MATLAB EXPO 2019

Becoming a Data-Centric Engineering Team

Paul Peeling
A path for how your team can better explore, understand and analyze data.
Data Science Maturity Levels

Ad-hoc Individual Analysis

Generally Useful Tools for Analysis

Common Infrastructure; Tested and Documented

Ease of scaling to more people

Cost of asking a question
Data Science Maturity Levels

Ad-hoc Individual Analysis

Generally Useful Tools for Analysis

Common Infrastructure; Tested and Documented

- Goal is to be fast: reduce time to insight
Overview of Flight Data

- **35** unique aircraft
- **180,000** unique flights
- **300 GB** of data

Source:
- NASA Dash Link: Sample Flight Data
  - [https://c3.nasa.gov/dashlink/projects/85/](https://c3.nasa.gov/dashlink/projects/85/)
**Big Data** Creates Opportunities

- Find rare events, then dive deeper
- Build and validate test scenarios that match real-world conditions
- Perform fleet-wide calculations
Exploring a New Dataset

1. Control data import with ImportOptions
2. Working with table and timetable datatypes
3. Interacting with data in the Live Editor
4. Filling in outliers with a Live Task
5. Synchronizing time-based data
6. New geographic visualizations

```matlab
% Example code

t = synchronize(t1hz,t4hz);
t = fillmissing(t,'previous');
inFlight = t.WOW==1;
geoplot(t.LATP(inFlight),t.LONP(inFlight),'LineWidth',3);
```
Getting Started: Exploring a New Dataset
Getting Started: **Exploring a New Dataset**

**Import Data**

`Import1HzData`

`flightData1Hz = 4636×89 table`

<table>
<thead>
<tr>
<th>Time</th>
<th>ABRK</th>
<th>ACMT</th>
<th>AIL_1</th>
<th>AIL_2</th>
<th>ALTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-May-20...</td>
<td>119.9836</td>
<td>59</td>
<td>91.8789</td>
<td>91.5925</td>
<td>6000</td>
</tr>
<tr>
<td>10-May-20...</td>
<td>119.9836</td>
<td>59</td>
<td>91.8994</td>
<td>91.5925</td>
<td>6000</td>
</tr>
<tr>
<td>10-May-20...</td>
<td>119.9836</td>
<td>60</td>
<td>91.8994</td>
<td>91.5925</td>
<td>6000</td>
</tr>
<tr>
<td>10-May-20...</td>
<td>119.9836</td>
<td>59</td>
<td>91.8994</td>
<td>91.5720</td>
<td>6000</td>
</tr>
<tr>
<td>10-May-20...</td>
<td>119.9836</td>
<td>60</td>
<td>91.8789</td>
<td>91.5720</td>
<td>6000</td>
</tr>
<tr>
<td>10-May-20...</td>
<td>119.9836</td>
<td>60</td>
<td>91.9607</td>
<td>91.5720</td>
<td>6000</td>
</tr>
<tr>
<td>10-May-20...</td>
<td>119.9836</td>
<td>59</td>
<td>91.9198</td>
<td>91.5516</td>
<td>6000</td>
</tr>
<tr>
<td>10-May-20...</td>
<td>119.9836</td>
<td>60</td>
<td>91.8789</td>
<td>91.5925</td>
<td>6000</td>
</tr>
<tr>
<td>10-May-20...</td>
<td>119.9836</td>
<td>59</td>
<td>91.8175</td>
<td>91.5516</td>
<td>6000</td>
</tr>
</tbody>
</table>
Getting Started: **Exploring a New Dataset**

![Dataset Exploration Diagram](image-url)
Getting Started: **Exploring a New Dataset**

```
flightData1Hz = flightData1Hz(1:9,:);

Sort Largest to Smallest
Sort Smallest to Largest

Min: 60
Max: 119.9836

80.3635 119.9836

Include Missing (NaN) - 0 rows
```
Getting Started: Exploring a New Dataset

Slicing the data
Access rows or columns of a table:

```
flightData1Hz(1,:)
```

```
ans = 1x89 table

<table>
<thead>
<tr>
<th>Time</th>
<th>ABRK</th>
<th>ACMT</th>
<th>AIL_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-May-20</td>
<td>119.9836</td>
<td>59</td>
<td>01.8789</td>
</tr>
</tbody>
</table>
```

```
flightData1Hz(:,1)
```

```
ans = 4636x1 table

<table>
<thead>
<tr>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-May-20</td>
</tr>
</tbody>
</table>
```

