Background

The Methow Valley in north central Washington was heavily impacted by wildfire smoke in September of 2020 as a result of two Washington wildfires in grass and steppe and the colossal wildfires in the Oregon Cascades. In Washington, the Cold Springs and Pearl Hill fires burned over 410,000 acres throughout the month. In Oregon, wildfires began on Labor Day and created a massive plume of smoke that was carried to Washington State by southerly winds. Eventually the smoke settled in the valleys and basins of Washington and air pollution reached unhealthy levels across the state (https://wasmoke.blogspot.com/2020/09/).

Since 2018, the Methow Valley Clean Air Ambassador (MVCAA) program has deployed low-cost air quality sensors in the valley to monitor conditions. Mazama Science has just released a set of MVCAA Tutorials — RMarkdown documents that demonstrate how to get the most out of PurpleAir low-cost sensor data and nearby regulatory monitoring data using a dedicated suite of air quality R packages: PWFSLSmoke, AirSensor and AirMonitorPlots.

The MVCAA Tutorials show how to create straightforward and explanatory plots for air quality monitoring data and is meant for grad students, researchers and any member of the public concerned about air quality and comfortable working with R and RStudio.

R Packages

PWFSLSmoke

The PWFSLSmoke R package was developed with funding from the US Forest Service AirFire research team to help modelers, scientists and air quality professionals more easily work with PM2.5 data from monitoring locations across North America.

The package makes it easier to obtain data, perform analyses and generate reports. It includes functionality to:

- download and easily work with regulatory PM2.5 data from the EPA and AirNow
- download and quality control raw monitoring data from AIRSIS and WRCC
- convert between UTC and local timezones
- apply various algorithms to the data (nowcast, rolling means, aggregation, *etc.*)
- provide interactive timeseries and maps through RStudio's Viewer pane
- create a variety of publication ready maps and timeseries plots.

Visit the Reference webpage to get familiar with the **PWFSLSmoke** function catalogue.

AirSensor

The AirSensor R package was developed with funding from the South Coast Air Quality Management District AQ-SPEC program to help air quality analysts, scientists and interested members of the public more easily work with air quality data from consumer-grade air quality sensors. Initial focus is on PM2.5 measurements from sensors produced by PurpleAir.

The package makes it easier to obtain data, perform analyses and create visualizations. It includes functionality to:

- download and easily work with PM2.5 data from PurpleAir
- visualize raw "engineering-level" data from a PurpleAir sensor
- visualize data quality using built-in analytics and plots

- aggregate raw data onto an hourly axis
- create interactive maps and time series plots
- convert aggregated PurpleAir data into sensor objects of class `ws_monitor` appropriate for use with the PWFSLSmoke package.

Visit the Reference webpage to get familiar with the **AirSensor** function catalogue.

AirMonitorPlots

The AirMonitorPlots R package provides plotting functionality to create production-ready plots for air quality monitoring data. It builds on the **PWFLSmoke** package, integrating the data with **ggplot2** plotting functions. High-level plotting functions make it easy for users to create beautiful plots of monitoring data. Since these functions are built on **ggplot2**, users familiar with **ggplot2** can use the functions in this package to create custom plots.

Visit the Reference webpage to get familiar with the **AirMonitorPlots** function catalogue.

