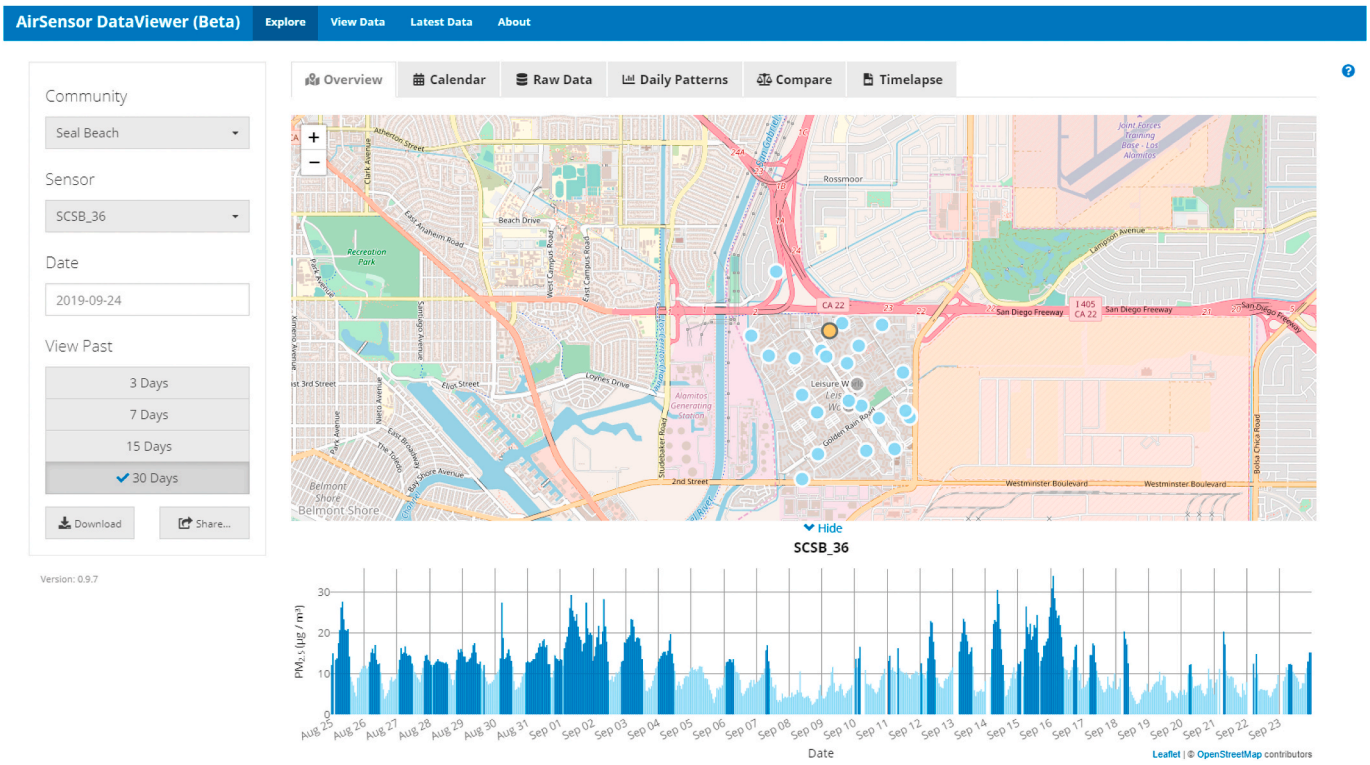


Fig. 12. Overview tab in the *DataViewer* application showing $PM_{2.5}$ concentrations and sensor locations for Seal Beach, CA.

chosen (3–30 days). Additionally, the user can quickly scan the bar chart for when higher than typical $PM_{2.5}$ concentrations were recorded for a particular sensor.

In the Calendar tab, a 1-year calendar plot is rendered for a single selected sensor. The user selects a community, sensor, and date with a calendar plot being generated for the entire calendar year of the date selected (Fig. 13). The calendar plot is interactive and when the user hovers over a date, the 24-Hr averaged $PM_{2.5}$ concentration is displayed.

