

# Introduction to Low Code Data Analysis with MATLAB®

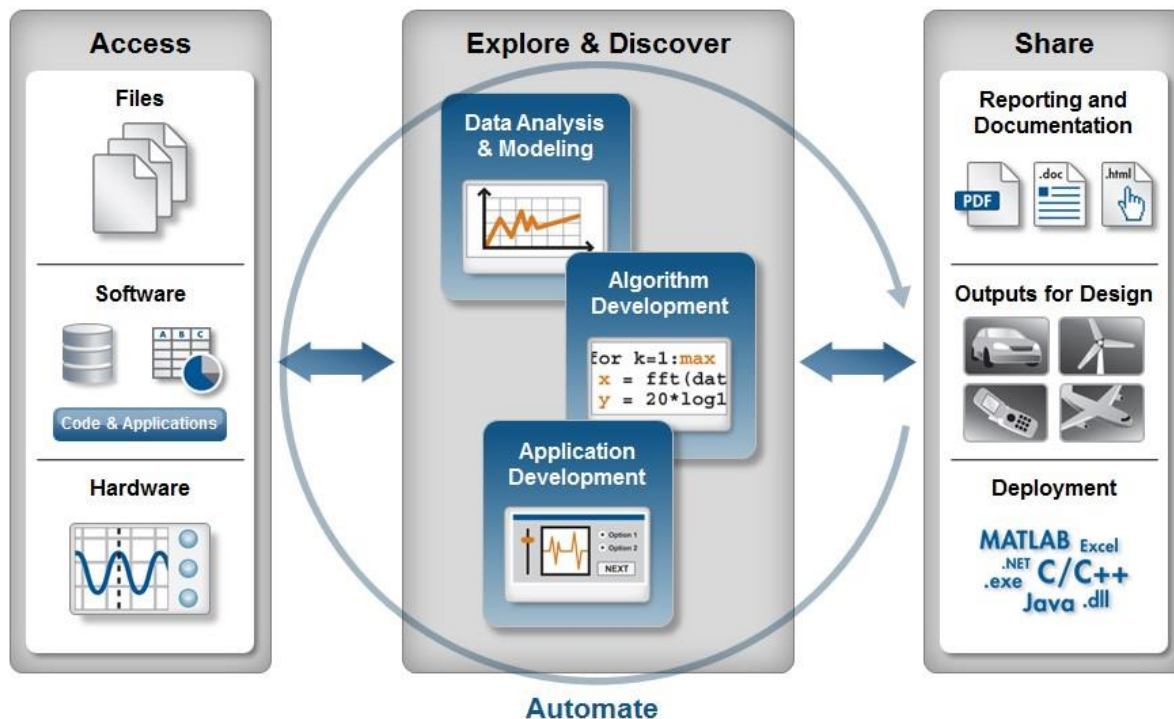
## *A Hands-On Workshop*

### Introduction

This workshop will provide a basic introduction to performing data analysis and visualization in MATLAB. Using a simple example, you will learn how to perform routine analysis tasks, including:

- Importing data
- Visualizing and analyzing data
- Preprocessing and cleaning data
- Automatically generating MATLAB code to automate future analysis
- Sharing results with others by automatically creating reports

The example will follow this general data analysis workflow shown here.

