Wideband Integrated Bioaerosol Sensor (WIBS) Data Analysis Project

Jeff Peterson

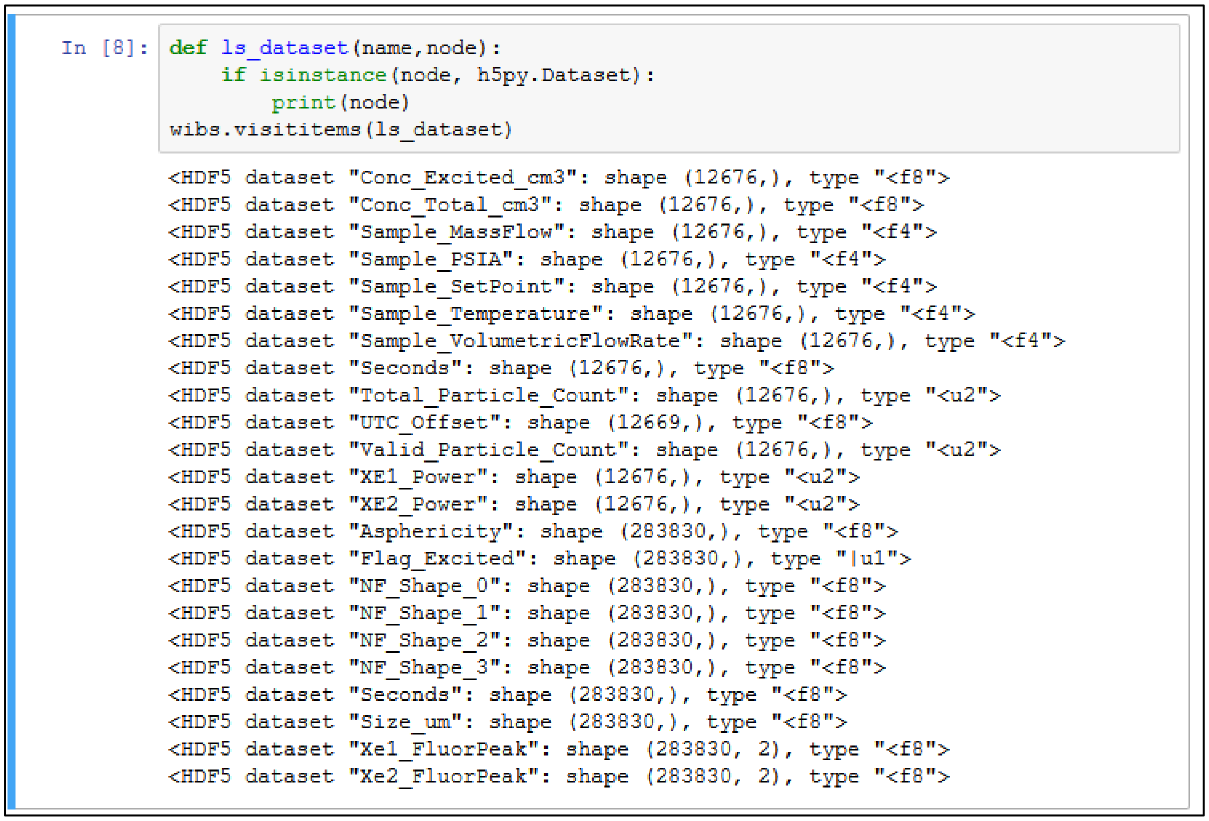
BACKGROUND

This project will focus on developing a data extraction, analysis, and plotting method for the System X test currently underway at my work. This particular test uses various instruments to measure signal responses and particle sizes throughout the trial. Historically, we have used custom software developed by the Wideband Integrated Bioaerosol Sensor (WIBS) instrument manufacturer to record, analyze, and plot the data and required calculations.[1] Unfortunately, due to increased network security issues, we are no longer able to use this software on a government computer system.[2] A short term solution was to have the WIBS manufacturer analyze our data for us. This is not a cost effective nor viable solution for the long term. Therefore, this project will focus on developing a Jupyter Notebook script to perform the necessary calculations and plots for the System X test. This report will also act as a user's guide to the Jupyter Notebook so that future users can understand the calculations and included code.

LOADING REQUIRED FILES

There are six files that are required for each test. A description of each data file and the information required from each is described below.

1. WIBS File: this is the main file recorded from the instrument during the test. It uses the h5 data format, which does not load the data directly into a DataFrame for viewing.[3-4] Instead, the data is loaded based on the types of data within the file. Figure 1 shows the screenshot of the WIBS file once it was loaded into the Jupyter Notebook.