Tutorials

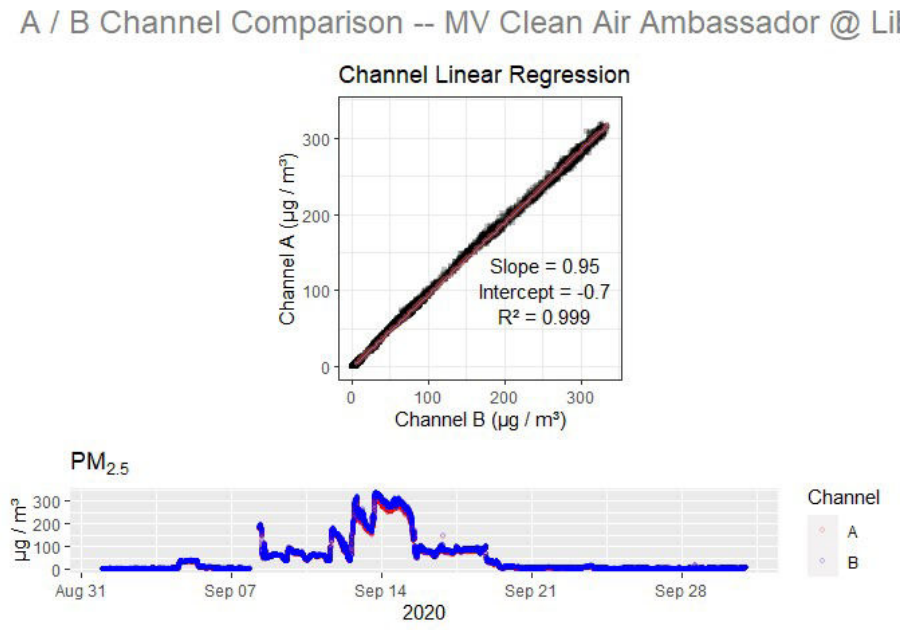
Tutorial 1: Creating PAT Data

The Creating PAT Data tutorial covers the creation of Purple Air Synoptic `pas` and Purple Air Timeseries `pat` objects for a single month for the Methow Valley. After creation, data are saved to a local directory for easy reloading.

Tutorial 2: Exploring PAT

Exploring PAT Data, as the name suggests, explores a variety of `pat_~()` functions available in the **AirSensor** package and explains their most important arguments and the interpretation of output graphics.

One useful function for quickly assessing a sensor's internal performance is `pat_internalFit()`, which uses a linear model to fit data from two internal data channels (A and B) and which creates a fit and a time-series plot to help interpret the results. As we can see, the Balky Hill sensor performed very well!



Tutorial 3: Creating Airsensor Data

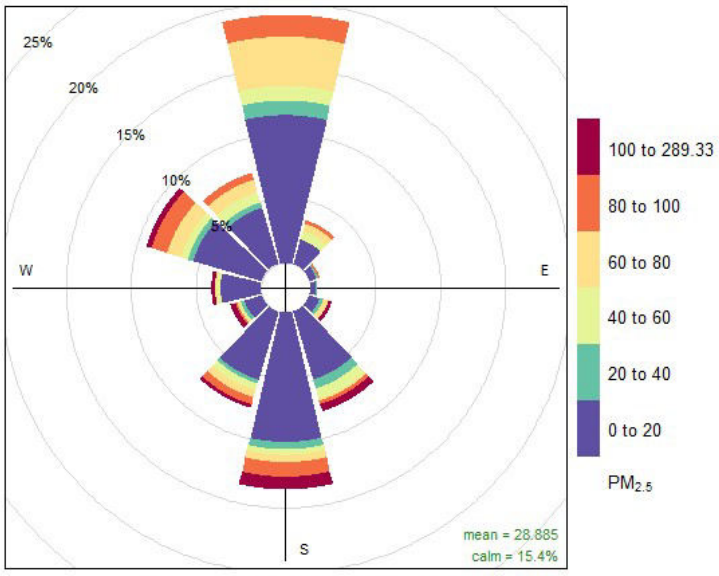
The Creating Airsensor Data tutorial walks through the process of loading previously generated 'pat' data, converting it into an hourly-aggregated, multi-sensor `airsensor` object, and saving it to a local directory. Aggregation of raw data to quality controlled, hourly averages allows sensor data to be compared with meteorological and air quality data from other sources. Hourly aggregation is the *de facto* standard for most air quality analysis

Tutorial 4: Exploring Airsensor Data

Having created hourly `airsensor` objects, Exploring Airsensor Data demonstrates how to use a variety of `sensor_~()` functions available in the **AirSensor** package and explains their most important arguments and the interpretation of output graphics.

One of the first questions often asked about air pollution is: "Where is it coming from?" Functions in the **AirSensor** package can help answer that.

The `sensor_pollutionRose(statistic = "prop.count")` function shows you which wind direction is associated with highest PM2.5 concentrations (plot below). Curious about which wind direction is instead carrying the *largest proportion* of all PM2.5 concetrations? Tutorial 4 has the answer! Check it out!