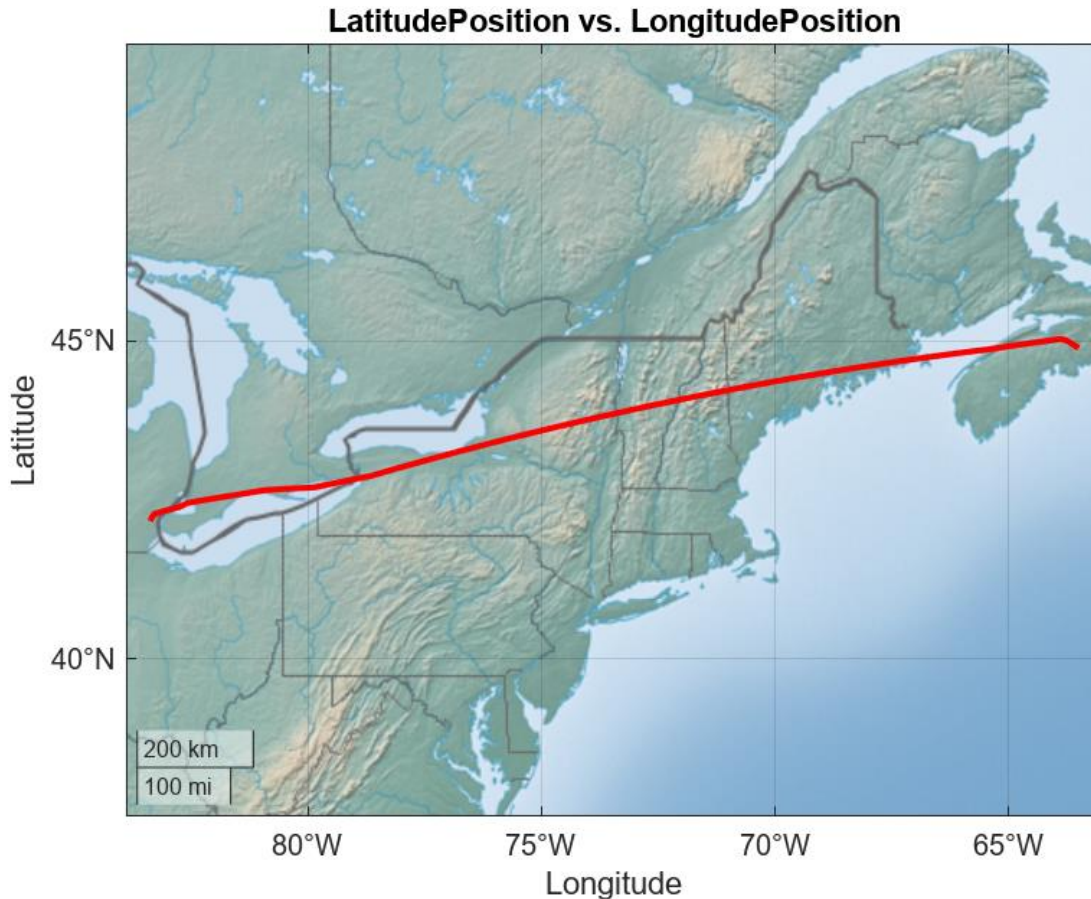


### Overview of Example

The example for this workshop uses commercial flight data provided by NASA. This is the original source of the data:

- [NASA Dash Link: Sample Flight Data](#)

In our analysis, we will import this data into MATLAB, perform some basic visualization and manipulation of the data, and develop a predictive model for one of the sensors. We will also document the steps and results of our analysis.



## Overview of MATLAB Online

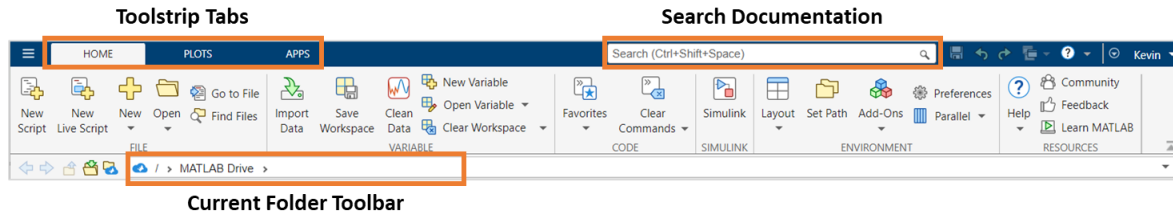
We will be leveraging [MATLAB Online](#) for this workshop, which allows you to access MATLAB through your web browser. The only requirement for this is that you have a MathWorks account set up. You can sign in, or create an account, [here](#). Once you've got an account set up, you can [start using MATLAB Online](#).

The toolstrip, which appears at the top of the MATLAB desktop within MATLAB Online, contains the following components:

- **Toolstrip Tabs** – By default, you will see the **HOME**, **PLOTS** and **APPS** tabs; each tab groups together functionality associated with common tasks. Additional tabs appear in the Toolstrip as needed to support your workflow (e.g., **EDITOR**, **PUBLISH** and **VARIABLE**).
- **Search** – this box allows you to search the documentation.

## Introduction to MATLAB for Data Analysis – A Hands-On Workshop

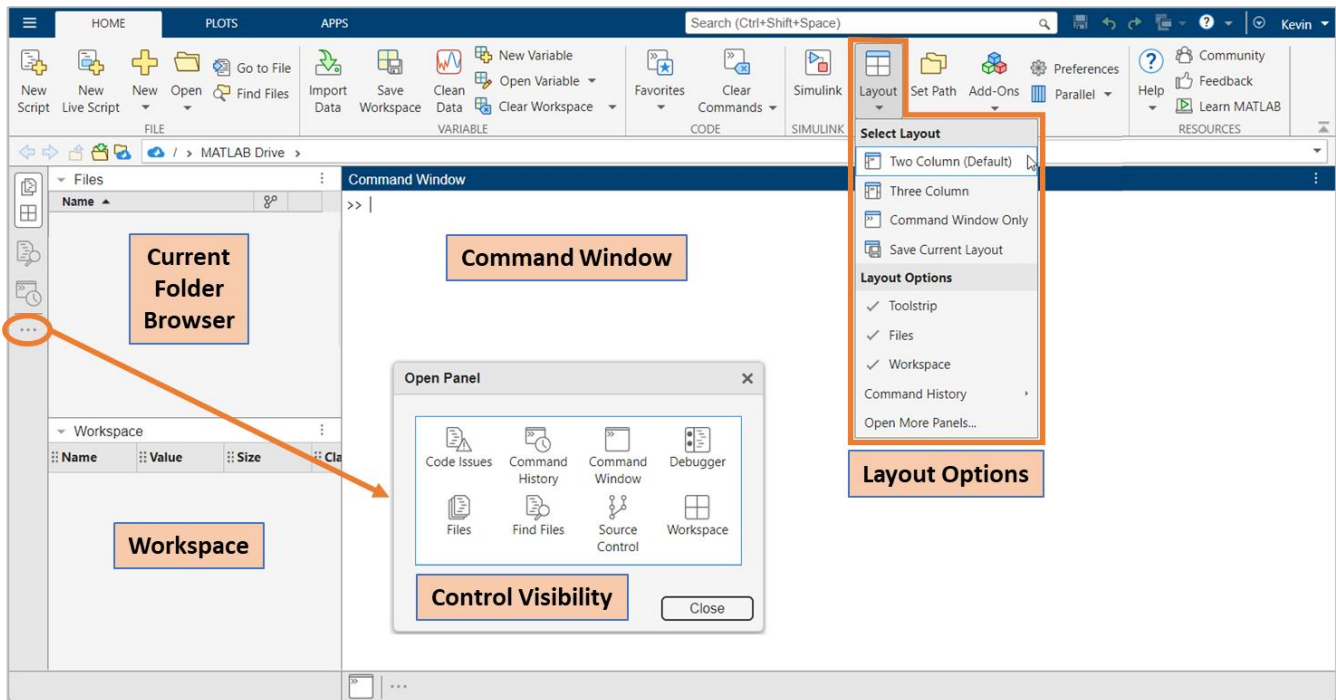
- **Current Folder Toolbar** – this enables you to specify the current working directory.



