**Methods**

**Github:** [**https://github.com/catti2d/marshall\_fires**](https://github.com/catti2d/marshall_fires)

**purple\_air\_extraction.Rmd**

To begin, we download the Purple Air sensor data based on GIS data of the areas surrounding the fires. The areas were based on Boulder voter precincts, Broomfield voter precincts, and the city of Westminster. We use the ‘bjzresc’ package to download a list of all of the sensors, which we first filter by intersecting the locations of the sensors with the municipal boundaries. After getting the sensors only within our area of interest, this package downloads the data for these specific sensors between 12/30/2021 and 5/1/2022. The sensors are each downloaded to an individual file, which we then read into R and combine to get one large data frame of all the Purple Air sensors.

To clean the data, we transform the time ‘created\_at’ column into POSIX format so that R will treat the data as a timeseries. Some columns are renamed as follows: ‘PM2.5\_CF\_ATM\_ug/m3\_A’ to ‘pm25\_a’, ‘PM2.5\_CF\_ATM\_ug/m3\_B’ to ‘pm25\_b’, ‘Temperature\_F\_A’ to ‘temp’, and ‘Humidity\_%\_A’ to ‘rh’. Any missing data for every sensor is made ‘NA’ to better represent times that have no pm2.5 data. The sensors are geocoded to get addresses, city, and zip codes for each sensor.

The data are then classified with a ‘fire\_period’ column where any times before 2021-12-30 10:00:00 is “pre\_fire\_period”, between that and 2022-01-01 11:59:59 is “fire\_period”, and anything afterwards is “post\_fire\_period”. For each of these periods, the number of non-NA values are calculated to determine a new ‘Status’ column. If there is >75% complete values during the fire period, >85% complete values after the fire, and >95% complete values before the fire, the sensor is classified as having ‘Complete data through the fire period.’ If there is less than 75% complete values during the fire period, but the other two conditions hold, the sensor is classified as ‘Sensor offline during fire, returned online.’ If there are less than 75% complete values after the fire, the sensor is classified as ‘Sensor offline during fire, did not return online’. If there are less than 75% complete values during the fire and after the fire, and no values from before the fire, the sensor is classified as ‘Sensor added during fire, did not return online’. With the same conditions as the last category, but more than 75% complete values after the fire, the sensor is classified as ‘Sensor added during the fire, returned online.’ The last category is if both the fire period and pre fire period have no values, the sensor is then classified as ‘Sensor came online after fire’.

A data frame is created that contains information for each individual sensor, including the information from Purple Air (such as ID number, name, etc.), as well as the geocoded address, month it came online, and the status as classified above. Both the air quality data frame and the sensor information data frame are exported to individual csv files.

**clean\_AQ.Rmd**

This R script starts by reading in the Purple Air data that was downloaded in the previous script. The data are cleaned according to the EPA fire correction equation (<https://fire.airnow.gov/#correction-equation>). A data point is valid if the A and B channel PM2.5 measurements are within either 5 micrograms per cubic meter or 70% relative percent difference. A new column is created called “a\_b\_agree” which holds either “agree” if the data point is valid, or “disagree” if the cleaning equation states that it is invalid. We then calculate some summary statistics and visualize them for the number of points that agree or disagree per sensor.

All of the geographic information is read in about the counties/areas around the fire, the fire boundary, and the sensors. The area geometries are plotted with the sensors on top to visualize the distribution of sensors. Then, the same plot is created, but with the sensors colored by the percentage of points where the two channels agree.

The sensors where there is at least one point that disagrees during the fire period (12/30/21 – 1/2/22) to visualize where the sensors with worse data are located. 29.88% of data during the fire period is lost when the EPA cleaning formula is used on the data. Some plots are created to look at how the percentage of disagreeing points change over time, to see if it’s possible that the fire influenced any of the differences in channels that we see.

Summary statistics are calculated for each channel that show the number of NA values and then the percentage of valid points according to the EPA cleaning method.