Accessing a variable:
```
flightData1Hz.ABRK
```

```
ans = 4636x1

119.9836
119.9836
119.9836
119.9836
119.9836
119.9836
119.9836
```

Access variables by type:
```
flightData1Hz(:,vartype("numeric"))
```

```
ans = 4636x88

<table>
<thead>
<tr>
<th>0.0001</th>
<th>0.0001</th>
<th>0.0001</th>
<th>0.0001</th>
</tr>
</thead>
</table>
```

```
flightData1Hz(:,vartype("datetime"))
```

```
ans = 4636x1 datetime array

10-May-2001 17:03:54
10-May-2001 17:03:55
10-May-2001 17:03:56
```

Getting Started: **Exploring a New Dataset**

```matlab
1hz = table2timetable(flightData1Hz)
```

```matlab
1hz = 4636×88 timetable
```

<table>
<thead>
<tr>
<th>Time</th>
<th>ABRK</th>
<th>ACMT</th>
<th>AIL_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-May-20...</td>
<td>119.9836</td>
<td>59</td>
<td>91.8789</td>
</tr>
<tr>
<td>10-May-20...</td>
<td>119.9836</td>
<td>59</td>
<td>91.8994</td>
</tr>
<tr>
<td>10-May-20...</td>
<td>119.9836</td>
<td>60</td>
<td>91.8994</td>
</tr>
<tr>
<td>10-May-20...</td>
<td>119.9836</td>
<td>59</td>
<td>91.8994</td>
</tr>
<tr>
<td>10-May-20...</td>
<td>119.9836</td>
<td>60</td>
<td>91.8789</td>
</tr>
<tr>
<td>10-May-20...</td>
<td>119.9836</td>
<td>60</td>
<td>91.9607</td>
</tr>
<tr>
<td>10-May-20...</td>
<td>119.9836</td>
<td>59</td>
<td>91.9198</td>
</tr>
<tr>
<td>10-May-20...</td>
<td>119.9836</td>
<td>60</td>
<td>91.8789</td>
</tr>
<tr>
<td>10-May-20...</td>
<td>119.9836</td>
<td>59</td>
<td>91.8175</td>
</tr>
</tbody>
</table>

Sorted Newest to Oldest

Sorted Oldest to Newest

Min: 10-May-2001 17:03:54
Max: 10-May-2001 18:21:09

Include Missing (NaN) - 0 rows
Getting Started: **Exploring a New Dataset**

Index a timetable using time.

```matlab
starttime = datetime(2001,5,10,17,30,0);
endtime = datetime(2001,5,10,17,30,5);
t1hz(timerange(starttime,endtime),:)
```

```plaintext
ans = 5×88 timetable

<table>
<thead>
<tr>
<th>Time</th>
<th>ABRK</th>
<th>ACMT</th>
<th>AIL_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 10-May-20</td>
<td>119.9836</td>
<td>56</td>
<td>83.7775</td>
</tr>
<tr>
<td>2 10-May-20</td>
<td>119.9836</td>
<td>56</td>
<td>83.8184</td>
</tr>
<tr>
<td>3 10-May-20</td>
<td>119.9836</td>
<td>0</td>
<td>83.8798</td>
</tr>
<tr>
<td>4 10-May-20</td>
<td>119.9836</td>
<td>56</td>
<td>84.0230</td>
</tr>
<tr>
<td>5 10-May-20</td>
<td>119.9836</td>
<td>56</td>
<td>83.8389</td>
</tr>
</tbody>
</table>
```
Getting Started: **Exploring a New Dataset**

```matlab
stackedplot(t1hz,{'OIT_1','OIT_2','OIT_3','OIT_4'});
```

![Graph showing time series data with timestamps and values for OIT_1 to OIT_4](image)
Getting Started: Exploring a New Dataset
Getting Started: Exploring a New Dataset

Clean Outlier Data
Find, fill, or remove outliers

Select data
Input data: select, X-axis: default

Specify cleaning method
Cleaning method: Fill outliers

Define outliers
Detection method: Median, Threshold factor: 3

Visualize results
Filled outliers, Cleaned data, Input data, Outliers, Outlier thresholds, Outlier center
**Getting Started: Exploring a New Dataset**