The rest of the MATLAB desktop is composed of a collection of independent windows. The size of each window can be adjusted, as well as their visibility. There are multiple layouts to choose from. In the two-column default layout, the MATLAB desktop displays the following windows:

- **Current Folder Browser** – enables you to access the contents of the current working directory. Use the options in the current folder toolbar to change the current working folder.
- **Command Window** – lets you execute MATLAB commands at the command line, indicated by the prompt (`>>`) .
- **Workspace** – allows you to explore data that was created in MATLAB or imported from files.

Note that you can control the visibility of these and other windows by selecting the ellipsis on the left-hand side as shown below, and/or by choosing options from the Layout dropdown.



## Technical Issues

If you run into issues during the workshop, consider the following steps:

- **MATLAB Online**

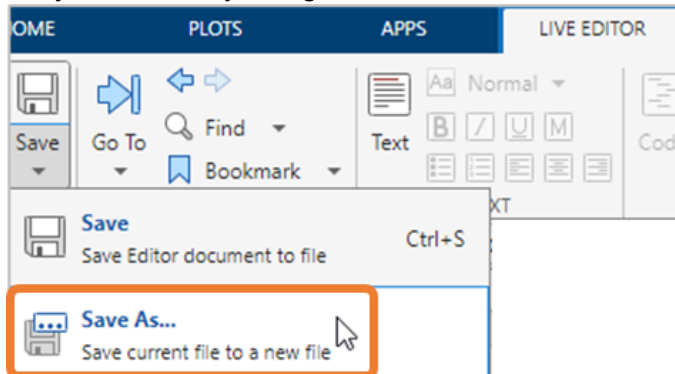
If you are having technical issues with MATLAB Online itself, try these options:

- Exit and rejoin in a new browser tab
- Post a question to the chat, and a facilitator will provide assistance

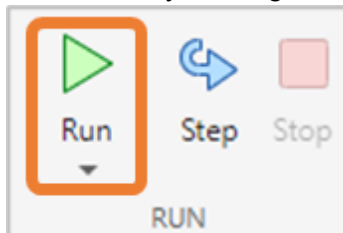
- **Catch-Up Files**

If you fall behind the presenter, you can attempt to catch up by opening the catch-up file called out throughout this guide. To do this:

- Open the **catch-up file** called out at the beginning of a specific step by double-clicking on it in the **Current Folder Browser**
- Immediately rename it by doing a **Save As** to a new file name:



- Run the entire file by clicking the **Run** button in the Live Editor tab:



## Step 1 – Importing Data into MATLAB

Here are the names of the folder you should be working in, and the data file.


- **Working Folder:** [Demo - Low Code Data Analysis](#)
- **Data File:** [FlightData.xlsx](#)

To import the data, follow these steps:

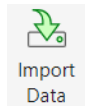
- **Navigate** to the working folder

To access this file, you should be in the current working directory. If you are not there, you can change the current folder as follows:

## Introduction to MATLAB for Data Analysis – A Hands-On Workshop

- Click on the  icon in the current folder toolbar.  
*This opens the folder navigation window.*
- Navigate to the working folder (**Demo – Low Code Data Analysis**)

- Click on the **Import Data** icon in the Home tab of the toolstrip:






- Select **FlightData.xlsx**, then click the “Open” option
- Alternatively, you can do either of the following in the **Current Folder Browser**:
  - Double-click on the **FlightData.xlsx** file in the Current Folder Browser, or...
  - Right-click on the **FlightData.xlsx** file and select Import Data

The Import Tool recognizes that this is an Excel file and provides a view of the data that will look something like this:


	A	B	C	D	E	F	G	H	I	J	K	L	M
	FuelQuantity	OilPressure	OilTemper...	LatitudePo...	Longitude...	Altitude	ExhaustTe...	FuelFlow	FanSpeed	TrueAirSpe...	WindDirec...	WindSpeed	WeightOn...
	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number
1	FuelQuantity	OilPressure	OilTempera...	LatitudePo...	LongitudeP...	Altitude	ExhaustTe...	FuelFlow	FanSpeed	TrueAirSpeed	WindDirection	WindSpeed	WeightOn...
2	8048	0	23.6748	44.8915	-63.5192	174	17.5	0	1.5	0	0	0	0
3	8048	0	23.6748	44.8915	-63.5192	174	17.5	0	1.5	0	0	0	0
4	8048	0	23.6748	44.8915	-63.5192	174	17.5	0	1.5	0	0	0	0
5	8048	0	23.6748	44.8915	-63.5192	175	17.5	0	1.5	0	0	0	0
6	8048	0	23.6748	44.8915	-63.5192	173	17.5	0	1.5	0	0	0	0
7	8048	0	23.6748	44.8915	-63.5191	174	17.5	0	1.5	0	0	0	0
8	8048	0	23.6748	44.8915	-63.5191	174	17.5	0	1.5	0	0	0	0

The **IMPORT** tab also lets you specify the range, the variable row containing the column names, as well as the data type (table, timetable, column vector, matrix, string array, etc.) of the imported variables. In this example, we will use the auto-selected options for Range, Variable Names Row, type of imported data, and options for unimportable cells.

