



Università degli Studi di Pavia  
Dipartimento di Ingegneria Industriale e dell'Informazione

## Corso di Identificazione dei Modelli e Analisi dei Dati

Data Import

Prof. Giuseppe De Nicolao, Federica Acerbi, Alessandro Incremona

# Outline

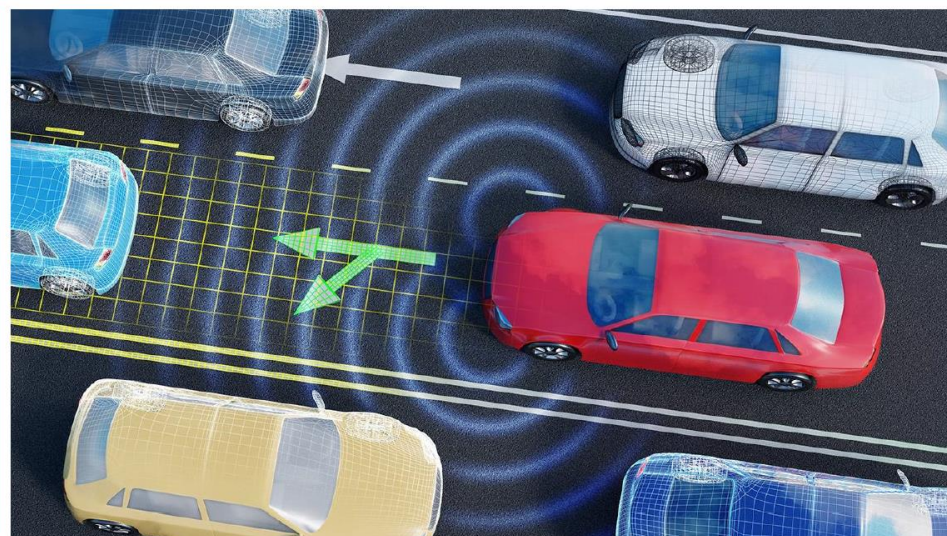
---

1. Introduction
2. Importing Tabular Data
3. Handling Large Dataset
4. Handling Missing Data

# Outline

---

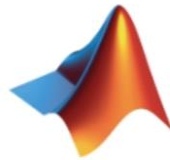
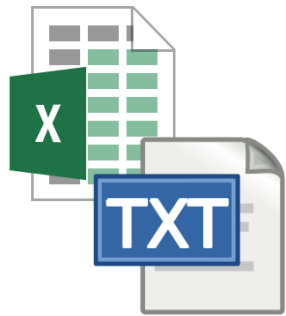
1. Introduction
2. Importing Tabular Data
3. Handling Large Dataset
4. Handling Missing Data