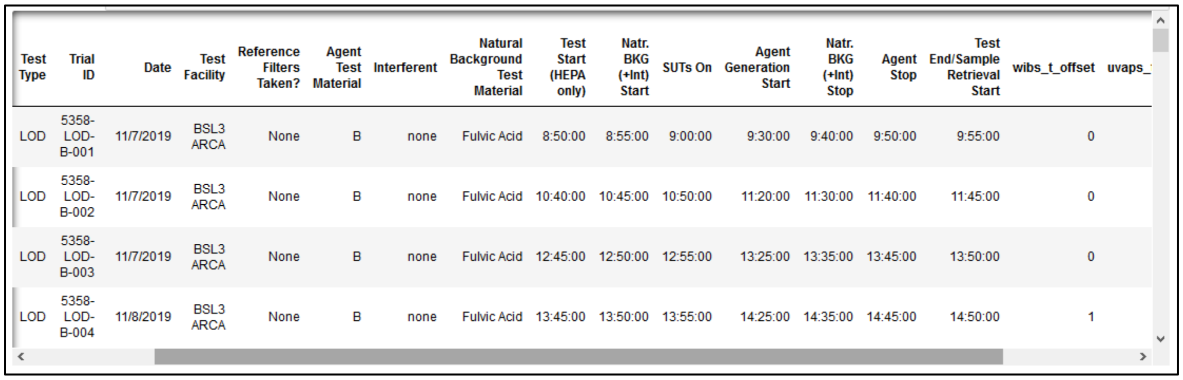
```
In [8]: def ls_dataset(name,node):
            if isinstance(node, h5py.Dataset):
                print(node)
        wibs.visititems(ls_dataset)
```

```
<HDF5 dataset "Conc_Excited_cm3": shape (12676,), type "<f8">
<HDF5 dataset "Conc_Total_cm3": shape (12676,), type "<f8">
<HDF5 dataset "Sample_MassFlow": shape (12676,), type "<f4">
<HDF5 dataset "Sample_PSIA": shape (12676,), type "<f4">
<HDF5 dataset "Sample_SetPoint": shape (12676,), type "<f4">
<HDF5 dataset "Sample_Temperature": shape (12676,), type "<f4">
<HDF5 dataset "Sample_VolumetricFlowRate": shape (12676,), type "<f4">
<HDF5 dataset "Seconds": shape (12676,), type "<f8">
<HDF5 dataset "Total_Particle_Count": shape (12676,), type "<u2">
<HDF5 dataset "UTC_Offset": shape (12669,), type "<f8">
<HDF5 dataset "Valid_Particle_Count": shape (12676,), type "<u2">
<HDF5 dataset "XE1_Power": shape (12676,), type "<u2">
<HDF5 dataset "XE2_Power": shape (12676,), type "<u2">
<HDF5 dataset "Asphericity": shape (283830,), type "<f8">
<HDF5 dataset "Flag_Excited": shape (283830,), type "|u1">
<HDF5 dataset "NF_Shape_0": shape (283830,), type "<f8">
<HDF5 dataset "NF_Shape_1": shape (283830,), type "<f8">
<HDF5 dataset "NF_Shape_2": shape (283830,), type "<f8">
<HDF5 dataset "NF_Shape_3": shape (283830,), type "<f8">
<HDF5 dataset "Seconds": shape (283830,), type "<f8">
<HDF5 dataset "Size_um": shape (283830,), type "<f8">
<HDF5 dataset "Xe1_FluorPeak": shape (283830, 2), type "<f8">
<HDF5 dataset "Xe2_FluorPeak": shape (283830, 2), type "<f8">
```

Figure 1. Screenshot of WIBS Data File Elements

2. WIBS FT File: this is also a file produced by the WIBS instrument and is called the Force Trigger (FT) file. The FT file serves as a background measurement that is recorded before the test begins. This FT file will need to be subtracted from the recorded signals in the regular WIBS file. Also, like the regular WIBS file, the WIBS FT file is also in the h5 format and produces a similar output as Figure 1 when loaded into the Jupyter Notebook.

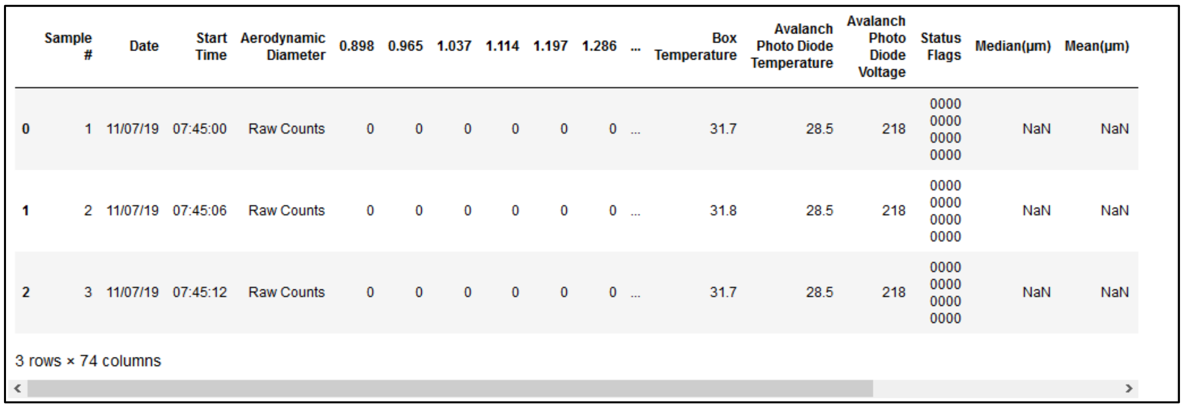
3. Test Log File: this is a file that records each of the tests conducted on a certain day. The file also contains the start and stop times for each trail, which is required for certain calculations. This test log file is a simple csv file and should be relatively easy to work with. Figure 2 shows a screenshot of the loaded test log file.



Test Type	Trial ID	Date	Test Facility	Reference Filters Taken?	Agent Test Material	Interferent	Natural Background Test Material	Test Start (HEPA only)	Natr. BKG (+Int) Start	SUTs On	Agent Generation Start	Natr. BKG (+Int) Stop	Agent Stop	Test End/Sample Retrieval Start	wibs_t_offset	uvaps_
LOD	5358-LOD-B-001	11/7/2019	BSL3 ARCA	None	B	none	Fulvic Acid	8:50:00	8:55:00	9:00:00	9:30:00	9:40:00	9:50:00	9:55:00	0	
LOD	5358-LOD-B-002	11/7/2019	BSL3 ARCA	None	B	none	Fulvic Acid	10:40:00	10:45:00	10:50:00	11:20:00	11:30:00	11:40:00	11:45:00	0	
LOD	5358-LOD-B-003	11/7/2019	BSL3 ARCA	None	B	none	Fulvic Acid	12:45:00	12:50:00	12:55:00	13:25:00	13:35:00	13:45:00	13:50:00	0	
LOD	5358-LOD-B-004	11/8/2019	BSL3 ARCA	None	B	none	Fulvic Acid	13:45:00	13:50:00	13:55:00	14:25:00	14:35:00	14:45:00	14:50:00	1	

Figure 2. Contents of Test Log File

4. APS Files: there are three Aerodynamic Particle Sizer (APS) files that are needed for each individual WIBS file. The APS instruments measure particle size diameters and are used a referee devices for the tests.[5-6] The files are in txt format. Figure 3 provides an overview of what a typical APS file looks like.