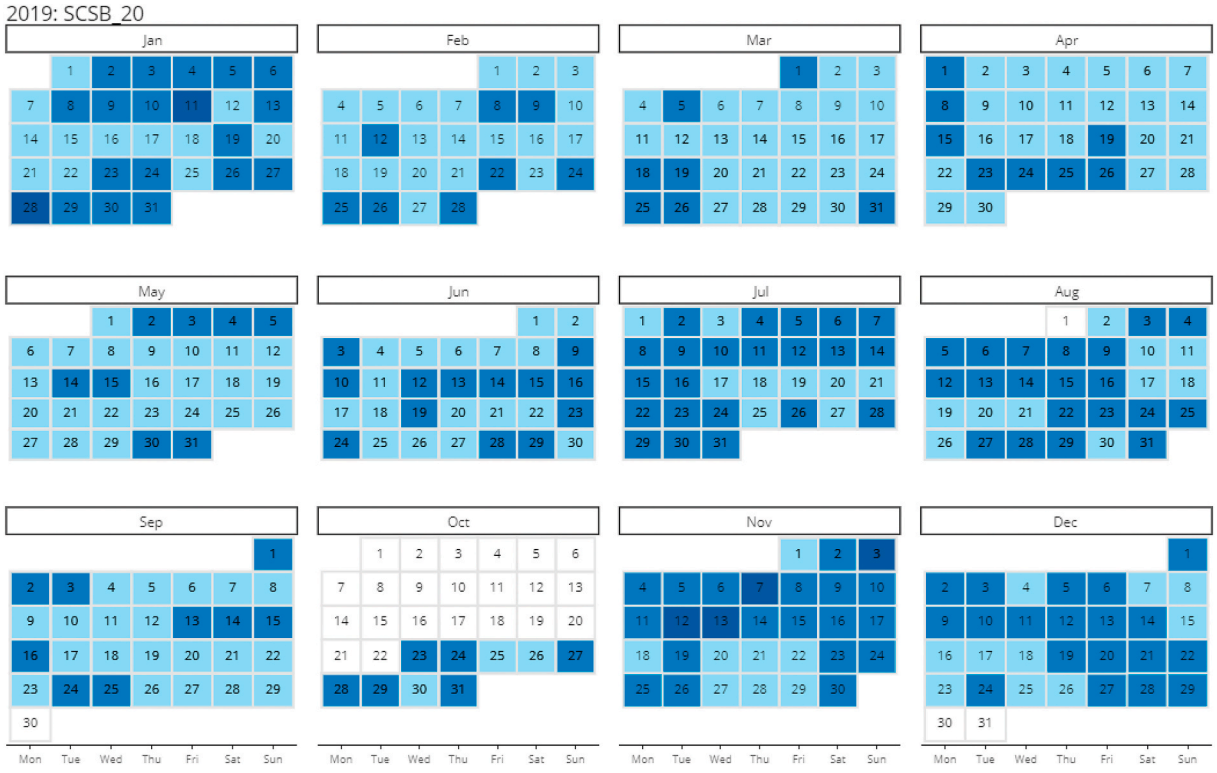


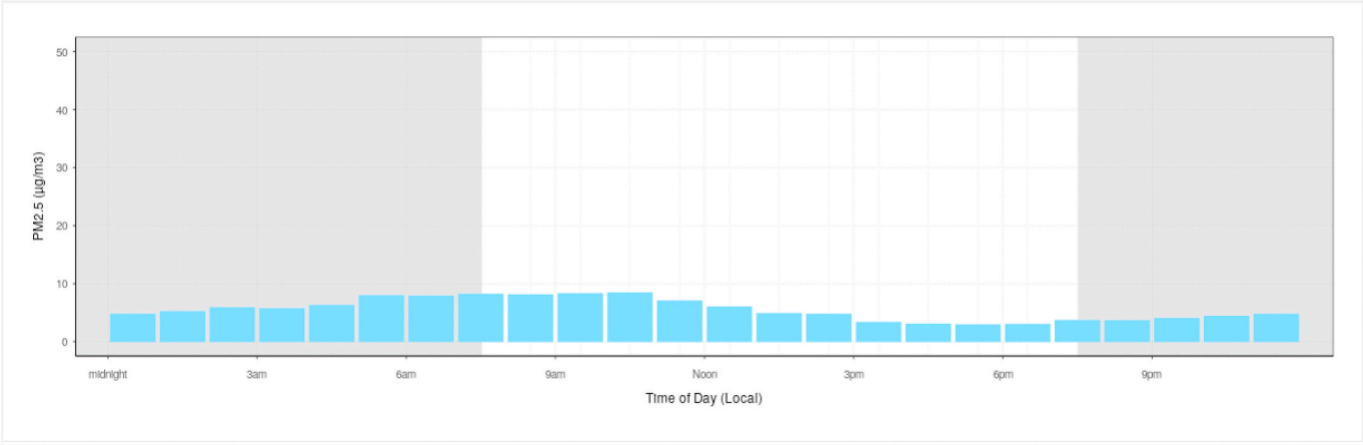
Fig. 13. Calendar Plot generated using the *AirSensor DataViewer* Application. The darker shades indicate higher levels of pollution as set forth in the color scale provided in Table 1. (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

