NASA/TM—2017-219487

Engine Icing Data—An Analytics Approach

Brooke A. Fitzgerald
Hampshire College, Amherst, Massachusetts

Ashlie B. Flegel
Glenn Research Center, Cleveland, Ohio

May 2017

NASA/TM—2017-219487

Engine Icing Data—An Analytics Approach

Brooke A. Fitzgerald
Hampshire College, Amherst, Massachusetts

Ashlie B. Flegel
Glenn Research Center, Cleveland, Ohio

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

May 2017

This work was sponsored by the Advanced Air Vehicle Program
at the NASA Glenn Research Center

Engine Icing Data—An Analytics Approach

Brooke A. Fitzgerald[*]
Hampshire College
Amherst, Massachusetts 01002

Ashlie B. Flegel
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

Abstract

Engine icing researchers at the NASA Glenn Research Center use the Escort data acquisition system in the Propulsion Systems Laboratory (PSL) to generate and collect a tremendous amount of data during testing. Currently these researchers spend a significant amount of time processing and formatting their data, selecting important variables, and plotting relationships between variables, all by hand, generally analyzing data in a spreadsheet-style program (such as Microsoft Excel). Though spreadsheet-style analysis is familiar and intuitive to many, processing data in spreadsheets is often unreproducible and small mistakes are easily overlooked. Spreadsheet-style analysis is also time inefficient. The same formatting, processing, and plotting procedure has to be repeated for every dataset, which leads to researchers performing the same tedious data method of manually converting data from one raw form into another format to enable a more convenient consumption of the data over and over instead of making discoveries within their data. This paper documents a data analysis tool written in Python hosted in a Jupyter notebook that vastly simplifies the analysis process. From the file path of any folder containing time series datasets, this tool batch loads every dataset in the folder, processes the datasets in parallel, and imports them into a widget where users can search for and interactively plot subsets of columns in a number of ways with a click of a button, easily and intuitively comparing their data and discovering interesting dynamics. Furthermore, comparing variables across data sets and integrating video data (while extremely difficult with spreadsheet-style programs) is quite simplified in this tool. This tool has also gathered interest beyond engine icing, and may be used by researchers across NASA Glenn Research Center. This effort exemplifies the enormous benefit of automating data processing, analysis, and visualization, and will help researchers move from raw data to insight in a much smaller time frame.

Introduction

The hidden trends, insight, and information buried in data are best uncovered and understood through data analysis and visualization. However, the specific tools and practices researchers use to analyze their data can often be an afterthought. Researchers turn to what they know, even if utilizing what they know is tedious, time-consuming, and can easily be improved.

In today's day and age, spreadsheets (Excel spreadsheets in particular) have become ubiquitous with data analysis, especially among lay persons and nonprogrammers. For simple, one-off uses, Excel works extremely well as an analysis tool with its familiar graphics and processing capabilities. However, Excel cannot maintain complete reproducibility, because it stores no record of what changes were made to the data at what time. This lack of reproducibility means that if there is a calculation error, researchers have no way of going back to a version that was error free.

[*]NASA Glenn Research Center, LERCIP summer intern from Hampshire College.

Because spreadsheet-style data analysis is generally done manually by clicking and dragging, it is extremely easy for small human errors to pervade analysis. In fact, a meta-analysis of 13 field audits of real-world spreadsheets conducted by Panko (Ref. 1) showed that 88 percent of all spreadsheets contained some error. Furthermore, as the datasets and necessary analysis becomes more and more complicated, Excel and its counterparts do not have the flexibility that researchers need to easily and immediately compare different runs, a multitude of different columns, and statistical attributes of the data. This need for immediate and reproducible processing and interactivity is the need that prompted this effort.

This effort developed a tool for use by researchers conducting engine icing tests in the Propulsion Systems Laboratory (PSL). The PSL facility is a ground based altitude test chamber (Ref. 2) that can test full scale engines (Ref. 3), rigs, and components for fundamental research (Ref. 4). The facility is outfitted with a spray bar system upstream of the test articles that can produce the desired ice crystal cloud. The data from these tests are used to provide insight into the physics of the ice crystal icing phenomenon and help develop models to predict the ice accretion inside current and future engine designs. Each test generates a large amount of data. The steady-state data acquisition system, ESCORT, alone is collecting data from 3,187 channels. This is in addition to videos, high speed data, and other specialty measurement systems which makes having an efficient way to analyze the large amount of data imperative.