	Sample #	Date	Start Time	Aerodynamic Diameter	0.898	0.965	1.037	1.114	1.197	1.286	...	Box Temperature	Avalanch Photo Diode Temperature	Avalanch Photo Diode Voltage	Status Flags	Median(µm)	Mean(µm)
0	1	11/07/19	07:45:00	Raw Counts	0	0	0	0	0	0	...	31.7	28.5	218	0000 0000 0000 0000	NaN	NaN
1	2	11/07/19	07:45:06	Raw Counts	0	0	0	0	0	0	...	31.8	28.5	218	0000 0000 0000 0000	NaN	NaN
2	3	11/07/19	07:45:12	Raw Counts	0	0	0	0	0	0	...	31.7	28.5	218	0000 0000 0000 0000	NaN	NaN

3 rows × 74 columns

Figure 3. Contents of APS File

PROCESSING THE WIBS FILE—Part I

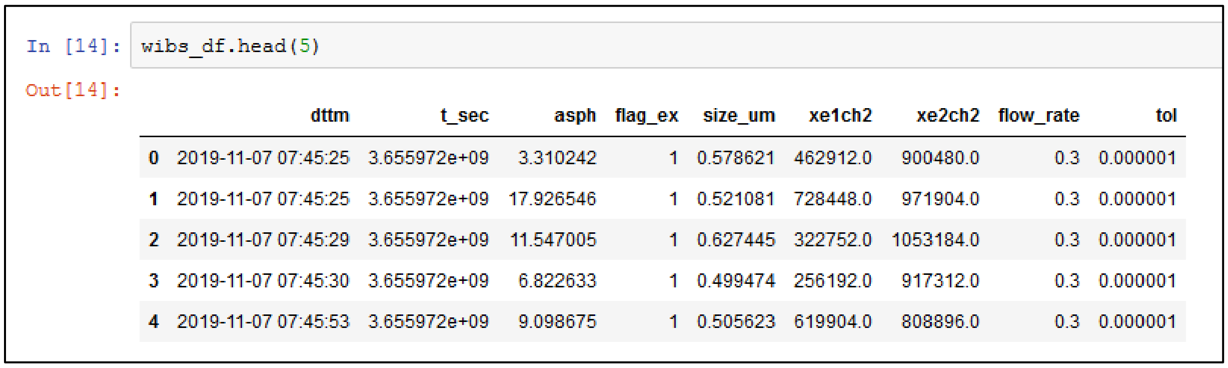
After speaking with the manufacturer, I don't need to use every element of the WIBS file. Below is a listing of the columns that are required for the analysis.

- ['NEO']['ParticleData']['Seconds']
- ['NEO']['ParticleData']['Asphericity']
- ['NEO']['ParticleData']['Flag_Excited']
- ['NEO']['ParticleData']['Xe1_FluorPeak']
- ['NEO']['ParticleData']['Xe2_FluorPeak']

The manufacturer told me that I needed to apply a very specific offset value to the ['NEO']['ParticleData']['Seconds'] column in order for the time to be synchronized to the correct testing day. The offset that I needed to apply is 66 years and 4 hours.

Furthermore, I was informed that I needed to include a couple of more columns that would be needed for some of the calculations. First was a column called “flow_rate”, which is the air flow constant that is the same value throughout the test. The next one was called “tol”, which is also a constant value and is described as the instrument tolerance value.

The end result is a DataFrame that is shown in Figure 4.



```
In [14]: wibs_df.head(5)
Out[14]:
```

	dttm	t_sec	asph	flag_ex	size_um	xe1ch2	xe2ch2	flow_rate	tol
0	2019-11-07 07:45:25	3.655972e+09	3.310242	1	0.578621	462912.0	900480.0	0.3	0.000001
1	2019-11-07 07:45:25	3.655972e+09	17.926546	1	0.521081	728448.0	971904.0	0.3	0.000001
2	2019-11-07 07:45:29	3.655972e+09	11.547005	1	0.627445	322752.0	1053184.0	0.3	0.000001
3	2019-11-07 07:45:30	3.655972e+09	6.822633	1	0.499474	256192.0	917312.0	0.3	0.000001
4	2019-11-07 07:45:53	3.655972e+09	9.098675	1	0.505623	619904.0	808896.0	0.3	0.000001

Figure 4. Sample of WIBS DataFrame

PROCESSING THE WIBS FT FILE

There are only two elements are needed from the FT file because it is meant to be used as a background subtraction. Therefore, we only need to look at the following columns.

- ['NEO']['ParticleData']['Xe1_FluorPeak']
- ['NEO']['ParticleData']['Xe2_FluorPeak']

The manufacturer stated that I needed to apply a correction factor to the Xe1 and Xe2 columns. This correction factor is calculated by subtracting the mean plus three standard deviations.