The calendar plot is easily understood by community members and provides an intuitive view of a complete year of $PM_{2.5}$ data for a single sensor. The calendar tab is great place to start when exploring a community data set to find dates with atypical 24-hr $PM_{2.5}$ concentrations. The user can then further examine these atypical pollution events at higher time resolution with other tabs available within the Explore page. The calendar plot especially resonated with the community members and sensor hosts; and therefore, was a priority for inclusion in the *DataViewer* application. Calculating and rendering the calendar plot is computationally expensive and may take a few moments to display when interacting with the *DataViewer* application, but the result is well worth the wait for this informative plot. Throughout the workshops, we received the most feedback and discussion from community members when showing the calendar plot. The calendar plot triggered the audience and facilitated effective discussions during community workshops. Community members who would be more silent or could not recall as to what might have caused poor air quality in their community during the past several months, were able to identify days with poor air quality and what might have caused them when they viewed the calendar plot with the color-coded concentrations.

The Raw Data tab provides the raw time-series data for channels A and B, humidity, and temperature. Below the time series plots, the Raw Data tab provides a comparison between the channel A and B sensors with both a time-series and a scatterplot that indicates the regression statistics between the channel A and B. This functionality uses the pat_internalfit() function from the *AirSensor* R package which was previously shown in Fig. 5. These comparison plots provide the user with the ability to check on the performance of an individual sensor by viewing how well the two internal raw sensors within the PA-II agree for the selected time period. If a user is concerned with the performance of an individual sensor, this tab can be used to determine if both the raw sensors are responding to changes in particle concentrations similarly. Low correlation and/or a large slope/intercept offset are indicative of a sensor performance issue and that one or both sensors may be experiencing a malfunction.

The Daily Patterns tab (Fig. 14) provides a bar chart illustrating the diurnal trend for $PM_{2.5}$, a pollution rose, and a summary table for the NOAA weather data for the date range selected. The daily patterns bar chart provides the average concentration by hour of day. With this tab, the *DataViewer* user can determine on average what hour of the day has the highest and lowest particle pollution. This plot helps to inform users as to historical trends within their community and provides information that the community member can infer what time of day may be best for scheduling physical activity to reduce particle pollution exposure based on historical air pollution trend data. The pollution rose allows the user to determine if pollution can be attributed to specific meteorological conditions.

The Compare tab provides a comparison between the sensor data and the nearest AMS equipped with a continuous regulatory $PM_{2.5}$ instrument. The Compare tab provides a map indicating the location of the sensor and nearest AMS along with a timeseries and scatter plot comparison for the two data sources, allowing the user to determine if the selected sensor follows the typical trends for $PM_{2.5}$ recorded at the nearby regulatory AMS for the date range selected. The *DataViewer* application is using the *AirSensor* pat_externalfit() plotting function which was shown prior in Fig. 7. While the distance between the regulatory monitor and the LCS is provided on the Sensor-Monitor Comparison timeseries plot, the map provided in the *DataViewer* on this tab