Tools

As data analytics rises in popularity, more programs and programming languages offer data munging and analysis capabilities. After much research about which tools to use, a data analysis tool was developed in Python and deployed in a Jupyter notebook. This section discusses the decision to use both.

Python

This tool was programmed in Python for a variety of reasons. First, Python is an extremely popular open source language (Ref. 5) whose wide usage indicates frequent updates and more support than the less popular data-science specific languages like R or Julia. Second, Python's popularity and high level syntax also means that a large population will be able to understand and adapt the code. Finally, Python has a number of packages that allow easy analysis and manipulation of both spreadsheet-style data with the package pandas and numerical data using the package numpy. Pandas stores spreadsheet-style data in a data structure called a DataFrame, and numpy stores numerical data in a data structure called an array, and both packages streamline and optimize their respective calculations.

Jupyter Notebook

In considering how users, in this case researchers, would interface with their data, it was important to maintain a familiar interface while still providing robust interactivity. While Python does have various Graphical User Interfaces (GUI) packages (Kivy, Tkinter, PyGame, etc.) they all seemed labor intensive difficult to distribute to researchers, and making the User Interfaces (UI's) have aesthetic appeal could have taken a lot of extra code. Furthermore, while perhaps possessing effective building blocks for interactivity, they did not offer integration with any of the spectacular interactive graphing libraries built for the web.

The requirements of familiarity and interactivity are both fulfilled with the Jupyter notebook, described on jupyter.org as "a web application that allows you to create and share documents that contain live code, equations, visualizations, and explanatory text" (Ref. 6).

Jupyter notebooks support code written in a variety of languages, Python being one of them. Since most people are familiar with the interface of a web browser, Jupyter notebooks work well for people not entirely comfortable with command line programming (e.g., lay persons, "the average user").

Jupyter notebooks also have robust JavaScript and HyperText Markup Language (HTML) integration. JavaScript/HTML capabilities used in this tool include interactive graphing using the plotly package, widget capabilities using the ipywidgets package, and video display and interaction using

HTML5 video. The video interaction feature was an important consideration due to the increasing prevalence of videos as a data source. For example, engine icing researchers analyze a considerable number of videos showing ice accumulating on different parts of the engine. Some spreadsheet-style analysis programs can embed videos, but there is no way to use web-scripting libraries that can link attributes of a video with attributes of data. While this tool currently only supports video loading and display, it offers future possibilities of interactive analysis of video data, something that is not possible with spreadsheet-style analysis.

Open Source

Another choice worth addressing is the decision to use open source technology such as Python and Jupyter Notebooks rather than existing data analysis tools and software. The purpose of this tool is not to replace all other forms of data analysis, but to simplify exploratory analysis and cater more specifically to researchers' needs.

There are some pieces of software whose sole purpose is to create a wide range of graphs and charts. This type of software works really well for creating publication-worthy plots, but often is not the best for interactive, exploratory data analysis. There are other pieces of software that are built for interactive data analysis, but do not support researchers' specific needs, like integrating video data or analyzing data within a specific range of time in the context of the data overall. This tool attempts to find a happy medium, decreasing the time and effort researchers need to put in to understand and make discoveries about their data while allowing easy data export in order to use other tools as more specific needs arise.

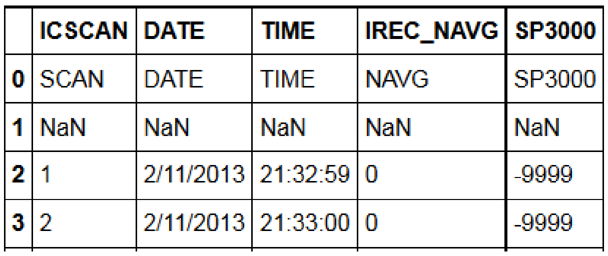
Furthermore, the analysis capabilities of proprietary data analysis tools often cannot be extended past their current state, while a tool made with open source technology can be improved by anyone with a good idea and some Python experience. Proprietary data analysis tools are also often cost-prohibitive, while this tool is distributed for free to any interested NASA Glenn Research Center (GRC) employee through the GRC GitLab version control system.