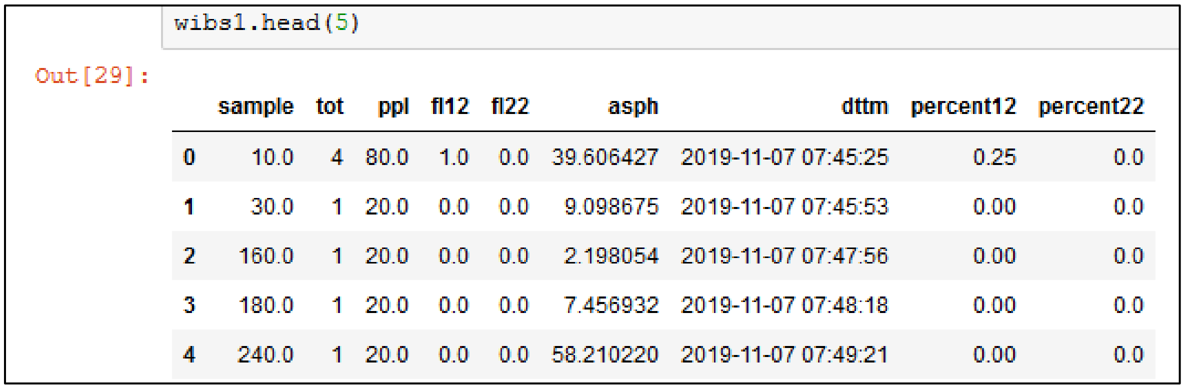
PROCESSING THE WIBS FILE—Part II

Once this correction factor was determined, it was applied to the wibs_df that was derived earlier and created two new columns: “fl_corrected12” and “fl_corrected22”.

Next, I needed to remove any negative values that resulted from applying the correction factor to the “fl_corrected12” and “fl_corrected22” columns. If there were negative values, they were changed to a value of zero.

If the “fl_corrected12” and “fl_corrected22” columns had values that were greater than the tol level (1e-06) we created in Part I, then I needed to add a column to the wibs_df that indicated that the particle would fluoresce. These columns were labeled “is_fluro12” and “is_fluor22”.

Finally, I created an elapsed time column and then grouped the DataFrame by number of fluorescent particles. The result is shown in Figure 5.


```
wibs1.head(5)
```

```
Out[29]:
   sample  tot   ppl  fl12  fl22       asph                 dttm  percent12  percent22
0    10.0    4  80.0   1.0   0.0  39.606427  2019-11-07 07:45:25       0.25        0.0
1    30.0    1  20.0   0.0   0.0   9.098675  2019-11-07 07:45:53       0.00        0.0
2   160.0    1  20.0   0.0   0.0   2.198054  2019-11-07 07:47:56       0.00        0.0
3   180.0    1  20.0   0.0   0.0   7.456932  2019-11-07 07:48:18       0.00        0.0
4   240.0    1  20.0   0.0   0.0  58.210220  2019-11-07 07:49:21       0.00        0.0
```

Figure 5. Corrected WIBS DataFrame

As a final check, I made a diagnostic plot for the entire day of testing to make sure that we were seeing the kinds of instrument responses we expected, and to make sure that all of the above calculations were correct. That plot is shown in Figure 6.

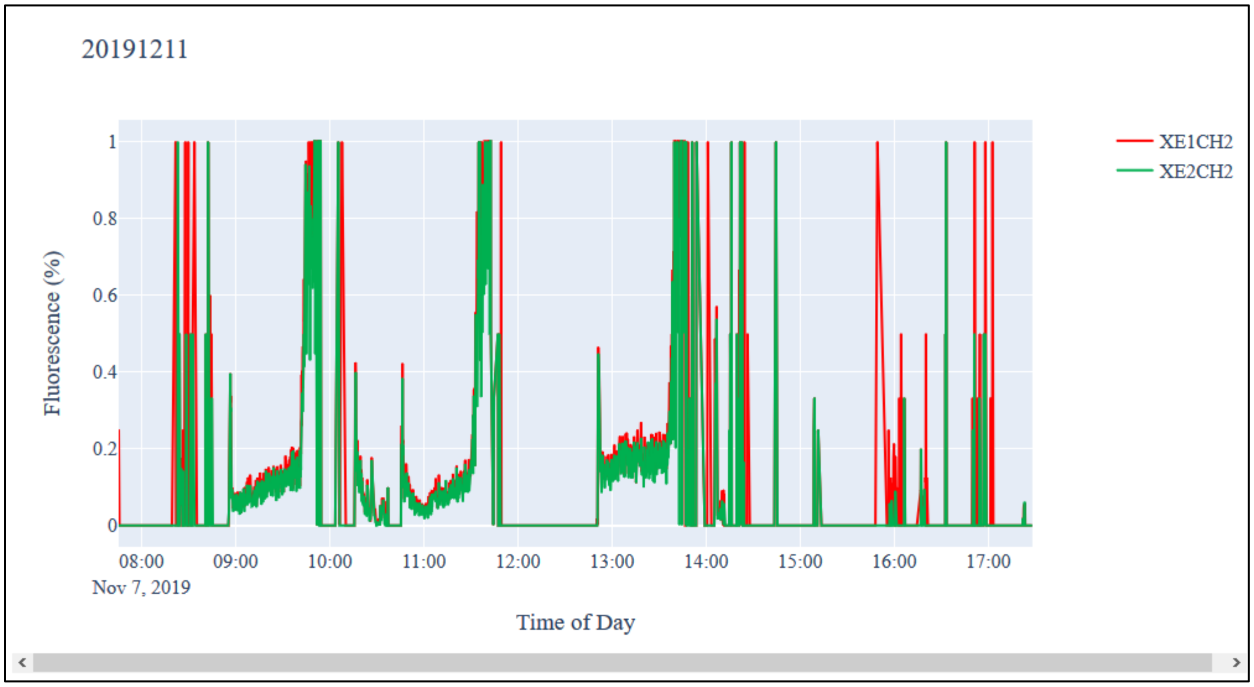
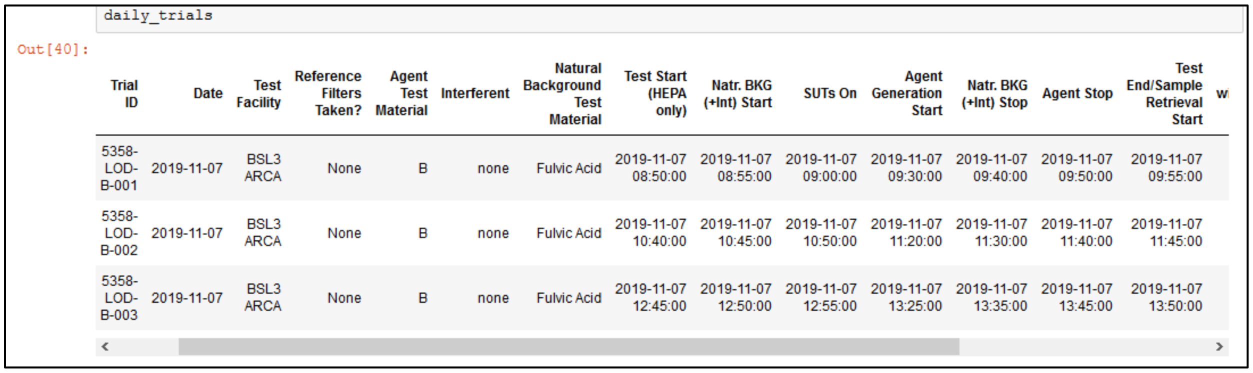