**Clean Outlier Data**

`cleanedData = ` Filled outliers in `t1hz.OIT_1` using the linear interpolation method

**Specify cleaning method**

Cleaning method: Fill outliers, Linear interpolation

**Define outliers**

Detection method: Median, Threshold factor: 3

**Visualize results**

- Filled outliers
- Cleaned data
- Input data
- Outliers
- Outlier thresholds
- Outlier center
Getting Started: Exploring a New Dataset

% Visualize results
clf
plot(tihz.Time,tihz.OIT_1,'Color',[100 185 226]/255,...
     'DisplayName','Input data')
hold on
plot(tihz.Time,cleandata,'Color',[0 114 189]/255,'LineWidth',1.5,...
     'DisplayName','Cleaned data')

% Plot outliers
plot(tihz.Time(outlierIndices),tihz.OIT_1(outlierIndices),'+',...
     'Color',[64 64 64]/255,'DisplayName','Outliers')
title(['Number of outliers: ' num2str(nnz(outlierIndices))])

% Plot filled outliers
plot(tihz.Time(outlierIndices),cleandata(outlierIndices),'.',...
     'MarkerSize',12,...
     'Color',[217 83 251]/255,'DisplayName','Filled outliers')

% Plot outlier thresholds
plot([xlim missing xlim],[thresholdLow[1 1] NaN thresholdHigh[1 1]],...
     'Color',[145 145 145]/255,'DisplayName','Outlier thresholds')
Getting Started: Exploring a New Dataset

![Image of MATLAB interface for defining outliers]

```matlab
% Define outliers
Detection method = 'Mean'; Threshold factor = 3;

% Visualize results
% Fill outliers
[cleanedData, outlierIndices, thresholdLow, thresholdHigh] = ...
    filloutliers(t1hz.OIT_1, 'linear', 'mean', 'SamplePoints', t1hz.Time);
```
Getting Started: **Exploring a New Dataset**

- **Missing Data**
  - `ismissing`
  - `rmmissing`
  - `fillmissing`

- **Outliers**
  - `isoutlier`
  - `rmoutliers`
  - `filloutliers`

- **Change Points**
  - `ischange`

- **Noisy Data**
  - `smoothdata`

- **and more…**

Getting Started: Exploring a New Dataset

Apply outlier filling method to all oil temperature variables in our timetable.

```matlab
thz = filloutliers(thz,'linear','mean','DataVariables',['OIT_1','OIT_2','OIT_3','OIT_4']);

stackedplot(thz,{{'OIT_1','OIT_2'},{'OIT_3','OIT_4'}});
```
Getting Started: **Exploring a New Dataset**

```matlab
t4hz = table2timetable(flightData4Hz)
```

<table>
<thead>
<tr>
<th>Time</th>
<th>ALT</th>
<th>ALTR</th>
<th>AOA1</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-May-20...</td>
<td>861</td>
<td>16</td>
<td>6.5918</td>
</tr>
<tr>
<td>10-May-2001 17:03:54.250001</td>
<td>-16</td>
<td>6.5918</td>
<td></td>
</tr>
<tr>
<td>10-May-20...</td>
<td>861</td>
<td>-16</td>
<td>6.5918</td>
</tr>
<tr>
<td>10-May-20...</td>
<td>861</td>
<td>-16</td>
<td>6.5918</td>
</tr>
<tr>
<td>10-May-20...</td>
<td>861</td>
<td>0</td>
<td>6.5918</td>
</tr>
<tr>
<td>10-May-20...</td>
<td>861</td>
<td>0</td>
<td>6.5918</td>
</tr>
<tr>
<td>10-May-20...</td>
<td>861</td>
<td>-16</td>
<td>6.5918</td>
</tr>
<tr>
<td>10-May-20...</td>
<td>861</td>
<td>0</td>
<td>6.5918</td>
</tr>
<tr>
<td>10-May-20...</td>
<td>861</td>
<td>0</td>
<td>6.5918</td>
</tr>
</tbody>
</table>
Getting Started: Exploring a New Dataset