Frequency of counts by wind direction (%)

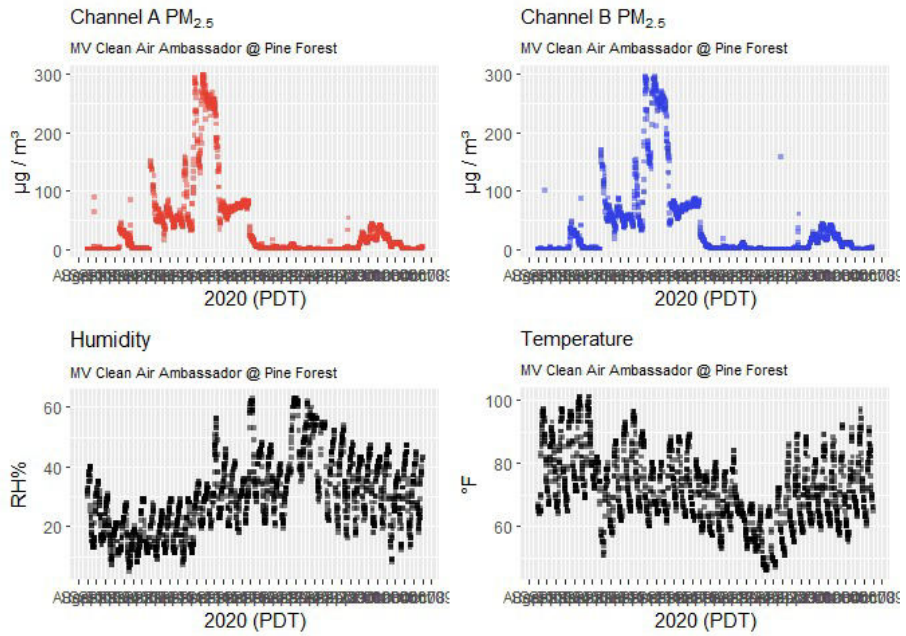
Tutorial 5: Building a Local Archive

The goal in Building a Local Archive is to explain the file naming protocol and directory structure required for data stored locally to be loaded using the `pat_load()` and `sensor_load()` functions. In this tutorial we build a multi-month archive for the Methow Valley and demonstrate accessing that data using package functions.

The code below shows how easy it will be to load `pat` data from your local archive and run your analyses offline after you complete this tutorial!

```
library(AirSensor)

# Load specific days of pat data for the Pine Forest sensor
Pine_Forest  <-
```

```
  pat_load(
    id = "0cbfeb2ce4c1553c_13661",
    pas = mvcaa,
    startdate = 20200901,
    enddate = 20201008,
    timezone = "America/Los_Angeles"
  )

# Basic plot for Pine_Forest
pat_multiplot(Pine_Forest)
```



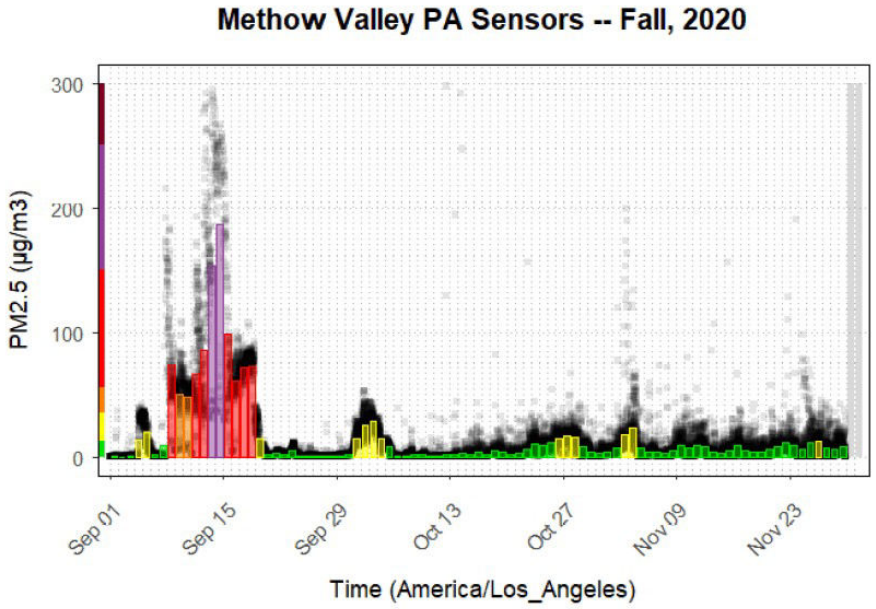
Tutorial 6: Methow Vallely Smoke

The final tutorial in the series is Methow Valley Smoke. Our goal here is to demonstrate how to evaluate the performance of low-cost sensors under very high PM25 concentrations due to wildfire smoke. We will do some basic data exploration and evaluate several of the MVCAA sensors, comparing their data with data from a nearby regulatory monitor.