Figure 6. Fluorescence Diagnostic Plot

PROCESSING THE TEST LOG FILE

Since most of the test log file consisted of specific test dates and times, the first thing I needed to do was to convert the string values into Pandas datetime formats. I also had to add the trial date to each of the test times in order for specific time calculations to work correctly. The final step was to subset the data to only the test date of interest. The resultant DataFrame is shown in Figure 7.



```
daily_trials
```

Out[40]:

Trial ID	Date	Test Facility	Reference Filters Taken?	Agent Test Material	Interferent	Natural Background Test Material	Test Start (HEPA only)	Natr. BKG (+Int) Start	SUTs On	Agent Generation Start	Natr. BKG (+Int) Stop	Agent Stop	Test End/Sample Retrieval Start	W
5358-LOD-B-001	2019-11-07	BSL3 ARCA	None	B	none	Fulvic Acid	2019-11-07 08:50:00	2019-11-07 08:55:00	2019-11-07 09:00:00	2019-11-07 09:30:00	2019-11-07 09:40:00	2019-11-07 09:50:00	2019-11-07 09:55:00	
5358-LOD-B-002	2019-11-07	BSL3 ARCA	None	B	none	Fulvic Acid	2019-11-07 10:40:00	2019-11-07 10:45:00	2019-11-07 10:50:00	2019-11-07 11:20:00	2019-11-07 11:30:00	2019-11-07 11:40:00	2019-11-07 11:45:00	
5358-LOD-B-003	2019-11-07	BSL3 ARCA	None	B	none	Fulvic Acid	2019-11-07 12:45:00	2019-11-07 12:50:00	2019-11-07 12:55:00	2019-11-07 13:25:00	2019-11-07 13:35:00	2019-11-07 13:45:00	2019-11-07 13:50:00	

Figure 7. Final Daily Trial DataFrame

WIBS CALCULATIONS

By far the most challenging aspects of this project was determining the specific WIBS calculations that the original software provided. This was a lot of trial and error and consulting with the scientists conducting the test. I contacted the manufacturer many times regarding these calculations and they were unable to help me. I don't think they were unwilling to help, I just don't think they knew how to do the calculations by hand and were therefore reluctant to say something that might be incorrect. Regardless, I was finally able to get these calculations to work correctly.

There are six calculations that were required for the desired output. The specific time points mentioned are listed in the Test Log file. They are as follows:

1. WIBS Agent Fluorescence Average for Xe1
   a. This calculation determines the average fluorescence for the Xe1 channel from the [Agent Stop] time to the [Natr. BKG (+Int) Stop] time.
2. WIBS Agent Fluorescence Average for Xe2
   a. This calculation determines the average fluorescence for the Xe2 channel from the [Agent Stop] time to the [Natr. BKG (+Int) Stop] time.
3. WIBS Natural Background Fluorescence Average for Xe1
   a. This calculation determines the average fluorescence for the Xe1 channel from the [SUTs On] time to the [Natr. BKG (+Int) Stop] time.
4. WIBS Natural Background Fluorescence Average for Xe2
   a. This calculation determines the average fluorescence for the Xe2 channel from the [SUTs On] time to the [Natr. BKG (+Int) Stop] time.
5. WIBS Natural Background + Agent Fluorescence Average for Xe1
   a. This calculation determines the average fluorescence for the Xe1 channel from the [Agent Generation Start] time to the [Agent Stop] time.
6. WIBS Natural Background + Agent Fluorescence Average for Xe2
   a. This calculation determines the average fluorescence for the Xe2 channel from the [Agent Generation Start] time to the [Agent Stop] time.

Each of the calculations correspond to a specific time frame event during the test and these are marked on the output plot as dashed vertical lines. Figure 8 shows the plot for trial 1 with the corresponding WIBS calculations shown at the bottom.