After all of the cleaning & summary statistics for the cleaning method, the EPA fire correction (<https://fire.airnow.gov/#correction-equation>) is applied to all the sensors. Summary stats about the difference between the A and B channels are calculated to see how much the two channels vary in general. Finally, the cleaned & corrected data is exported to a csv file called corrected\_AQ\_data.csv in the intermediary\_outputs folder.

**visualization.Rmd**

All of the air quality and sensor data is imported, along with the geographic data. For the geographic borders, there are a lot of unnecessary columns that are dropped, as well as conversion into the same CRS as the Boulder geographic data. The destroyed & damaged business and home data are read into the script & made to match the CRS of the other geographic data. The sensors have a column called “loc\_status” which combines the classifications created in the first fire with the location of the sensors (whether it’s indoor or outdoor).

Leaflet is used to map the sensor locations along with the destroyed/damaged buildings. There are specific markers for the sensors versus the buildings, and different color palettes for each type of data. One map doesn’t show the data on when sensors came on/offline, which is then added on the second map.

Spatial objects are created with the sp and spacetime packages, which are then transformed into spatiotemporal objects with the STFDF package. Each row in this data frame is one sensor at one point in time, with the temporal resolution being 10 minutes. This data are then mapped to visually check that the data hasn’t been altered from the leaflet maps. The stplot function is used to create spatiotemporal timeseries for the relative humidity, temperature, and pm2.5 data. Each sensor has a different line on the plot for the entire time range that we have data for. The “xt” argument changes these plots to be space-time plots which show spatial data in addition to temporal data for all of the sensors.

All of the data is narrowed down to only sensors that have complete data throughout the fire period, then plotted in time series format.

The “Kriging” section of the script starts with timeseries plots for 4 different sensors which were showing up weirdly when the original variograms were created. These were created to see if the data for these sensors might be thrown off, and something we should drop. The “krig\_data” object is created from the spatiotemporal dataframe (STFDF) by selecting all of the columns and sensors for only the time period between 12/30/2021 and 1/2/2022. These data are then limited to exclude indoor sensors, and two of the bad sensors that was investigated is dropped because PurpleAir says there is 0% confidence in its data. Additionally, one sensor has some values >1000, probably due to snow, so all of those values over 1,000 are dropped as well. This data is then reprojected to utm and aggregated into hour time stamps by taking the mean from the 10 minute temporal resolution for each sensor.

For kriging, the data is then split into different dataframes for each day in the fire time period. From this, we then split the day of December 30th into 3 periods, before the fire (midnight to 11am), during the fire (11am to 5pm), and the evening of the fire (5pm to midnight). The 6 periods we end up with for kriging are: pre-fire, during the fire, evening of the fire, December 31st, January 1st, and January 2nd.

We create variograms for each of these time periods, along with variogram clouds, to see if kriging might be possible. We then try to fit a sample variogram to the experimental variograms for each of the time periods, however most of them do not have a good fit, meaning we won’t be able to krige the data. We try to split the pre-fire data into smaller, 1 hour blocks to see if a finer temporal resolution will result in better variograms, however there isn’t a great result from this.

A grid is created so that we can interpolate values between the sensors and get a better idea of how air quality changed around the fire. Inverse Distance Weighting interpolation is done for each of the six periods and mapped.

Leave-one-out cross validation is then done where each sensor is left out once when constructing the IDW, to get residuals for all of the sensors and their predictions in our IDW models. These residuals are used to calculate z-scores, which are mapped to show a consistent scale where error can be compared between different time periods.

There were some weird values when first calculating the residuals, so after the IDW plotting we look at some of the specific sensors which fed back into when we made the krige dataset.

The IDW process is repeated on an hourly timescale for the fire period (instead of using the 6 pre-defined periods), and the plots are then turned into gifs for easy visualization. This is also done for the entire range of data, except on a daily time scale, so that each day has a fully interpolated map to view.