Join Tables

joinedData = Combine flightData1Hz and flightData4Hz using outer join

Select data

Left table \(\text{flightData1Hz}\) \(\Rightarrow\) Right table \(\text{flightData4Hz}\)
Merging variable \(\text{Time}\) \(\Rightarrow\) Merging variable \(\text{Time}\)

Specify join

- Outer join
- Left outer join
- Right outer join
- Inner join
- Join

Combine merging variables

Visualize results

% Join tables

\[
\text{joinedData} = \text{outerjoin} (\text{flightData1Hz}, \text{flightData4Hz}, 'Keys', 'Time', ...
\text{'MergeKeys'}, \text{true})
\]
Getting Started: Exploring a New Dataset

Synchronize 1Hz and 4Hz Data
Join the 1Hz and 4Hz data using the synchronize command.

The synchronize command gives us flexibility in how the synchronize occurs.

Here, we use the default synchronize method which synchronizes the data onto a union of the timestamps from t1hz and t4hz.

\[
t = \text{synchronize(t1hz,t4hz)}
\]

<table>
<thead>
<tr>
<th></th>
<th>Time</th>
<th>ABRK</th>
<th>AGMT</th>
<th>AIL_1</th>
<th>AIL_2</th>
<th>ALTS</th>
<th>APFD</th>
<th>ATEN</th>
<th>AL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10-May-20...</td>
<td>119.9836</td>
<td>59</td>
<td>91.8789</td>
<td>91.5925</td>
<td>6000</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10-May-20...</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>3</td>
<td>10-May-20...</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>4</td>
<td>10-May-20...</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>5</td>
<td>10-May-20...</td>
<td>119.9836</td>
<td>59</td>
<td>91.8994</td>
<td>91.5925</td>
<td>6000</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10-May-20...</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>7</td>
<td>10-May-20...</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>8</td>
<td>10-May-20...</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>9</td>
<td>10-May-20...</td>
<td>119.9836</td>
<td>60</td>
<td>91.8994</td>
<td>91.5925</td>
<td>6000</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Getting Started: **Exploring a New Dataset**

```matlab
t = fillmissing(t, 'previous')
```

$t = 18544 \times 137$ timetable

<table>
<thead>
<tr>
<th>Time</th>
<th>ABRK</th>
<th>ACMT</th>
<th>AIL_1</th>
<th>AIL_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10-May-20...</td>
<td>119.9836</td>
<td>59</td>
<td>91.8789</td>
</tr>
<tr>
<td>2</td>
<td>10-May-20...</td>
<td>119.9836</td>
<td>59</td>
<td>91.8789</td>
</tr>
<tr>
<td>3</td>
<td>10-May-20...</td>
<td>119.9836</td>
<td>59</td>
<td>91.8789</td>
</tr>
<tr>
<td>4</td>
<td>10-May-20...</td>
<td>119.9836</td>
<td>59</td>
<td>91.8789</td>
</tr>
<tr>
<td>5</td>
<td>10-May-20...</td>
<td>119.9836</td>
<td>59</td>
<td>91.8994</td>
</tr>
<tr>
<td>6</td>
<td>10-May-20...</td>
<td>119.9836</td>
<td>59</td>
<td>91.8994</td>
</tr>
<tr>
<td>7</td>
<td>10-May-20...</td>
<td>119.9836</td>
<td>59</td>
<td>91.8994</td>
</tr>
<tr>
<td>8</td>
<td>10-May-20...</td>
<td>119.9836</td>
<td>59</td>
<td>91.8994</td>
</tr>
<tr>
<td>9</td>
<td>10-May-20...</td>
<td>119.9836</td>
<td>60</td>
<td>91.8994</td>
</tr>
</tbody>
</table>
Getting Started: Exploring a New Dataset

```matlab
inFlight = t.WOW==1;
geoplot(t.LATP(inFlight),t.LONP(inFlight), 'LineWidth', 3);
```

https://www.mathworks.com/help/matlab/geographic-plots.html
Getting Started: **Exploring a New Dataset**

![Map with coordinates](image1)

```
ax = gca;
chart = ax.Children(1);
datatip(chart,42.25,-83.35);
```

![Map with coordinates and code](image2)

```
coolimits([33.2 47.5],[-96.0 -72.2])
```

![Map with coordinates and code](image3)
Data Science Maturity Levels

- Ad-hoc Individual Analysis
- Generally Useful Tools for Analysis
- Common Infrastructure; Tested and Documented