Additional NOAA Weather Data

Average Wind Speed (m/s)	2.43
Minimum Wind Speed (m/s)	0.00
Maximum Wind Speed (m/s)	9.30
Average Wind Direction (deg)	211.00
Average Temperature (C)	14.40
Minimum Temperature (C)	6.70
Maximum Temperature (C)	25.00
Average Humidity (%)	67.19
Minimum Humidity (%)	6.69
Maximum Humidity (%)	94.62

Wind Rose Plot

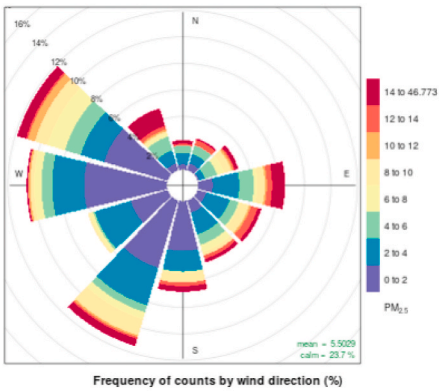


Fig. 14. Snapshot of the Daily Patterns tab in the *DataViewer* application.

allows the user the opportunity to visualize the distance between and the spatial context of the two monitoring locations. Understanding the siting of the LCS and the regulatory AMS is crucial to understanding the information provided by the comparison plot. If either the sensor or regulatory monitor is installed in a near-source environment (i.e. near-road), the user should not expect the two measurements to agree.

The final tab in the Explore page is the Timelapse tab. This tab provides the user with the ability to generate a 6-day timelapse $PM_{2.5}$ concentration video on a per community basis (Fig. 15). Right-clicking on the video allows the user to save a MP4 video to their computer and share if desired. This timelapse concentration map allows the user to view pollution events that may have taken place within a community during a selected time frame and visualize the flow of pollutants through a community. An informative approach to using this timelapse video is first to use the calendar plot feature to identify dates with elevated $PM_{2.5}$ mass concentrations ($\mu g/m^3$). After identifying those dates, the user can then choose an inclusive date range to view the community timelapse to better understand the pollution event.

4. Discussion

While online systems exist to view real-time and recently recorded measurements, FOSS tools for accessing, processing, and analyzing historical AM data collected by LCS are less available to the public. Developing FOSS tools for archiving, interpreting, and communicating data from sensors has been identified as a concrete next step towards building a system for filling the air quality data gap (Pinder et al., 2019). This work provides a FOSS R package and a web application designed to fill that gap by providing the software tools to view both real-time and historical hyper-local air quality information generated by LCS networks. Access to hyper-local air quality information is expected to spawn an increased desire to interact with air quality information and allow community members to take appropriate actions based on results generated from their community monitoring networks. The *AirSensor* R package and *DataViewer* application provide a framework and data flow for communities to transform their community monitoring data sets into insightful information through interactive data experiences and data explorations. When meaningful results and observations are formulated, community members can take appropriate actions to reduce their exposure to air pollutants. These actions could include planning transportation (e.g., walk, bike, motor vehicle) routes to reduce air pollution exposure and scheduling physical activity events (e.g., golf game, sporting practice, sporting event) during hours of the day or day of the week that have been identified to have lower $PM_{2.5}$ pollution based on historical data analysis. Our experience with sharing the *DataViewer* with the community leaders and members participating in the project has been positive with users enjoying the interactive data experience provided within the *DataViewer*. These community members have shared how this *DataViewer* provides them with the analysis capabilities to better understand their local air quality conditions. Plots that previously seemed out of reach due to required technical data analysis skills and coding experience are now readily available and generated with only a few selections and mouse clicks within the *DataViewer* application.

FOSS software developments provide efficiency by building a community of proactive data users around shared tools and allowing for multiple parties (i.e. agencies, entities, individuals) to contribute to software development and enhancing software functionalities. This benefit has already been realized as with the USFS AirFire group funding further developments to *AirSensor* for functions to calculate state-of-health metrics designed to categorize whether sensors are functioning properly. This information will be used in the context of wildfire air quality response. FOSS allows for researchers to collaborate and build upon the foundation established in this development. FOSS developments can also provide a high level of transparency in terms of data analysis and integrity as the end-user is able to select which post-processing steps are appropriate for their data analysis. With FOSS tools and publicly available data sets, researchers can reproduce data analysis techniques and develop additional functions with the interoperability associated with FOSS development.