Selection/History	A2:M4731...	Name	FlightData	Replace unimportable cells with NaN	 Import Selection
Variable Names Row	1	Type	Table		
<input type="checkbox"/> No Variable Names SELECTION		 Table Options IMPORTED VARIABLE		UNIMPORTABLE CELLS	

- Click the **Import Selection** icon in the **IMPORT** tab of the toolstrip: 

The data will now appear in the MATLAB **Workspace**:

Workspace			
Name	Value	Size	Class
 FlightData	47312x13 timetable	47312x13	timetable

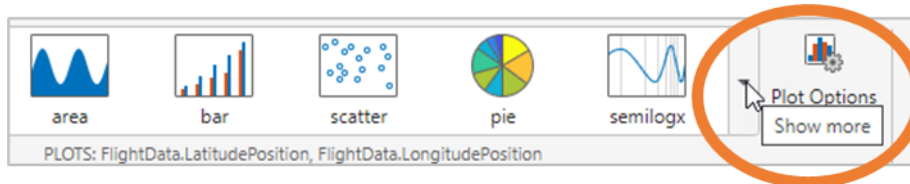


## Step 2 – Visualizing Data

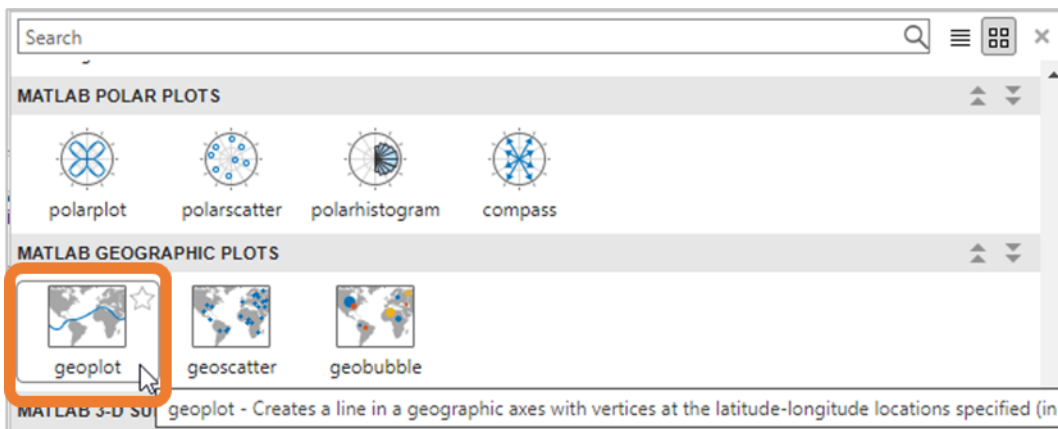
- Open the data in the Variable Editor by double-clicking on **FlightData** in the Workspace
- Select **LatitudePosition** and **LongitudePosition** by selecting their respective columns, in that order.

FlightData.xlsx × FlightData ×						
47312x13 table						
	1	2	3	4	5	6
	FuelQuantity	OilPressure	OilTemperature	LatitudePosition	LongitudePosition	Altitude
Min	3216	0	-390	0	-83.377	152
Max	8048	94.6275	157.2173	45.0282	0	31022
Mean	6113.6324	74.1827	99.0233	43.3991	-72.7072	24315.6414
Std Dev	1632.2277	13.984	26.8754	4.1916	9.3211	11023.1467
Missing	0	0	0	0	0	0
Class	double	double	double	double	double	double
1	8048	0	23.6748	44.8915	-63.5192	174
2	8048	0	23.6748	44.8915	-63.5192	174
3	8048	0	23.6748	44.8915	-63.5192	174
4	8048	0	23.6748	44.8915	-63.5192	175
5	8048	0	23.6748	44.8915	-63.5192	173
6	8048	0	23.6748	44.8915	-63.5191	174
7	8048	0	23.6748	44.8915	-63.5191	174

- Choose the “Show more” dropdown from the **VARIABLE** or **PLOTS** tab to view the different plotting options for the data selected.



- Choose **geoplot** by either scrolling to find it in the **MATLAB GEOGRAPHIC PLOTS** category or searching for it.



- Note that the corresponding commands are shown in the **Command Window**.

```
Command Window

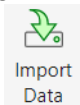
>> geoplot(FlightData.LatitudePosition,FlightData.LongitudePosition);
title("LongitudePosition vs LatitudePosition");
legend("show");
>>
```

### Step 3 – Code Generation from Import Tool

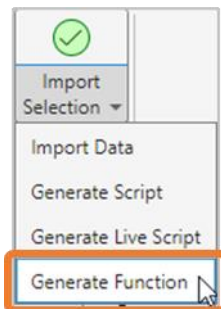
The **Import Tool** enabled us to *interactively* import the data into MATLAB. Next, we will autogenerate a function to automate this task; this will enable us to repeat this import operation on a similar specified .xlsx file, without manually interacting with the **Import Tool**.