- Explore and understand data
- Document analysis
- Tools will be re-used in next steps
Data Science Maturity Levels

Ad-hoc Individual Analysis

- Apply to different datasets
  - Functions/Scripts
  - MATLAB Apps

Generally Useful Tools for Analysis

Common Infrastructure; Tested and Documented

• Trend: Work with **BIG DATA**
Reusable Tools for Big Data Analysis

1. Custom datastore for reading all the data
2. Extension of the datastore for specific queries
3. Extending datastore with tall variables

```matlab
ds = flightDataStore('\Data\666\');
ds.SelectedVariableNames = {'FIRE_2'};
ds.AddFlightIdentifier = true;
while hasdata(ds)
    t = read(ds);
    if any(t.FIRE_2)
        results = [results; {t.Flight(1), sum(t.FIRE_2)}];
    end
end
```

```
t = tall(ds)
t = Mx6 tall timetable
binEdges = 0:80;
histogram(t.FE,binEdges);
```
Big Data Requires New Tools

Create a datastore from all CSV files

```matlab
ds = datastore('*.csv')
```

Read a single file of data

```matlab
data = read(ds);
```

Reset the datastore back to the first file

```matlab-reset(ds);
```

Find the maximum value of “Y” in each file

```matlab
X = [];
while hasdata(ds)
    data = read(ds);
    X(end+1) = max(data.Y);
end
```

---

<table>
<thead>
<tr>
<th>Built-In Datastores</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General</strong></td>
</tr>
<tr>
<td>datastore</td>
</tr>
<tr>
<td>spreadsheetDatastore</td>
</tr>
<tr>
<td>tabularTextDatastore</td>
</tr>
<tr>
<td>fileDatastore</td>
</tr>
<tr>
<td><strong>Database</strong></td>
</tr>
<tr>
<td>databaseDatastore</td>
</tr>
<tr>
<td><strong>Image</strong></td>
</tr>
<tr>
<td>imageDatastore</td>
</tr>
<tr>
<td>denoisingImageDatastore</td>
</tr>
<tr>
<td>randomPatchExtractionDatastore</td>
</tr>
<tr>
<td>pixelLabelDatastore</td>
</tr>
<tr>
<td>augmentedImageDatastore</td>
</tr>
<tr>
<td><strong>Audio</strong></td>
</tr>
<tr>
<td>audioDatastore</td>
</tr>
<tr>
<td><strong>Predictive Maintenance</strong></td>
</tr>
<tr>
<td>fileEnsembleDatastore</td>
</tr>
<tr>
<td>simulationEnsembleDatastore</td>
</tr>
<tr>
<td><strong>Simulink</strong></td>
</tr>
<tr>
<td>SimulationDatastore</td>
</tr>
<tr>
<td><strong>Automotive</strong></td>
</tr>
<tr>
<td>mdfDatastore</td>
</tr>
</tbody>
</table>
Big Data Requires New Tools

Custom Datastore

- Customize a datastore to work with your dataset
- Gives you control over how data is loaded and formatted
- MATLAB subclass: “fill-in-the-blanks”
- Build a piece of infrastructure, then re-use it in your analyses

```matlab
function [data,info] = read(ds)
    ...
end

function tf = hasdata(ds)
    ...
end

function reset(ds)
    ...
end

function p = progress(ds)
    ...
end

function data = readall(ds)
    ...
end
```
A Custom Datastore for Flight Data

```matlab
ds = flightDataStore('Data\666');
ds.SelectedVariableNames = {'FIRE_2'};
ds.AddFlightIdentifier = true;
```

```
ds = flightDataStore with properties:
  CurrentFileIndex: 1
  NumberOfFiles: 6691
  VariableNames: {1x186 cell}
  SelectedVariableNames: {'FIRE_2'}
  AddFlightIdentifier: 1
```

```
preview(ds)
```

```
ans = 8×2 timetable
    Time       FIRE_2       Flight
   1   01-Apr-200...     0  666200104...
   2   01-Apr-200...     0  666200104...
   3   01-Apr-200...     0  666200104...
   4   01-Apr-200...     0  666200104...
   5   01-Apr-200...     0  666200104...
   6   01-Apr-200...     0  666200104...
   7   01-Apr-200...     0  666200104...
   8   01-Apr-200...     0  666200104...
```
A Custom Datastore for Flight Data