5. Conclusions

This novel work brings these software systems to the end-users or community members in a FOSS format with all the advantages of open software developments. Not only is the end-user able to access, process,

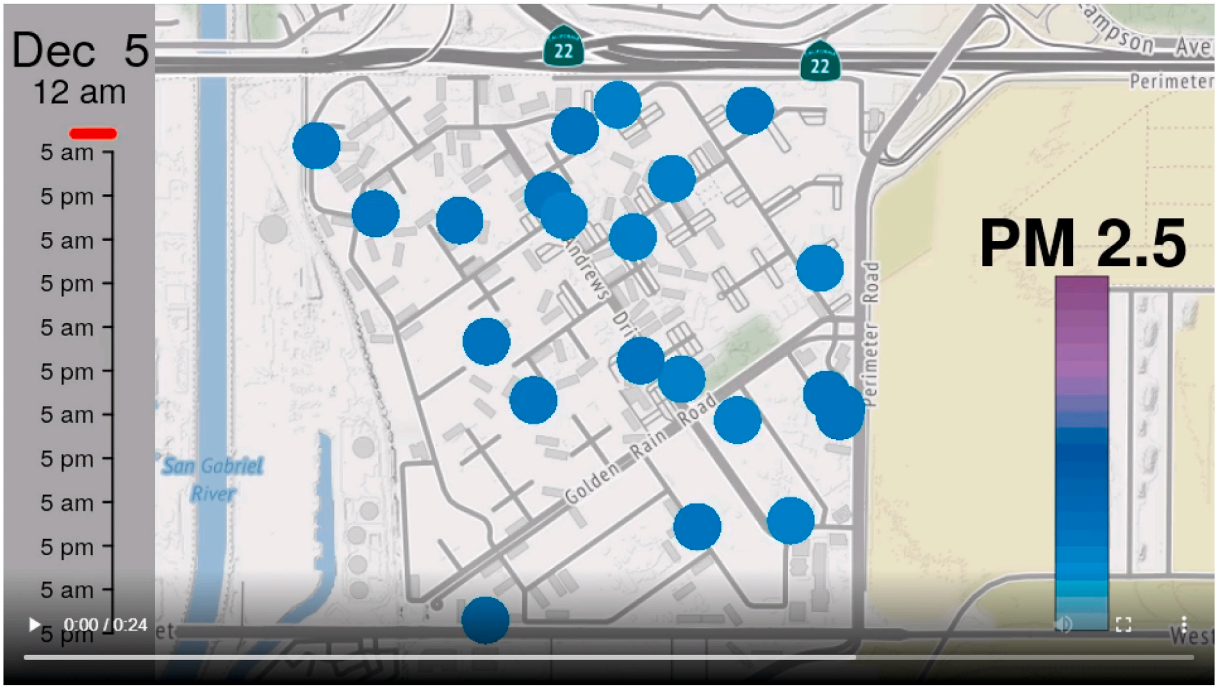


Fig. 15. Community timelapse video tab.

and analyze historical sensor data, but the user also has access to the source code and functions with the option to create their own custom functions for QC, filters, and advanced analytics. Allowing the community to build upon this existing work provides benefits to the sensing community as a whole. Developing this software in the R-environment also provides for data fusion enrichment by coupling the collected AM sensor data with meteorological data and regulatory AM data through other open-source packages in the R environment. The *AirSensor* package has established a foundation upon which further enhancements and refinements can be developed. Both *AirSensor* and *DataViewer* source codes are available on Github and the authors invite collaboration and input to help shape the *AirSensor* open source project to best meet the needs of the air sensing community.