To generate a function:


- Open the **Import Tool** again (as above, in step 2) – e.g., by:
  - Clicking on the **Import Data** icon in the **HOME** tab of the toolstrip:



- Double-clicking the **FlightData.xlsx** file in the **Current Folder Browser**, or...
  - Right-clicking on the file and selecting **Import Data**
- Click the dropdown arrow on the **Import Selection** icon in the **IMPORT** tab of the toolstrip and choose the “**Generate Function**” option:

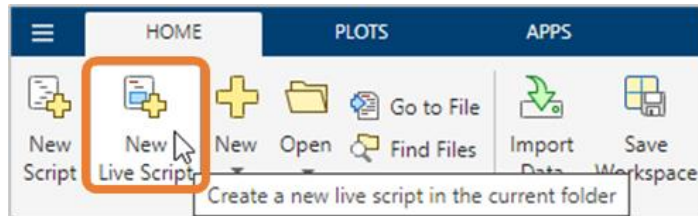


The Import Tool generates a new function file **untitled\*** and opens it in the **Editor** window.

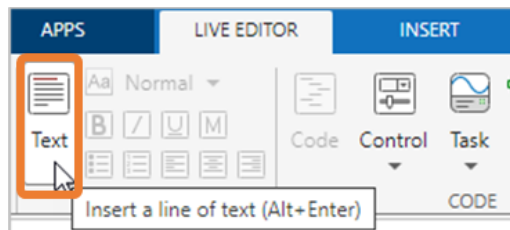
- To save the function, click on the **Save** icon  in the **EDITOR** tab. Save the file as **importfile.m** in the **Demo - Low Code Data Analysis** folder.
- Close the Editor window.

### Step 4 – Create a Live Script to Capture Steps

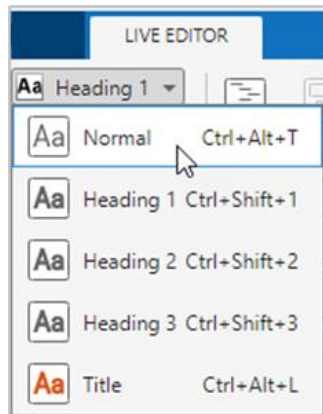
- From the **HOME** tab, choose the **New Live Script** option:



- Use the **Text** button to add formatted text to a MATLAB live script.



- You can select different formatting options for text:



- Enter some text, and format it to indicate that this section is for importing the data. For example:

**Import raw data**  
Read data from Excel

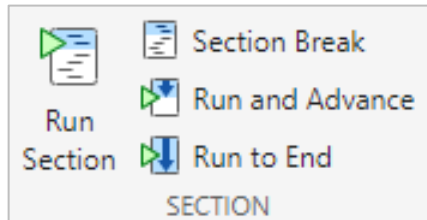
- Use the **Code** button to add MATLAB code.



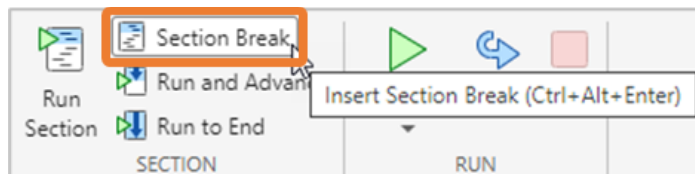
- Add the following command to reuse the `importfile` function to import the data:  
`FlightData = importfile('FlightData.xlsx')`



In addition to including comments in the script, you can further organize your scripts by dividing it into smaller sections. Besides serving as an organization tool, the code section feature in MATLAB also helps with prototyping your analysis. Dividing your script into smaller sections gives you the option to focus on and execute only a subset of the analysis, without evaluating the rest. Use these options to control which portions of the script are run, and whether to advance or not.



- Use the Section Break button to add a new code section.



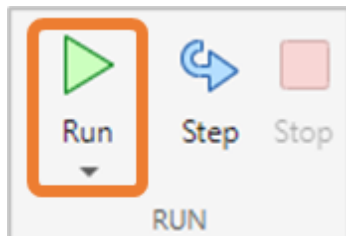
- Add some **text** to indicate that this section is focused on visualizations.
- Add the following **code** for the `geoplot`:

```
geoplot(FlightData.LatitudePosition,FlightData.LongitudePosition)
title("LongitudePosition vs LatitudePosition")
legend("show")
```

### Step 5 – Use Indexing to Filter the Data

Catch Up File: [low\\_code\\_step5.mlx](#)

- Run your entire script by clicking the **Run** button in the Live Editor tab:



This will ensure that you can interactively manipulate the table in the import section.