Data Processing

The first component of automating data analysis is to get data in a format that allows it to be analyzed. This process, known as data munging or data processing, is especially important to the huge amounts of data gathered by the facility steady state data acquisition system, Escort. The data gathered by this system is comprised of second by second readings of thousands of sensors from the Propulsions Systems Laboratory (PSL) or any other facility using the Escort system. Part of the reason why there are so many variables in the data is because there are sensor channels without sensors attached to them. These are called "spare channels" and they exist primarily to allow researchers to add new sensors in the future. However, even though these spare channels generally do not contain any actual data, they are not removed from the data automatically to prevent accidentally omitting important parameters that were not labeled correctly.

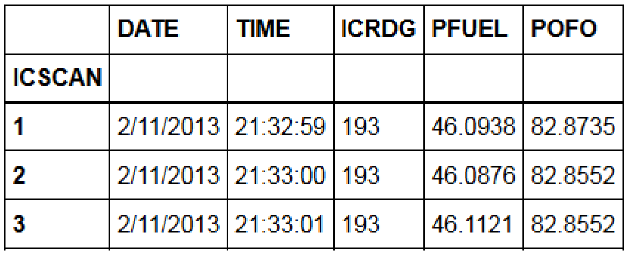
Similarly, there are also cases where the Escort system recognizes that there was a bad data read or that a sensor malfunctioned, and the data is marked with a numeric charater indicating the failure mode. The data also contains many system level parameters that remain constant throughout the run and do not actually help explain the dynamics of the system. As such, these parameters do not need to be plotted and can be stripped out and examined separately if needed.

After importing the data (Fig. 1), the units and the customer column names are filtered out. Then the columnsmarked as bad reads by the Escort system are deleted, as are the spare channels as shown in Figure 2. If the spare channels do contain important parameters, the researcher can always change the variable name in the original data and the analysis can continue from there. The system level parameters are then pulled out of the data and returned. A final calculation can be performed to calculate a specific range of interest. For this effort, the engine icing resesrcahers were interested in determinging when the the ice crystal cloud was turnined on and off. This information is used to give context in various graphs about that time period.



	ICSCAN	DATE	TIME	IREC_NAVG	SP3000
0	SCAN	DATE	TIME	NAVG	SP3000
1	NaN	NaN	NaN	NaN	NaN
2	1	2/11/2013	21:32:59	0	-9999
3	2	2/11/2013	21:33:00	0	-9999

Figure 1.—Unprocessed DataFrame. First four rows and five columns of unprocessed icing data.



	DATE	TIME	ICRDG	PFUEL	POFO
ICSCAN					
1	2/11/2013	21:32:59	193	46.0938	82.8735
2	2/11/2013	21:33:00	193	46.0876	82.8552
3	2/11/2013	21:33:01	193	46.1121	82.8552

Figure 2.—Processed DataFrame. First three rows and five columns of processed icing data.

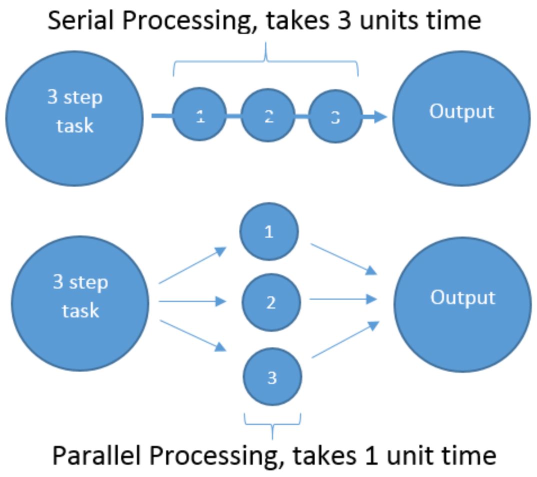

Serial Processing, takes 3 units time
3 step task
1
2
3
Output
3 step task
1
2
3
Output
Parallel Processing, takes 1 unit time

Figure 3.—Diagram of parallel processing. This diagram illustrates parallel processing's potential time savings (constrained by processors).

Batch/Parallel Processing

Having a function to process the data is a marked improvement over spreadsheet-style analysis. However, in order to minimize the amount of time the researcher spends processing data, this data analysis tool implements batch importing and parallel processing. A diagram of how parallel processing works is shown in Figure 3. As researchers do more and more tests, they gather more and more data. This means that the processing and visualization demands will only increase over time, and thus this data analysis tool needs to be able to scale to meet those needs. Batch importing and parallel processing leverages the computing power and time savings of automation and help researchers more easily scale their analysis.