The *AirSensor* R package is sensor specific, working with any publicly registered Purple Air PA-II sensors. The *DataViewer* solution is both sensor- and project-specific and therefore limited to the PA-II sensors deployed by South Coast AQMD in SGSC. The authors believe that the data flow works well for AM sensor data with the data objects going from synoptic data to time-series data and then to hourly QC sensor data. The blueprint developed to make the *DataViewer* operational could be applied to other projects and communities to visualize data collected by their PurpleAir LCS networks. The work discussed in this paper focused on the initial data handling and analysis capabilities required for a community AM network of $PM_{2.5}$ sensors. Planned future work will focus on several improvements to the AirSensor R package, the data archive database design, and the *DataViewer* application. The AirSensor R package and archive will be improved by adding functionality to handle unique timeseries identifiers and incorporating PM_1 and PM_{10} data. Additional plotting functionality will include enhancements to create multi-sensor comparison plots and visualize sensor state-of-health metrics for both individual sensors and sensor networks. Additional enhancements to the R package may include developing models to provide hyper local air quality forecast for the community. The *DataViewer* will be enhanced by improving the appearance, usability, data handling, and performance of the application.

Funding

This research has been supported by a grant from the U.S. Environmental Protection Agency's Science to Achieve Results (STAR) program to the South Coast Air Quality Management District.

Disclaimer statement

This publication was developed under Assistance Agreement No. R836184 awarded by the U.S. Environmental Protection Agency to South Coast AQMD. It has not been formally reviewed by EPA. The views expressed in this document are solely those of the authors and do not necessarily reflect those of the U.S. EPA. The South Coast AQMD and U. S. EPA do not endorse any products or commercial services mentioned in this publication.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors would like to thank Dr. Jonathan Callahan and Hans Martin at Mazama Science, Inc. (Seattle, WA) for their collaboration and contributions in the development of the *AirSensor* R-package and the *DataViewer* application tools along with their valuable feedback on this manuscript. The sensor data used and presented in this paper was collected by the Air Quality Sensor Performance Evaluation Center (AQ-SPEC) at South Coast AQMD. The authors would also like to thank the community groups' leaders, trainers, coordinators, and members/sensor hosts that participated in the U.S. EPA STAR grant and provided valuable feedback that allowed us to create and improve this work. The authors thank Ms. Emma Ranheim who assisted in user testing the *AirSensor* R package.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.envsoft.2020.104832.