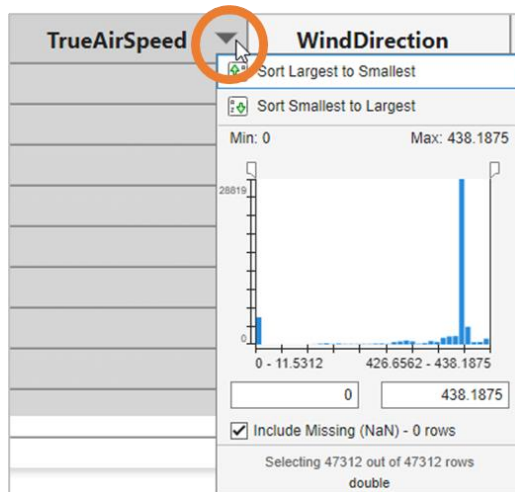
- Scroll to the right of the table until the `TrueAirSpeed` variable is visible:

## Introduction to MATLAB for Data Analysis – A Hands-On Workshop

FlightData = 47312x13 table

	FuelFlow	FanSpeed	TrueAirSpeed	WindDirection	WindSp
1	0	1.5000	0	0	
2	0	1.5000	0	0	
3	0	1.5000	0	0	
4	0	1.5000	0	0	
5	0	1.5000	0	0	
6	0	1.5000	0	0	
7	0	1.5000	0	0	
8	0	1.5000	0	0	
9	0	1.5000	0	0	

- Use the dropdown option to open the options for interacting with the table output:



- Filter out the rows when the plane is not moving by:
  - Setting the minimum value to 10
  - Unchecking the option to include missing values
  - Choosing the option to Update Code
    - This last step will update the script with the code necessary to replicate this data filtering.

## Introduction to MATLAB for Data Analysis – A Hands-On Workshop

FlightData = 42584x13 table | Filtered from 47312 rows

	FuelFlow	FanSpeed	TrueAirSpeed	WindDirection	WindSp
1	2632	90.8750	100.12		
2	2632	90.8750	100.18		
3	2632	90.8750	101.75		
4	2632	90.8750	102.18		
5	2632	90.8750	102.81		
6	2632	90.9062	102.81		
7	2632	90.9375	104.87		
8	2632	91	105.50		
9	2632	90.9375	107.25		

Code ^

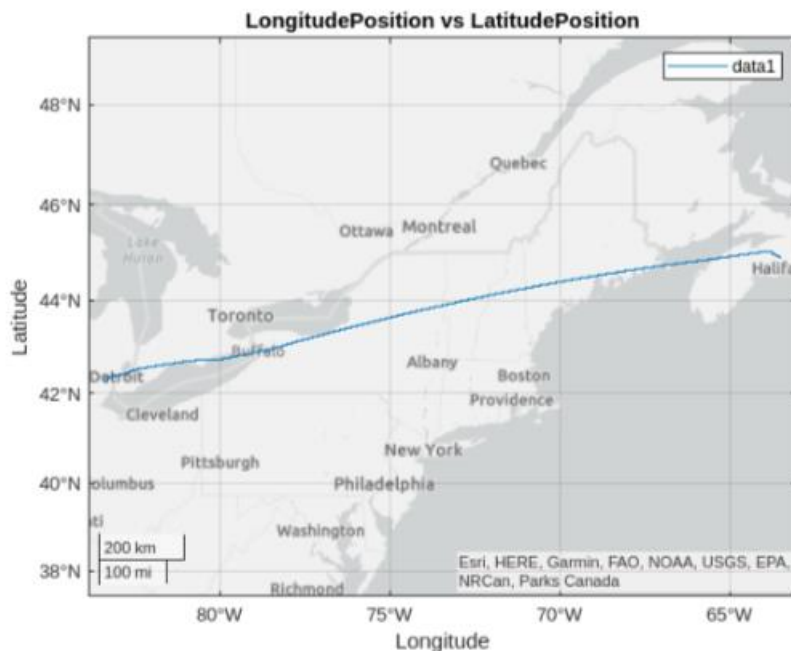
```
FlightData = FlightData(FlightData.TrueAirSpeed >= 10,:)
```

Update Code

- Choose the **Run and Advance** option in the **SECTION** portion of the **LIVE EDITOR** tab for the current section.
- Choose the **Run and Advance** option in the **SECTION** portion of the **LIVE EDITOR** tab for the next section of code, which generates the **geoplot**. You should see the outlier data is no longer plotted, as it has been filtered out.

## Visualize Data

```
geoplot(FlightData.LatitudePosition,FlightData.LongitudePosition)|
title("LongitudePosition vs LatitudePosition")
legend("show")
```

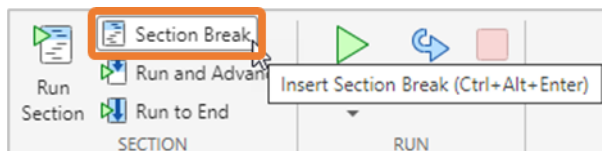


## Step 6 – Clean Outlier Data, Using a Live Editor Task

Catch Up File: [low\\_code\\_step6.mlx](#)

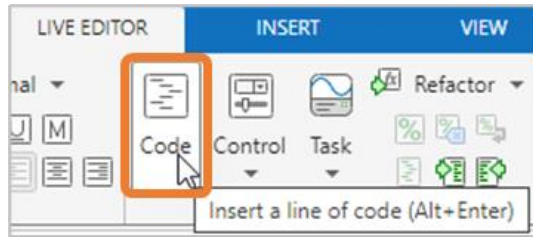
Live Editor tasks are simple point-and-click interfaces that can be added to a live script to perform a specific set of operations. Through these tasks you can explore and see immediate results of your selected parameters as well as automatically generate the corresponding MATLAB code. We will use the **Clean Outlier Data** task in this section to review and clean outliers for two of the variables in our dataset.