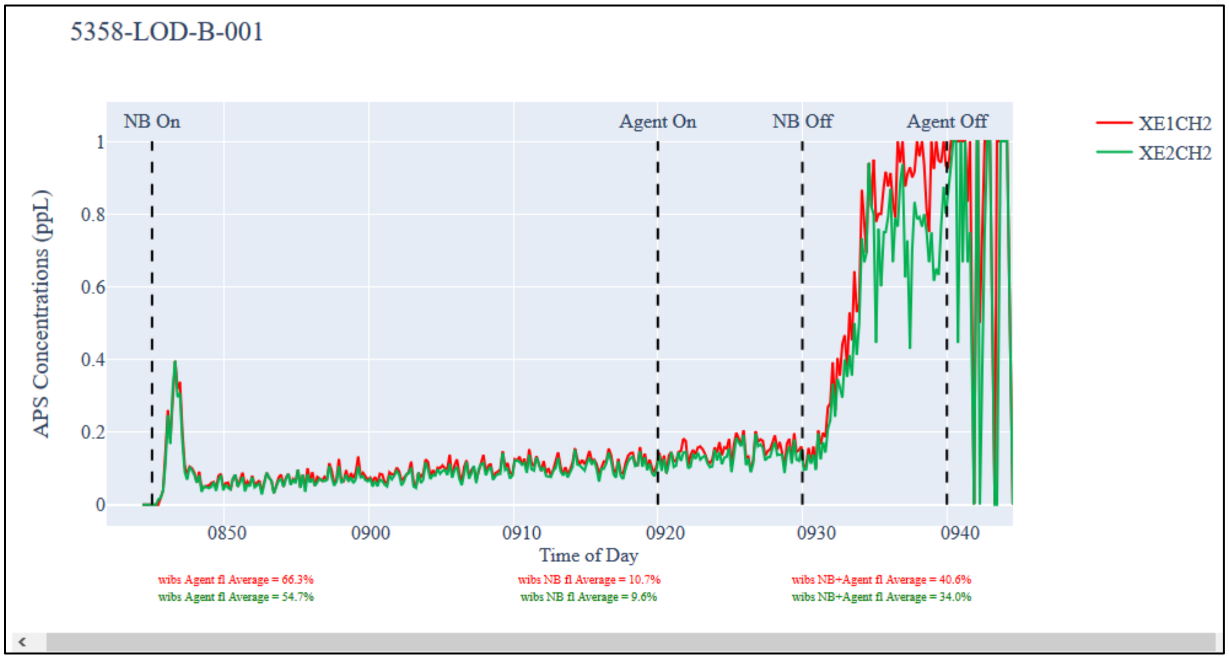
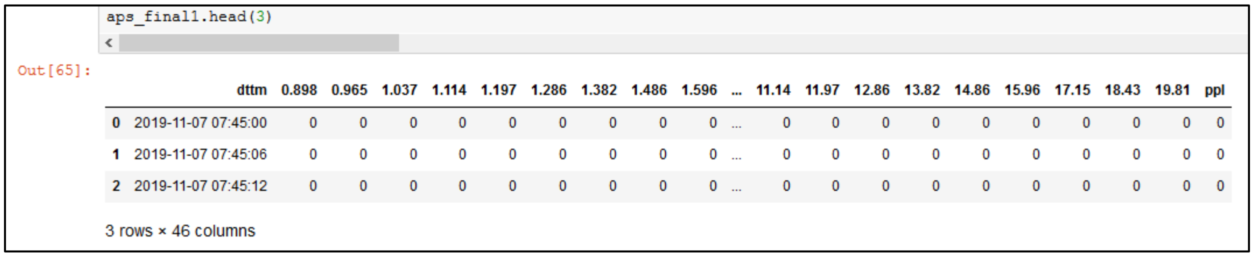


Figure 8. Plot of Trial 1 with Corresponding WIBS Calculations

PROCESSING THE APS FILES

The APS instrument measures particle sizes from 0.523 µ up to 19.81 µ. For our tests, we are only interested in the range between 0.898 µ - 19.81 µ. Sometimes the instrument operator manually sets that collection range so that only those values are recorded in the data file. Other times, however, all ranges are collected. Therefore, when we process the APS files, we need to make sure that our DataFrame only contains the 0.898 µ - 19.81 µ sizes.

Another important column that needs to be added to the APS DataFrame is a Particle per Liter (ppL) value. This is the main value that will be used for the APS plots and for other APS calculations. To determine this number, we sum the number of particles counted for each bin size across each timestamp. This number is then divided by the average sampling interval between the testing timestamps (for most APS trials this average sampling interval is six seconds). This results in a single column with ppL values for each timestamp in a trial, which is then added to our APS DataFrame. This process is repeated three times, one for each APS file. A typical APS DataFrame is shown in Figure 9.



```
aps_final1.head(3)
```

Out[65]:

	dttm	0.898	0.965	1.037	1.114	1.197	1.286	1.382	1.486	1.596	...	11.14	11.97	12.86	13.82	14.86	15.96	17.15	18.43	19.81	ppl
0	2019-11-07 07:45:00	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	2019-11-07 07:45:06	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	2019-11-07 07:45:12	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