Batch importing allows the user to specify a folder containing their data, and then the tool finds all of the .csv format files in the folder and loads them into memory. The user can either specify a processing function that they wrote or indicate that they have engine icing data or data gathered by the Escort data acquisition system.

The tool then uses the multiprocessing package to apply the processing function to each of the Data Frames simultaneously, utilizing all of the CPU cores that the user's machine has. Parallel processing can cut down processing time significantly especially when the number of Data Frames to process is large.

Variable Calculation

Variable calculation is a very robust component of spreadsheet-style analysis. However, as mentioned in the introduction, it is extremely easy to make mistakes in spreadsheet-style variable calculation and extremely difficult to fix said mistakes. Thus, this tool does contain the functionality to calculate new variables that the researchers are interested in.

TABLE 1.—FUNCTIONS FOR CALCULATION

• minimum	• standard deviation	• scale
• maximum	• variance	• quotient
• mean	• product	• difference
• median	• sum	• normalized
• mode	• percent change	

Though admittedly lacking the huge number of functions within spreadsheet-style programs, this data analysis tool is capable of performing all of the calculations seen in Table 1. The tool also has the capability to perform functions across rows as well as columns, to increase the options the researcher has for calculation. With minimal Python experience, users can add their own functions to the tool, but many numerical calculations can be done by applying available functions to a variable in sequence. The variable calculation is not currently integrated into the widget, although it could be part of a future update.

User Interface

The main user interface (UI) was developed entirely by combining various widgets within a Jupyter notebook, and contains two tabs: a settings tab and a plotting tab. The user interface of this tool puts a maximum amount of related information and features in a minimum amount of screen space. This UI was also created with the future in mind. If other features are added, they can be added as other tabs, and other plot types can be added as line of buttons under the current plotting options on the plotting tab.

The settings tab (see Fig. 4) displays a list of data sources, a list of videos, a list of presets, and various settings and options. The “Preview Data” button displays the first five rows of the selected data source(s). The “Add Data Source” button allows users to add a data source that was not batch imported and processed. The “Edit Data Source” button allows users to change features of the data source like the name of the data source, a range of interest to the user, how the range of interest is annotated, etc.

A preset is a selection of variables that the user expects to examine often and can save to easily plot in the future. The “Preview Preset” button displays the variables that were saved as a preset and the data source(s) that the variables came from and the “Export Preset” button exports the subset of variables to a csv. The data sources and videos are automatically populated with the options generated from whatever data was batch imported and processed (described above). The plotting tab, shown in Figure 5, displays the columns that are in all selected data sources and the columns that are missing from one or more data sources, as well as search fields for each. The “Add Selected” and “Remove Selected” buttons add and remove the columns that are selected with the mouse or with the search feature and add them to the “Columns to Plot” box. Pressing any of the buttons in the bottom row of the tab will plot the columns in the “Columns to Plot” box as indicated by which button is pressed. The “Add Linked Column” button does the same thing as the “Add Column” button, but allows columns that are named differently from different data sources to be plotted as the same variable. The “Remove All Links” button removes all of the linked columns from the “Columns to Plot” box and the “Remove All Columns” removes everything. The “Add All Columns” button adds every column to the “Columns to Plot” box, and is especially useful for data sources with few variables.

The dropdown button labeled “View All Sources” allows the user to only view the missing columns from a specific data source. The “Save Columns as Preset” button saves every variable currently in the “Columns to Plot” box as a preset that can then be loaded into the “Columns to Plot” box by selecting it from the “Select Preset” dropdown. The other available customization and plotting options are described in the next section.