t = readFullFlight(ds, '666200303212002')

t = 44032x187 timetable

<table>
<thead>
<tr>
<th>Time</th>
<th>ACID</th>
<th>DATE_DAY</th>
<th>DATE_MONTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 21-Mar-200...</td>
<td>666</td>
<td>21</td>
<td>3</td>
</tr>
<tr>
<td>2 21-Mar-200...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 21-Mar-200...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 21-Mar-200...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 21-Mar-200...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 21-Mar-200...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 21-Mar-200...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8 21-Mar-200...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 21-Mar-200...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

signalMetaInfo(ds)

ans = 186x5 table

<table>
<thead>
<tr>
<th>Signals</th>
<th>Rate</th>
<th>Units</th>
<th>Description</th>
<th>Alpha</th>
</tr>
</thead>
<tbody>
<tr>
<td>'ABRK'</td>
<td>1.0000</td>
<td>'DEG'</td>
<td>'AIRBRAKE PO...</td>
<td>'ABRK'</td>
</tr>
<tr>
<td>'ACID'</td>
<td>0.2500</td>
<td></td>
<td>'AIRCRAFT NU...</td>
<td>'ACID'</td>
</tr>
<tr>
<td>'ACMT'</td>
<td>1.0000</td>
<td></td>
<td>'ACMS TIMING...</td>
<td>'ACMT'</td>
</tr>
<tr>
<td>'AIL_1'</td>
<td>1.0000</td>
<td>'DEG'</td>
<td>'AILERON POS...</td>
<td>'AIL.1'</td>
</tr>
<tr>
<td>'AIL_2'</td>
<td>1.0000</td>
<td>'DEG'</td>
<td>'AILERON POS...</td>
<td>'AIL.2'</td>
</tr>
<tr>
<td>'ALT'</td>
<td>4.0000</td>
<td>'FEET'</td>
<td>'PRESSURE A...</td>
<td>'ALT'</td>
</tr>
<tr>
<td>'ALTR'</td>
<td>4.0000</td>
<td>'FT/MIN'</td>
<td>'ALTITUDE RA...</td>
<td>'ALTR'</td>
</tr>
<tr>
<td>'ALTS'</td>
<td>1.0000</td>
<td>'FEET'</td>
<td>'SELECTED AL...</td>
<td>'ALTS'</td>
</tr>
<tr>
<td>'AOA1'</td>
<td>4.0000</td>
<td>'DEG'</td>
<td>'ANGLE OF AT...</td>
<td>'AOA1'</td>
</tr>
</tbody>
</table>
A Custom Datastore for Flight Data

Read a file from the Dash Link Flight Data

https://c3.nasa.gov/dashlink/projects/85/

```
function tt = readFlightFile(filename, varnames, addidentifier)

% Load the data
if nargin==1
    % Load the entire file
    s = load(filename);
else
    % Load only the specified variables
    if ~isempty(varnames)
        s = load(filename, varnames{:});
    else
        s = struct([]);
    end
end
```

ExtractStartTime

Extract the flight start time from the flight identifier (located in the flight filename). The flight identifier is of the form:

NNNyyyyMMddhhmm

Where NNN is the tail number, and yyyyMMddhhmm is a timestamp.

Note: Some files have a trailing "_#" (in the case where there are multiple files from the same aircraft that started at the same minute).

```
function st = extractStartTime(s, flightIdentifier)

    % Handle files that have an underscore and numeric identifier
    if contains(flightIdentifier, '_')
        flightIdentifier = extractBefore(flightIdentifier, '_');
    end

    % The starting second is not in the flight identifier, but we can get it from the raw data
    sec = s.GMT_SEC.data(1);

    % Add the seconds and the timestamp together to get the start time
    st = datetime(flightIdentifier(4:end),...
                  'InputFormat', yyyyMMddhhmm',...
                  'TimeZone','Europe/London')...
         + seconds(sec);
```