- Insert a **section break** in the script:

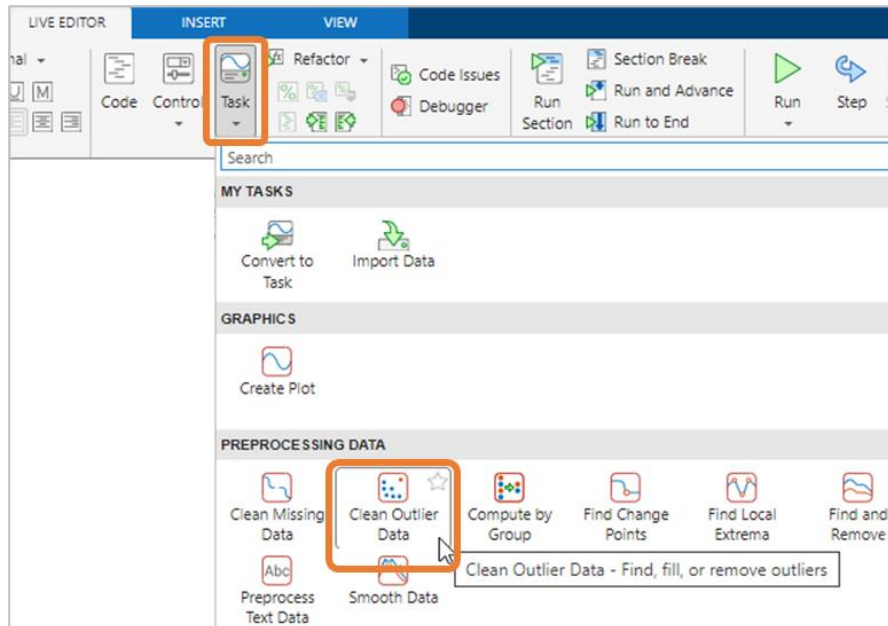


- Ensure you are in **text** mode, and then add some text to indicate that this section is about cleaning the data, focusing on the outliers we observed earlier.
- Switch to **Code** mode:

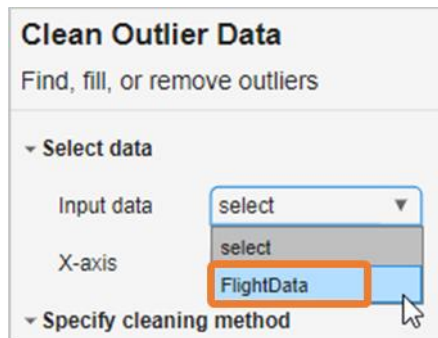
## Introduction to MATLAB for Data Analysis – A Hands-On Workshop



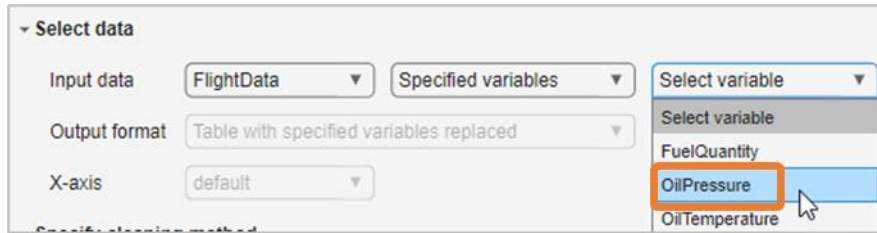
- Insert a **Clean Outlier Data** Live Editor task in the script:



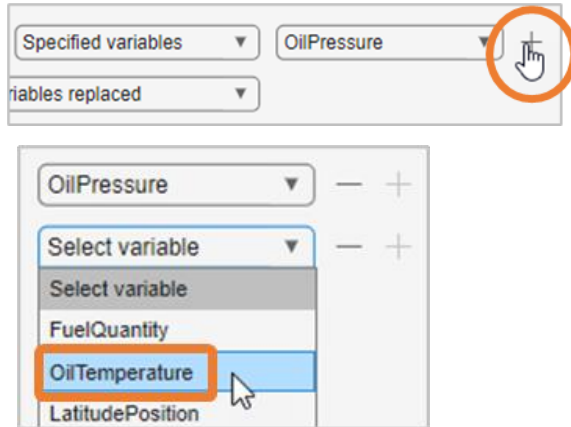
- **Data** - Choose **FlightData** as the input data:



- **Variables**
  - Select **OilPressure** as the first variable:

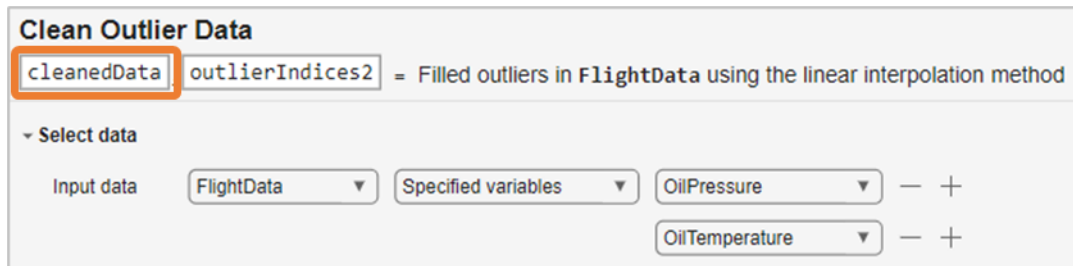


- Add a second variable by selecting the + button, and then selecting **OilTemperature**:



## ▪ Output

- Change the name of the output variable (e.g., to **cleanedData**)



## ▪ Options

- Cleaning Method: Use the default values
- Outliers > Detection Method: Use the default values
- Outliers > Moving Window
  - Use the default window type as "Centered"
  - Set the window length to be 100, centered about the current point



Specify cleaning method

Cleaning method: Fill outliers, Linear interpolation

Define outliers

Detection method: Moving median, Threshold factor: 3

Moving window: Centered, 100

Display results

Window units centered about current element

## Results / Visualizations

- Choose the “**Select all**” option to display all variables

Display results

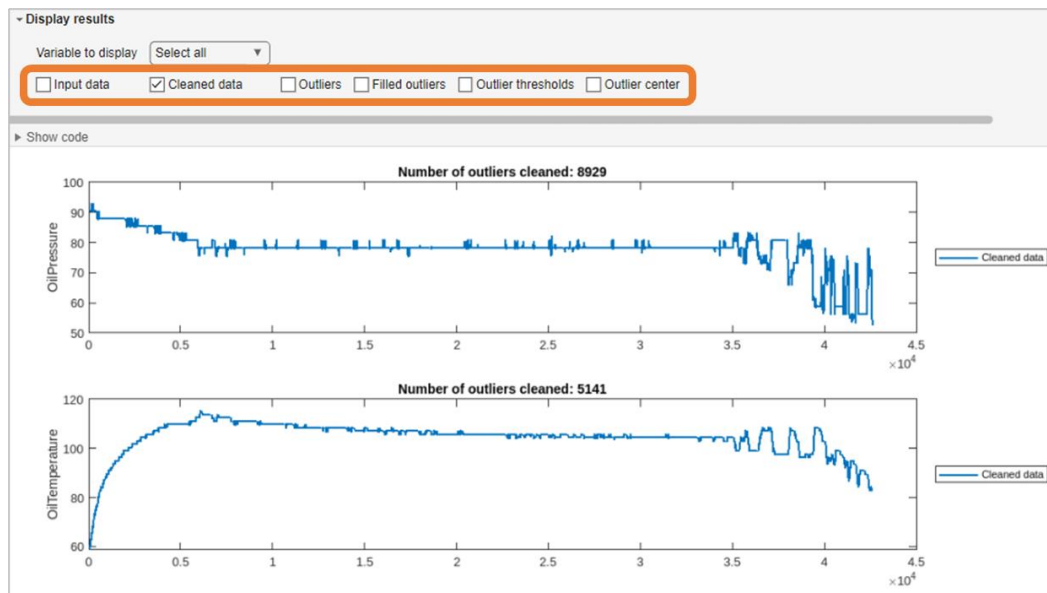
Variable to display: OilPressure

Input data: ☒

Show code

Select all

- Toggle elements on and off to more clearly see the input data, the cleaned data, and information about the outliers.



## Step 7 – Develop a Predictive Model

Catch Up File: [low\\_code\\_step7.mlx](#)

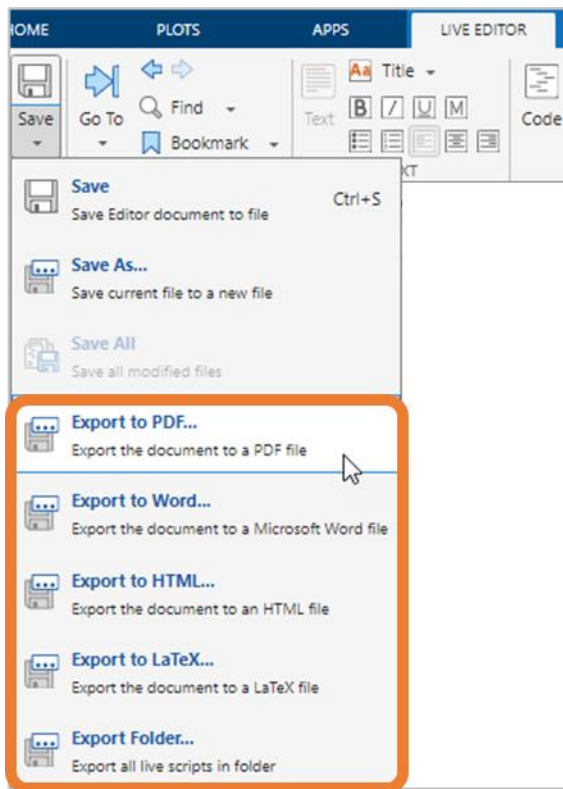
For this final step of the analysis, the presenter will do a demo of the Regression Learner app for developing a model to predict the true air speed. It is not expected that attendees will follow along during this step.

## Step 8 – Final Script and Report

A final version of our script, with additional comments and formatting, and the results of our predictive model, can be found in the final folder: [Final/VirtualSensorModel.mlx](#)

This script could be shared with others directly, allowing them to run the code themselves.

You could also create a static document or report of your analysis directly from the live script. To do this, simply choose one of the export options from the Save dropdown:



The selections provide a well-formatted report that captures the steps of our analysis, as well as the generated results and figures.

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See [www.mathworks.com/trademarks](http://www.mathworks.com/trademarks) for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.