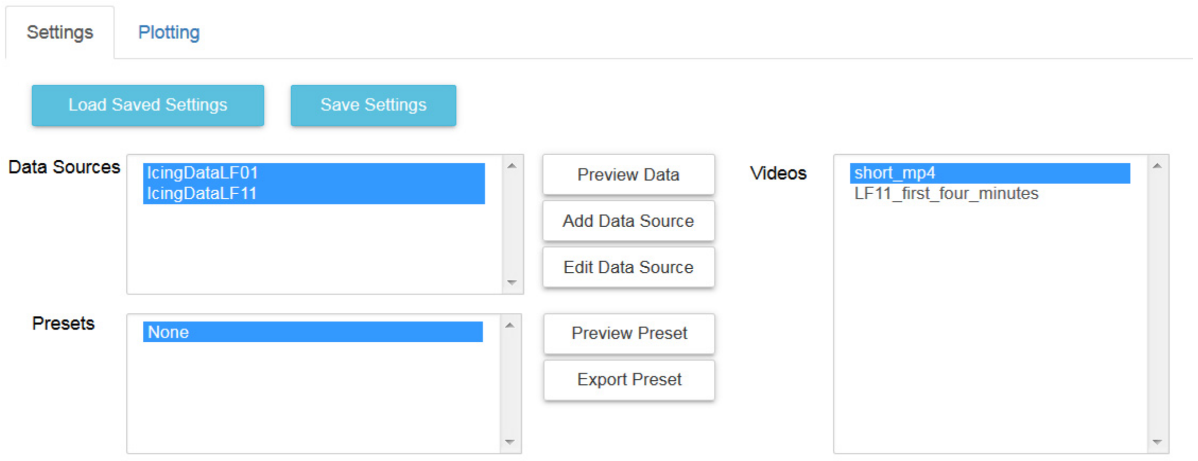


Figure 4.—Settings tab of the data analysis tool.

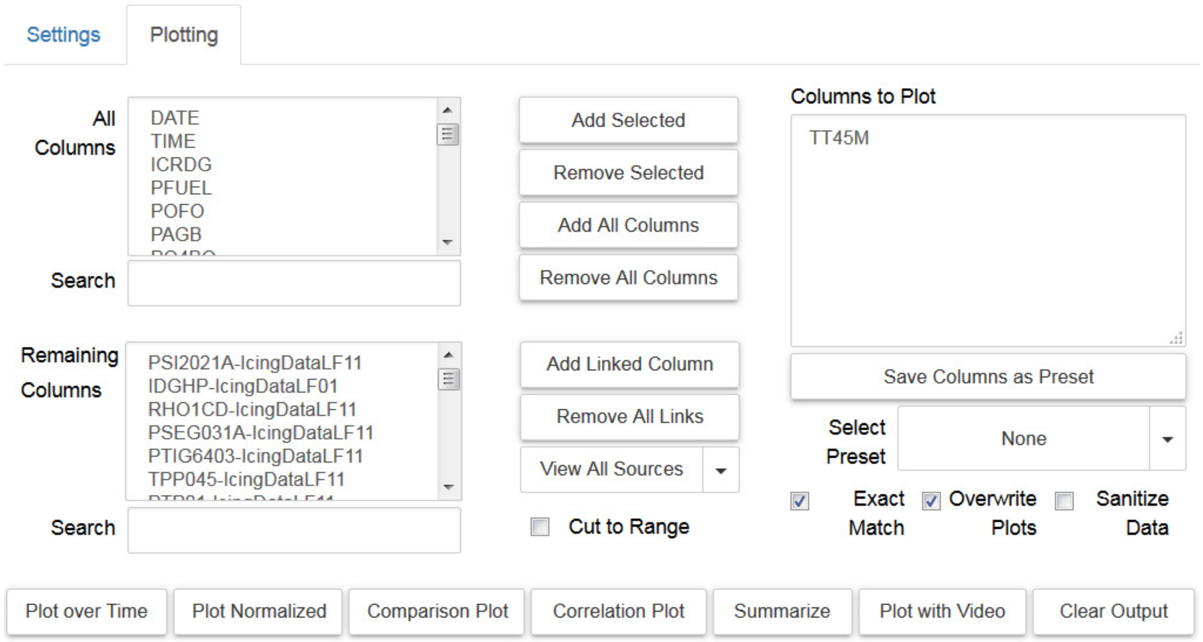


Figure 5.—Plotting tab of the data analysis tool.

Plotting and Data Analysis

"Overview first, zoom and filter, then details on demand" (Ref. 7), Ben Schneiderman's information-seeking mantra, was the motivating factor behind the plotting capabilities of this data analysis tool. As the researchers' data is time series data, the most basic overview of the data is a plot of the selected columns over time. The graphs are interactive, and as such the user can toggle traces on and off, get exact numbers and units on hover, and click and drag to zoom into specific areas.