References

Air Fire Tools, 2020. WFAQRP-AirFire tools information. Viewed 02/05/2020, from. https://tools.airfire.org.
Air Quality Egg, 2020. Air quality Egg portal. Viewed 02/05/2020, from. https://airqualityegg.com/portal/.
AirNow, 2020. "AirNow developer tools." AirNow API. Viewed 02/04/2020, from. https://docs.airnowapi.org/.
Allaire, J., 2020. RStudio, PBC. https://blog.rstudio.com/2020/01/29/rstudio-pbc/2020.
BreezoMeter, 2020. Air quality map. Viewed 02/04/2020, from. https://breezometer.com/air-quality-map.
Callahan, J., Aras, R., Dingels, Z., Hagg, J., Kim, J., Martin, H., Miller, H., Pease, S., Thompson, R., Yang, A., 2019. PWFSLSmoke: Utilities for Working with Air Quality Monitoring Data. R package version 1.2.103, from. https://github.com/MazamaScience/PWFSLSmoke.
Carslaw, D., 2019. Worldmet: Import Surface Meteorological Data from NOAA Integrated Surface Database (ISD). R package version 0.8.7, from. http://github.com/davidcarslaw/worldmet.
Carslaw, D.C., Beevers, S.D., 2013. Characterising and understanding emission sources using bivariate polar plots and k-means clustering. Environ. Model. Software 40, 325–329.
Carslaw, D.C., Ropkins, K., 2012. Openair - an R package for air quality data analysis. Environ. Model. Software 27–28, 52–61.
Conway, D., 2013. The data science venn diagram. Viewed 05/31/19, from. http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram.
Dawes, B., 2019. Using FFmpeg to convert image sequences to video. Analogue + Digital Viewed 12/27/19, from. http://brendandawes.com/blog/ffmpeg-images-to-video.
Feenstra, B., Papapostolou, V., Hasheminassab, S., Zhang, H., Boghossian, B.D., Cocker, D., Polidori, A., 2019. Performance evaluation of twelve low-cost PM2.5 sensors at an ambient air monitoring site. Atmos. Environ. 216, 116946.
FFmpeg, 2019. FFmpeg. Viewed 12/27/19, from. https://www.ffmpeg.org/about.html.
Gibert, K., Horsburgh, J.S., Athanasiadis, I.N., Holmes, G., 2018. Environmental data science. Environ. Model. Software 106, 4–12.
Grange, S.K., Lewis, A.C., Carslaw, D.C., 2016. Source apportionment advances using polar plots of bivariate correlation and regression statistics. Atmos. Environ. 145, 128–134.
HabitatMap, 2020. AirCasting map. Viewed 02/05/2020, from. http://aircasting.habitatmap.org/mobile_map.
Hagler, G.S.W., Williams, R., Papapostolou, V., Polidori, A., 2018. Air quality sensors and data adjustment algorithms: when is it No longer a measurement? Environ. Sci. Technol. 52 (10), 5530–5531.
IQAir, 2020. AirVisual map. from. https://www.airvisual.com/air-quality-map.
Kadiyala, A., Kumar, A., 2017a. Applications of Python to evaluate environmental data science problems. Environ. Prog. Sustain. Energy 36 (6), 1580–1586.
Kadiyala, A., Kumar, A., 2017b. Applications of R to evaluate environmental data science problems. Environ. Prog. Sustain. Energy 36 (5), 1358–1364.
Luftdaten, 2020. Measuring air data with citizen science. Viewed 02/05/2020, from. https://luftdaten.info/en/home-en/.
Magi, B.I., Cupini, C., Francis, J., Green, M., Hauser, C., 2019. Evaluation of PM2.5 measured in an urban setting using a low-cost optical particle counter and a Federal Equivalent Method Beta Attenuation Monitor. Aerosol. Sci. Technol. 13.
Mintz, D., Stone, S., Dickerson, P., Davis, A., 2013. Transitioning to a New NowCast Method. Viewed 11/21/19, from. https://www3.epa.gov/airnow/ani/pm25_aqi_reporting_nowcast_overview.pdf.
Ok Lab Stuttgart, 2020. Open data Stuttgart. Viewed 02/05/2020, from. www.github.com/opendata-stuttgart.
OpenAQ, 2020. Open data: countries. Viewed 01/24/2020, from. https://openaq.org/#/countries.
Pinder, R.W., Klopp, J.M., Kleiman, G., Hagler, G.S.W., Awe, Y., Terry, S., 2019. Opportunities and challenges for filling the air quality data gap in low- and middle-income countries. Atmos. Environ. 215, 116794.
Plume Labs, 2020. Air quality map. Viewed 02/04/2020, from. https://air.plumelabs.com/air-quality-map.
PurpleAir, 2020. PurpleAir map. Viewed 02/05/2020, from. https://www.purpleair.com/map.
RStudio, 2019. R studio. Viewed 11/08/2019, from. https://rstudio.com/.
Sandhaus, S., Kaufmann, D., Ramirez-Andreotta, M., 2019. Public participation, trust and data sharing: gardens as hubs for citizen science and environmental health literacy efforts. Int. J. Sci. Educ. Part B-Communication and Public Engagement 9 (1), 54–71.

Smart Citizen Kit, 2020. Smart Citizen Kit Map, 02/05/2020, from https://smartcitizen.me/kits/.
Snyder, E.G., Watkins, T.H., Solomon, P.A., Thoma, E.D., Williams, R.W., Hagler, G.S.W., Shelow, D., Hindin, D.A., Kilaru, V.J., Preuss, P.W., 2013. The changing paradigm of air pollution monitoring. Environ. Sci. Technol. 47 (20), 11369–11377.
The R environment, 2019. The R environment. Viewed 11/08/2019, 2019, from. https://www.r-project.org/about.html.
U.S. Environmental Protection Agency, 2019. Air sensor Toolbox: what do my sensor readings mean? Sensor scale pilot project. Viewed 11/21/2019, from. https://www.epa.gov/air-sensor-toolbox/what-do-my-sensor-readings-mean-sensor-scale-pilot-project.
uRADMonitor, 2020. Global Environmental Monitoring Network. Viewed 02/05/2020, from. https://www.uradmonitor.com/.
World Air Quality Index Project, 2020. World's air pollution: real-time air quality Index. Viewed 01/24/2020, 2020, from. https://waqi.info/.