A Custom Datastore for Flight Data

```matlab
classdef flightDataStore < matlab.io.Datastore & ...
    matlab.io.datastore.Partitionable & ...
    matlab.io.datastore.HadoopFileBased

    properties (SetAccess = protected)
        CurrentFileIndex double
        NumberOfFiles double
        VariableNames cell
    end

    properties (Dependent)
        SelectedVariableNames cell
    end

    properties
        AddFlightIdentifier logical
    end

    properties (Access = protected, Hidden = true)
        FileSet matlab.io.datastore.DsFileSet
    end

    function fds = flightDataStore(location)
        %FLIGHTDATASTORE Datastore for reading flight data
        % DS = FLIGHTDATASTORE(LOCATION) creates a flightDataStore DS
        % of the data. LOCATION is a folder that contains .MAT files
        % that contain flight data. Alternatively, LOCATION can be
        % an individual .MAT file of flight data. If LOCATION is a
        % folder, .MAT files are searched for recursively (there can
        % be multiple levels of folders).

        % Use the DsFileSet class to find and keep track of files
        fds.FileSet = matlab.io.datastore.DsFileSet(location,...
            'FileExtensions','.mat',...'
            'FileSplitSize','file',...
            'IncludeSubfolders',true);
        fds.CurrentFileIndex = 1;
        fds.AddFlightIdentifier = false;

        fds.NumberOfFiles = fds.FileSet.NumFiles;
        fds.VariableNames = extractVariableNames(fds);
        fds.SelectedVariableNames = fds.VariableNames;

        reset(fds);
    end
```
A Custom Datastore for Flight Data

%% Basic datastore implementation
function [data,info] = read(fds)  
  % Read data and information about the extracted data.
  % Info about the next file to read
  fileInfoTbl = nextfile(fds.FileSet);

  % Call the file reader
  data = readFlightFile(fileInfoTbl.FileName,fds.SelectedVariables);

  % Return info about the data that has been read
  info.Size = size(data);
  info.FileName = fileInfoTbl.FileName;
  info.Offset = fileInfoTbl.Offset;

  % Update CurrentFileIndex for tracking progress
  fds.CurrentFileIndex = fds.CurrentFileIndex + 1;
end

%% Custom methods for reading individual flight data
function data = readFullFlight(fds,flightIdentifier)  
function data = readFlight(fds,flightIdentifier)...

%% Extract metadata that is helpful to understand the dataset
function signalInfo = signalMetaInfo(fds)...

%% Support for parallel computing
function subds = partition(fds,n,i)...

%% Support for Hadoop
function initializeDatastore(fds,hadoopInfo)...
function loc = getLocation(fds)
Find Rare Events, then Dive Deeper

```
results = table('Size', [0 2], 'VariableTypes', { 'categorical', 'double' }, ...
    'VariableNames', {'Flight', 'Fire_Seconds'});
while hasdata(ds) 
    t = read(ds);
    if any(t.FIRE_2)
        results = [results; {t.Flight(1), sum(t.FIRE_2)}];
    end
end
```
Find Rare Events, then Dive Deeper

```
stackedplot(t,contains(t.Properties.VariableNames,'FIRE_'));
plot(t.LONP,t.LATP);
plot(t.LONP(logical(t.FIRE_2)),t.LATP(logical(t.FIRE_2)),'o');
xlabel('Longitude');
ylabel('Latitude');
legend('Flight Path','Fire Event');
```
Find Rare Events, then Dive Deeper

```
varidx = contains(t.Properties.VariableNames,{'FIRE_2','RALT','FF','N2','MW'});
stackedplot(t,varidx);
```
Find Rare Events, then Dive Deeper
Big Data Requires New Tools

Tall Arrays

\[ t = \text{tall}(ds) \]

\[ t = \]

Mx6 \textit{tall timetable}

- When to use tall
  - the function has already been implemented
  - the output of the operation does not fit in memory
  - multiple independent outputs are required
  - the algorithm works over a moving window
Perform Fleet-Wide Calculations

TAS: True Airspeed (knots)
FF_N: Fuel Flow N (lbs/hr)
WOW: Weight On Wheels (logical)

```matlab
ds.SelectedVariableNames = {'TAS','FF_1','FF_2','FF_3','FF_4','WOW'}
```

```matlab
t = tall(ds)
```

```matlab
No6 tall timetable
```