Interactivity engages users with their data, encouraging them to zoom and hover and dive into the details. In the example of interaction in Figure 6 the user recognizes the difference in scales of the three variables, and can then click the "Plot Normalized" button, rescaling each column between zero and one so that the dynamics of the variables are more easily compared. Normalizing is one way the user can filter their data, and another filtering method available is to specify a range of interest, and by simply clicking a checkbox the user can now compare the full dataset with the range that they are interested in. In Figure 7 you can see the annotations displaying the range of interest.

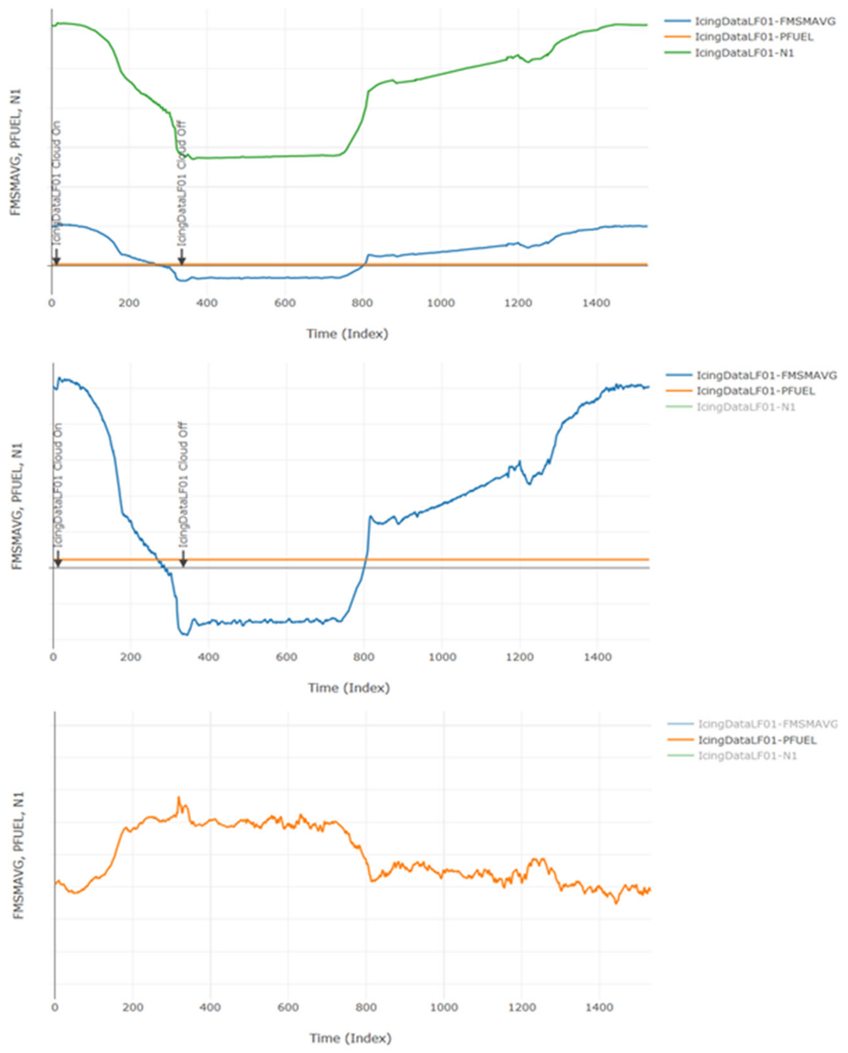


Figure 6.—Example interaction with plot over time. As the user toggles columns on and off, and zooms into the data, they discover that these three columns have interesting behavior, but have different scales.

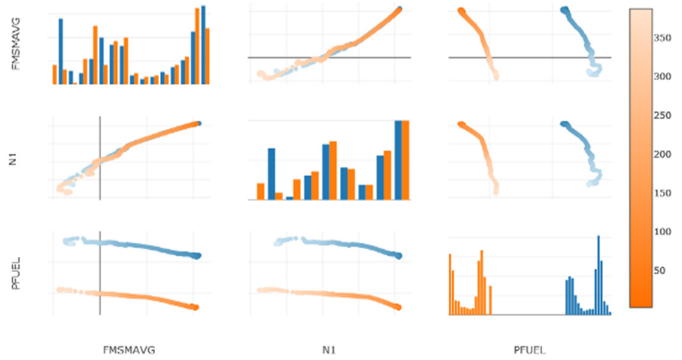


Figure 7.—Comparison Plot Scatterplot matrix of three variables from two different runs, colored by time. The histograms on the diagonal show each variable's distribution.