3 rows × 46 columns

Figure 9. Typical Final APS DataFrame

APS CALCULATIONS

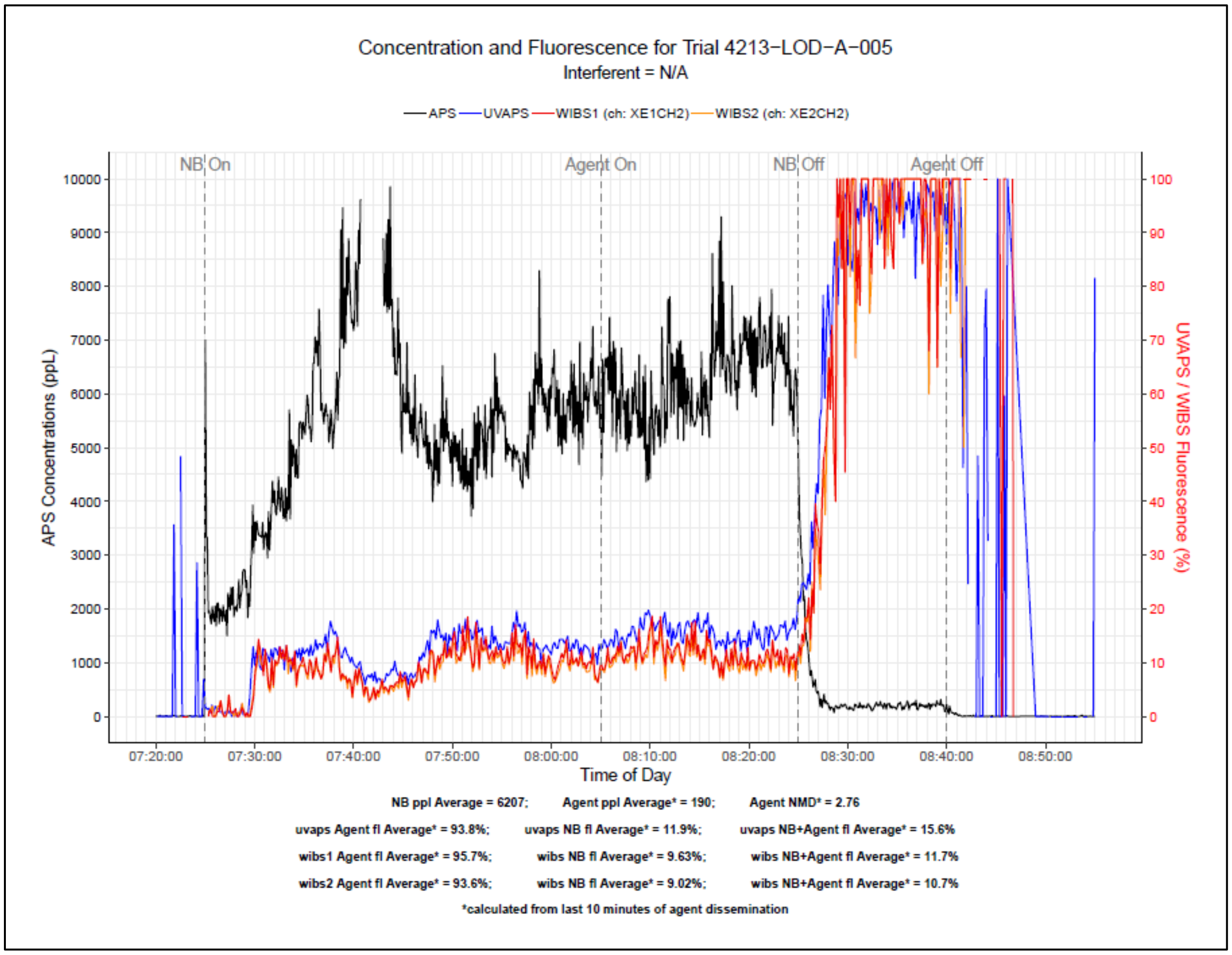
There are three main calculations that derive from the APS files. Like the WIBS calculations, these were very difficult to determine the correct time intervals needed for accurate values. There was a lot of trial and error involved in order to ensure that the code was producing the data needed. The three calculations required are:

1. Natural Background ppL Average
    a. This calculation determines the average natural background ppL counts from the [SUTs On] time to the [Natr. BKG (+Int) Stop] time.
2. Agent ppL Average
    a. This calculation determines the average agent ppL counts from the [Natr. BKG (+Int) Stop] time to the [Agent Stop] time.
3. Agent Number Mean Diameter (NMD) Average
    a. This calculation determines the average particle mean diameter from the [Natr. BKG (+Int) Stop] time to the [Agent Stop] time.

These calculations are repeated for each of the three APS files. Only one set of values is reported on the final plot, however. The particular APS file that will be used for reporting will be determined by the test officer based on what happened during each test. We are using three APS instruments for each test as triple-redundancy referee systems. Therefore, if all instruments perform correctly, then it shouldn't matter which instrument is used for the calculations as they all should be providing similar values.

MAKING THE FINAL PLOT

A lot of work went into producing this final plot beyond all of the previous calculations, data preparation, and analysis. The people that will use this plot are used to seeing it in a certain way, with very specific placement of labels, data elements, and fonts. To give some perspective, Figure 10 is a plot produced in 2018 by the software we are no longer able to use.



NB ppl Average = 6207; Agent ppl Average* = 190; Agent NMD* = 2.76

uvaps Agent fl Average* = 93.8%; uvaps NB fl Average* = 11.9%; uvaps NB+Agent fl Average* = 15.6%

wibs1 Agent fl Average* = 95.7%; wibs NB fl Average* = 9.63%; wibs NB+Agent fl Average* = 11.7%

wibs2 Agent fl Average* = 93.6%; wibs NB fl Average* = 9.02%; wibs NB+Agent fl Average* = 10.7%

*calculated from last 10 minutes of agent dissemination

Figure 10. Original Output Plot from Legacy Software

As one can see, there is a lot of information contained on each plot. There are three individual line plots spread across dual y-axes. In addition, there are vertical dashed lines that indicate important phases of the trial. Finally, all of the WIBS and APS calculation information is included below the x-axis.