<table>
<thead>
<tr>
<th>Time</th>
<th>WOW</th>
<th>TAS</th>
<th>FF_1</th>
<th>FF_2</th>
<th>FF_3</th>
<th>FF_4</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-Apr-2001 03:15:30</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>01-Apr-2001 03:15:39</td>
<td>NaN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>01-Apr-2001 03:15:39</td>
<td>NaN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>01-Apr-2001 03:15:31</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>01-Apr-2001 03:15:31</td>
<td>NaN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>01-Apr-2001 03:15:31</td>
<td>NaN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>01-Apr-2001 03:15:31</td>
<td>NaN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Limit to only cases where the aircraft was in the air
```matlab
inAir = t.WOW==1;
t = t(inAir,:);
```

Sum the total fuel flow to the engine (units are LBS/HR)
```matlab
t.FF_TOTAL = t.FF_1+t.FF_2+t.FF_3+t.FF_4;
```

Convert airspeed to MPH and calculate fuel efficiency in LBS/MILE
```matlab
knots2mph = 1.15078;
t.FE = t.FF_TOTAL ./ (t.TAS*knots2mph);
t.FE(isinf(t.FE)) = 0;
```
Plot Data Using Tall Arrays

Summarize Fuel Efficiency using a histogram

```matlab
binEdges = 0:80;
histogram(t.FE,binEdges);
```

Evaluating tall expression using the Parallel Pool 'local'
Data Science Maturity Levels

- Ad-hoc Individual Analysis
- Generally Useful Tools for Analysis
- Common Infrastructure; Tested and Documented

- Make it easy to navigate the data
- Re-use each time you analyze the dataset
Data Science Maturity Levels

- **Ad-hoc Individual Analysis**
- **Generally Useful Tools for Analysis**
- **Common Infrastructure; Tested and Documented**

- Collaborate: Work with others on a common code base
- Verify: Write well-tested software
- Share: Build tools for others
MATLAB for Data Science Teams

1. MATLAB Projects
2. Unit Testing
3. Toolbox Packaging
MATLAB Projects
Testing

```matlab
classdef testFlightDatastore < matlab.unittest.TestCase
    % For guidance on testing a custom datastore, see: https://www.mathworks.com
    properties
        sampleDataSet
ds
    end
    properties (TestProperty)
datastoreProperties = {'CurrentFileIndex','NumberOfFiles','VariableNames'};
datastoreMethods = {'copy','getLocation','hasData','initialize'};
infoFieldNames = {'Size','FileNames','Offsets'};
    end
    methods (TestMemberSetup)
        function buildTestingDatastore(testCase)
            s = what('+tests/sampleData');
            testCase.sampleDataSet = s.path;
            testCase.ds = flightDatastore(testCase.sampleDataSet);
        end
```
Testing

Testing Guidelines for Custom Datastores

All datastores that are derived from the custom datastore classes share some common behaviors. This test procedure provides guidelines to test the minimal set of behaviors and functionalities that all custom datastores should have. You will need additional tests to qualify any unique functionalities of your custom datastore.

If you have developed your custom datastore based on instructions in Develop Custom Datastore, then follow these test procedures to qualify your custom datastore. First perform the unit tests, followed by the workflow tests:

- Unit tests qualify the datastore constructor and methods.
- Workflow tests qualify the datastore usage.

**hasdata**

Unit test guidelines for the hasdata method

<table>
<thead>
<tr>
<th>Test Case Description</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call the hasdata method on the datastore object before making any calls to read</td>
<td>true</td>
</tr>
<tr>
<td>Call the hasdata method on the datastore object after making a few calls to read, but before all the data is read</td>
<td>true</td>
</tr>
<tr>
<td>When more data is available to read, call the readall method, and then call the hasdata method.</td>
<td>true</td>
</tr>
<tr>
<td>When no more data is available to read, call the hasdata method.</td>
<td>false</td>
</tr>
</tbody>
</table>
Creating a Toolbox

Flight Data Toolbox provides functions for working with the large repository of flight recorder data. It includes a datastore for accessing flight recorder data, and examples of analysis that can be performed.
Creating a Toolbox
Data Science Maturity Levels

- Ad-hoc Individual Analysis
- Generally Useful Tools for Analysis
- Common Infrastructure; Tested and Documented

- Scale-out to larger group of users
- Easier to maintain and share
Takeaways

- MATLAB has many new tools to help you better work with and utilize your data
- Create tools for you / your team / your organization to explore and analyze data
- Increasing maturity with data science is a journey; we’re here to help