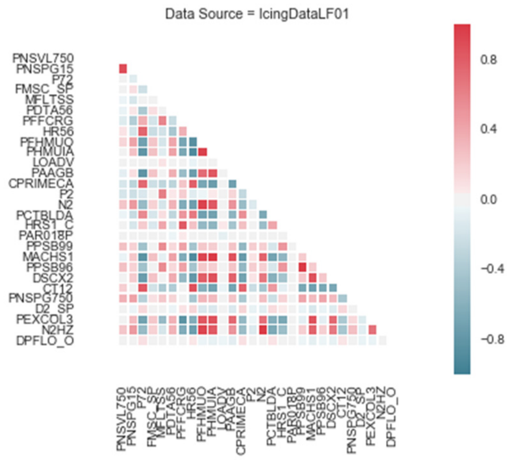


Figure 8.—Correlation Heatmap.

After viewing the overview and normalized plots, the user can now compare each variable against every other variable with a scatterplot matrix, as seen in Figure 7. The compare plot feature allows the user to get even more details about their data, seeing the distribution of each variable, and a colored trace indicating how each variable changes over time. Selecting multiple data sources at a time allows comparing the differences in distribution and behavior of each variable, giving the user even more information about their data and its broader context within multiple data sources.

As the researcher seeks to quantify more about their data, they can calculate and view the correlation of multiple columns, as well as get a statistical summary of each variable. The correlation of multiple columns is visualized as a heatmap - a plot where the data is graphically represented by color (see Fig. 8). The more positively correlated variables are represented as darker red squares, the more negatively correlated variables are represented as darker blue squares, and the columns with little to no correlation are represented as lighter blue/red or white squares. The statistical summary of each variable is displayed as a table within the tool. These features allow researchers to quickly and easily dive into the details of their data, exploring hypotheses and questions *as they arise* instead of after painful formatting.

Other Features

Saving and Loading Settings

If the user has a certain set of columns they plot a lot or if the user wants to change the names of the data sources every time they load the data analysis tool, they can save these settings and then load them at any time with the "Save Settings" and "Load Saved Settings" buttons. This is another way to cut down the time it takes to move from raw data and understanding the data fits into the larger context of research.

Exporting Data

As with any software development project, there is no tool that will be able to make every plot the user wants. Thus, if a researcher is interested in getting a certain subset of the data and making a graph more specific that the tool is currently capable of plotting, they can simply save the columns that they want to analyze as a preset and then click "export preset". This subsets only the columns in the preset from the data sources the preset was saved with, then exports them to csv files.

Sanitizing Data

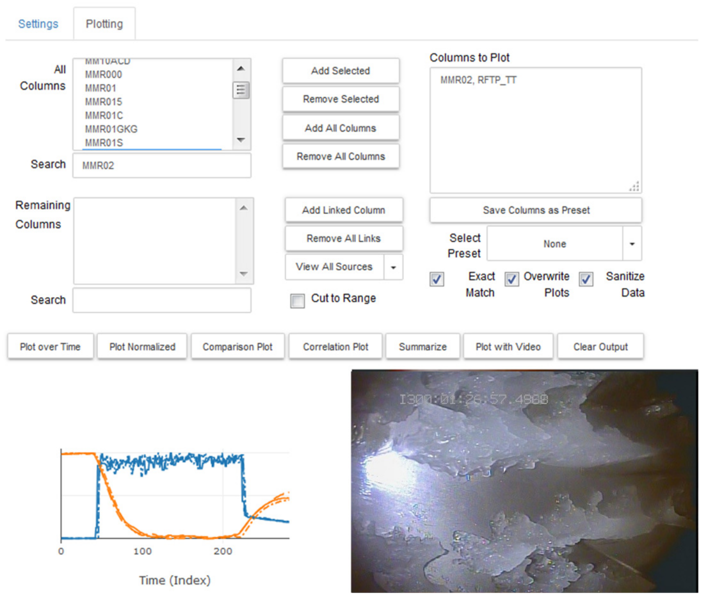
Because some research can include sensitive data, as is the case for engine icing for some tests, researchers need the option to get rid of the axes labels with sensitive information. Using spreadsheet-style analysis this can be cumbersome, occasionally making pasting a white box over the axes and taking a screenshot of the graph the easiest solution, but now it's as simple as checking a box and then creating plots as usual with the click of a button.