Figure 11 shows the final plot produced by my code.

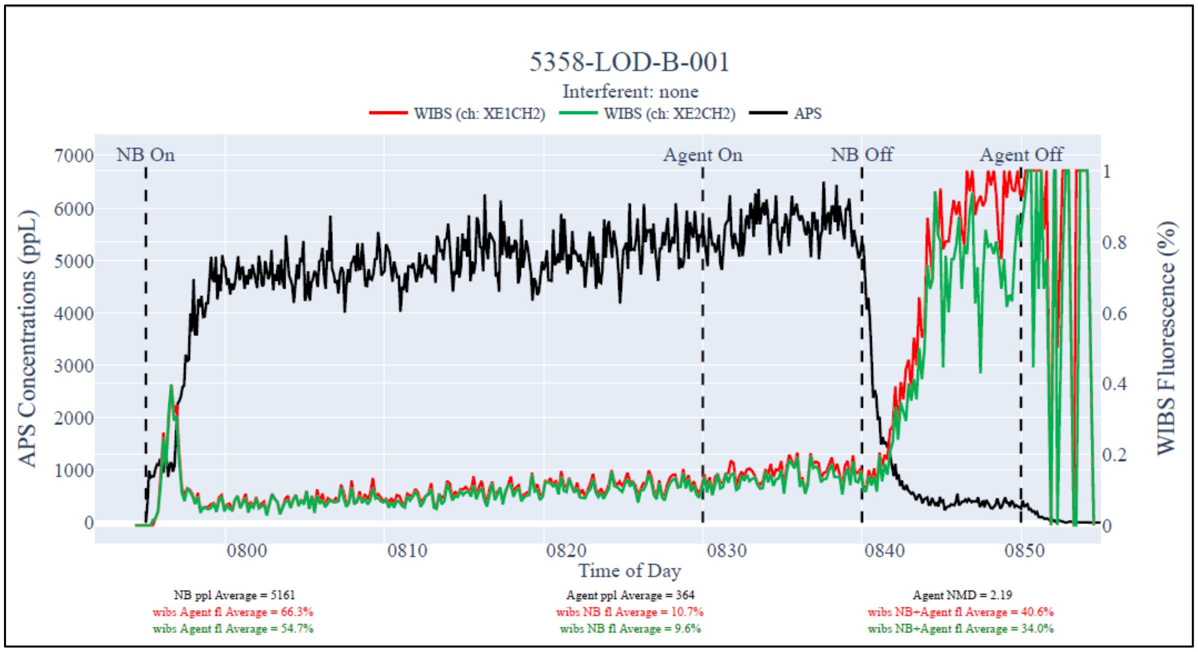


Figure 11. Final Plot from Jupyter Notebook Code

While not 100% identical to the original plot in Figure 10, the code produces output very similar to what the original software created.

Some of the challenges that were encountered while making this plot are described below:

- Dual Y-Axes—all of the usual plotting programs I typically use were not able to produce a dual axis that would work with this data. Either I was coding it incorrectly, or as was more common, the plotting library simply did not support a dual axis configuration. Therefore, I had to use Plotly, which is something I was unfamiliar with and required a lot of learning time on my part in order to get the dual axes to work correctly.
- Adding Vertical Lines—once I decided on using Plotly, I had to find ways to add all of the other features to the plot that were required. I found many ways to add vertical lines, but adding the text in the correct place proved problematic. I eventually came across an elegant solution that ultimately helped me add the calculations at the bottom of the plot as well, but I spent a lot of time working out this solution.
- Adding Calculation Text—one of the main points of producing the output plot is including the calculations at the bottom of the chart. Again, I spent a lot of time on this aspect but eventually found a solution that could easily be modified if additional information is ever required.
- X-Axis Scale—although only one set of x-axis labels are shown, there are actually two different time scales on the x-axis. The WIBS data actually correspond to the x-axis labels shown. The APS data, however, are recorded 60 minutes prior to those times. This is simply due to the fact that the APS instruments are turned on well before the test actually begins. While this is usually about 60 minutes prior to the start of the test, that isn't always the case and can vary up to plus or minus 10 minutes. So in order to have all of the plots show up at the correct times, I had to apply a time-shift to the APS plot and then write some code that would automatically position the x-axis time markers.

FUTURE WORK

Because this code was written in a Jupyter Notebook, it does not lend itself to automatically processing hundreds of files. My future plan for this code is to port it to an actual Python script and adjust the code to automatically load and process files. I haven't really investigated how to do this yet, but it is certainly something I will consider for the future.

REFERNCES

[1] WIBS-5/NEO. (2020). Retrieved from https://www.dropletmeasurement.com/product/wideband-integrated-bioaerosol-sensor/

[2] Takai, T. M. (2011). *Interim Guidance on Networthiness of Information Technology (It) Connected to DoD Networks*. Washington, DC: Department of Defense

[3] Hulslander, D. (2018, May). Open HDF5 files with Python Sample Code. Retrieved from https://www.neonscience.org/hdf5-intro-python

[4] Battelle. (n.d.). NEON AOP Hyperspectral Data in HDF5 format with Python - Tiled Data. Retrieved from https://www.neonscience.org/neon-aop-hdf5-tile-py

[5] Baron, P. A. (1986). Calibration and Use of the Aerodynamic Particle Sizer (APS 3300). *Aerosol Science and Technology*, *5*(1), 55–67. doi: 10.1080/02786828608959076

[6] Chen, B., & Crow, D. (1986). Use of an aerodynamic particle sizer as a real-time monitor in generation of ideal solid aerosols. *Journal of Aerosol Science*, *17*(6), 963–972. doi: 10.1016/0021-8502(86)90022-4