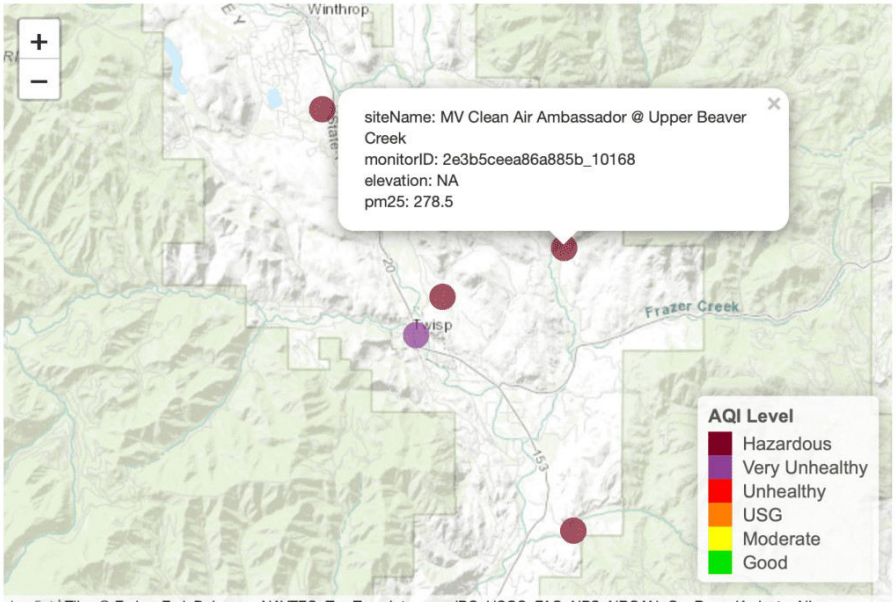
All great analysis should start with a great plot! This beautiful time-series plot, created with the **AirMonitorPlots** and **ggplot2** packages, shows you every hourly value (light black squares) and combined daily averages (colored bars) for all MVCAA sensors. Also notice the AQI color-coded bar along the y-axis. This plot has it all and it takes only a few lines of code!

```
# libraries
library(AirSensor)
library(PWFSLSmoke)
library(AirMonitorPlots)
library(ggplot2)

# Set the package archiveBaseDir to our local archive
setArchiveBaseDir("~/Data/MVCAA")

# Load sensor data
mvcaa_sensors_fall <-
  sensor_load(
```

```
    collection = "mvcaa",
    startdate = 20200901,
    enddate = 20201201,
    timezone = "America/Los_Angeles"
  )

# Plot all hourly values with daily averages
all_sensors_fall %>%
  AirMonitorPlots::ggplot_pm25Timeseries() +
  ggplot2::ggtitle("Methow Valley PA Sensors -- Fall, 2020") +
  AirMonitorPlots::geom_pm25Points(shape = "square", alpha = .1) +
  AirMonitorPlots::stat_dailyAQCategory(alpha = .5) +
  ggplot2::scale_y_continuous(limits = c(0, 300)) +
  AirMonitorPlots::custom_aqiStackedBar(width = 0.01)
```



Finally, no analysis of air quality data is complete without some nice maps. The **AirSensor** and **PWFSLSmoke** packages provides provide interactive, leaflet maps to show where monitors and sensors are located and what they are measuring. To fully understand air quality it is important to understand local topography, especially in mountainous terrain.


Winthrop
siteName: MV Clean Air Ambassador @ Upper Beaver Creek
monitorID: 2e3b5ceea86a885b_10168
elevation: NA
pm25: 278.5
Frazer Creek
Twisp
AQI Level
Hazardous
Very Unhealthy
Unhealthy
USG
Moderate
Good