Limitations and Future Work

Currently the biggest limitation of this tool is the size of the data that the Jupyter notebook can handle. Because all of the data is read into memory to be processed and displayed, the notebook can only ingest as much data as it can store in memory. This worked fine for much of the current data from engine icing, but there is definitely the potential for overload. Future improvements to how the data is accessed might look something like using Jupyter's integration with Spark or some other package that allows for data to be pulled from a database or server as needed instead of stored in memory.

Another limitation is that calculating new columns by applying functions to existing columns is not integrated with the widget. Although users can calculate new columns right now using the function discussed in the variable calculation section above, it is still not as easy as it could be. Another tab can be added that creates a user interface for column calculation, and perhaps also makes the single value columns searchable.

The current plotting with video feature plots the normalized data alongside a video. Future expansion of the video capabilities would include synchronizing a line scrubbing across the graph with the playback time of the video, so the user could follow along the graph as the video plays. This would provide a visual mapping of the quantitative data in the plot and the qualitative data in the video, encouraging a new perspective and further insight. See Figure 9.



Search: MMR02

Columns to Plot: MMR02, RFTP_TT

Select Preset: None

✓ Exact Match ✓ Overwrite Plots ✓ Sanitize Data

☐ Cut to Range

Figure 9.—Plotting tab with plot with video output User view of plot output, with selected columns plotted normalized alongside video data.

Overall, this effort is only as complete as the researchers want it to be. At any time, they can build on the infrastructure with new functions and plots that will help automate their research. Because the code is visible, the researchers can see the examples of how the tool currently works, and should feel no hesitation to build on what was accomplished this summer.

This tool is currently available on the NASA GRC GitLab version control system, and anyone with permission on GitLab who wants access can seek it out there.

Conclusion

Spreadsheet-style analysis is convenient and familiar, but the capabilities of modern day open-source programming languages and libraries have blazed past spreadsheet-style analysis in terms of reproducibility, capabilities, and flexibility. The tool developed represents a first step in the direction of informative, scalable, and most importantly *automated* analytics. The more quickly researchers can move from hypothesis to testing and from raw data to insight, the more researchers are going to discover about their data. The more researchers discover about their data, the more researchers learn about the world, and the more researchers learn about the world the better off the world will be. This tool will be used by researchers across NASA Glenn Research Center and as such it will hopefully demonstrate the usefulness and potential that automated data analysis can provide.

References

1. Panko, Raymond R., "What we know about spreadsheet errors," *Journal of Organizational and End User Computing (JOEUC)* 10.2 (1998): 15–21.
2. Griffin, T.A., Lizanich, P., and Dicki, D.J., "PSL Icing Facility Upgrade Overview," 6th AIAA Atmospheric and Space Environments Conference, Atlanta, GA, June 16–20, 2014, AIAA-2014-2896.
3. Flegel, A.B., Oliver, M.J., "Preliminary Results from a Heavily Instrumented Engine Ice Crystal Icing Test in a Ground Based Altitude Test Facility," 8th AIAA Atmospheric and Space Environments Conference, Washington, D.C, June 13–17, 2016, AIAA-2016-3894. NASA/TM—2016-29132.
4. Struk, P.M., Tsao, J.C., and Bartkus, T.P. "Plans and Preliminary Results of Fundamental Studies of Ice Crystal Icing Physics in the NASA Propulsion Systems Laboratory," 8th AIAA Atmospheric and Space Environments Conference, AIAA-2016-3738, 2016.
5. O'Grady, Stephen, "The RedMonk Programming Language Rankings: June 2015," *RedMonk.com* [open source industry analyst firm], URL: http://redmonk.com/sogrady/2015/07/01/language-rankings-6-15/ [cited 10 August 2016].
6. "Project Jupyer Home Page," *Jupyter.org* [Jupyter notebook website], URL: http://jupyter.org/ [cited 10 August 2016].
7. Shneiderman, Ben, "The eyes have it: A task by data type taxonomy for information visualizations," *Visual Languages*, *1996. Proceedings, IEEE Symposium.* IEEE, 1996.