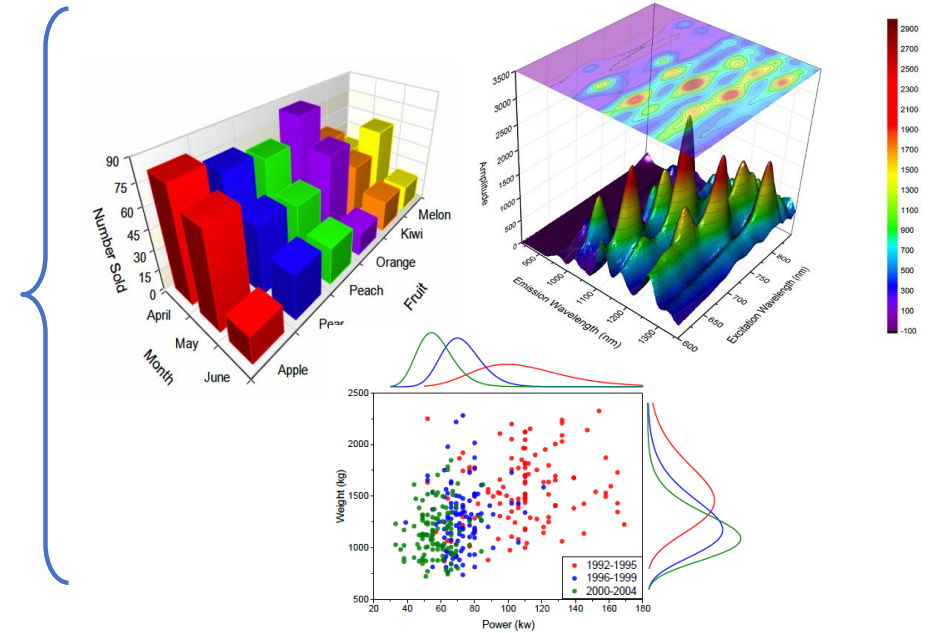
Matlab Expo 2018

# Lecture goals:

1. importing data from text files and spreadsheets into a single table or database
2. acquiring fundamentals of data visualization



Timestamp	Country	Windspeed	Pressure
'29-Jun-1991 12:00:00'	'N/A'	23	1012
'29-Jun-1991 18:00:00'	'N/A'	23	1012
'30-Jun-1991 00:00:00'	'N/A'	23	NaN
'30-Jun-1991 06:00:00'	'United States'	109	957
'30-Jun-1991 12:00:00'	'United States'	109	980
'1-Aug-1991 06:00:00'	'Canada'	70	1050



# Outline

---

1. Introduction
- 2. Importing Tabular Data**
3. Handling Large Dataset
4. Handling Missing Data

There are different types of file formats that you can import and export from the MATLAB® application:

File content	Extension	Description	Import function
Matlab formatted data	MAT	MATLAB variables	load
Text	CSV, TXT	Column-oriented delimited numbers or a mix of text and numbers	readtable
Spreadsheet	XLS, XLSM, XLSX	Column-oriented data in worksheet or range of spreadsheet	readtable
Extensible Markup Language	XML	XML-formatted text	xmlread
Image	BMP, GIF, JPEG	Windows bitmap, Graphics Interchange Format, Joint Photographic Experts Group	imread
Audio	MP4, WAV	MPEG-4, Microsoft WAVE sound	audioread
Video	MP4, AVI	MPEG-4, Audio Video Interleave	videoreader

There are different types of file formats that you can import and export from the MATLAB® application:


File content	Extension	Description	Import function
Matlab formatted data	MAT	MATLAB variables	load
Text	CSV, TXT	Column-oriented delimited numbers or a mix of text and numbers	readtable
Spreadsheet	XLS, XLSM, XLSX	Column-oriented data in worksheet or range of spreadsheet	readtable
Extensible Markup Language	XML	XML-formatted text	xmlread
Image	BMP, GIF, JPEG	Windows bitmap, Graphics Interchange Format, Joint Photographic Experts Group	imread
Audio	MP4, WAV	MPEG-4, Microsoft WAVE sound	audioread
Video	MP4, AVI	MPEG-4, Audio Video Interleave	videoreader




# Importing Tabular Data

---

## Text

Import Option	Description
Import Tool 	Import a file or range of data to column vectors, a matrix, a cell array, or a table. You can generate code to repeat the operation on multiple similar files.
<code>readtable</code>	Import column-oriented data into a table.
<code>csvread</code>	Import a file or range of comma-separated numeric data to a matrix.
<code>textscan</code>	Import a nonrectangular or arbitrarily formatted text file to a cell array.


## Spreadsheet

Import Option	Description
Import Tool 	Import a worksheet or range to column vectors, a matrix, a cell array, or a table. You can generate code to repeat the operation on multiple similar files.
<code>readtable</code>	Import a worksheet or range to a table.
<code>xlsread</code>	Import a worksheet or range to numeric and cell arrays.


# Importing Tabular Data

---

## Text

Import Option	Description
Import Tool 	Import a file or range of data to column vectors, a matrix, a cell array, or a table. You can generate code to repeat the operation on multiple similar files.
<code>readtable</code>	Import column-oriented data into a table.
<code>csvread</code>	Import a file or range of comma-separated numeric data to a matrix.
<code>textscan</code>	Import a nonrectangular or arbitrarily formatted text file to a cell array.


## Spreadsheet

Import Option	Description
Import Tool 	Import a worksheet or range to column vectors, a matrix, a cell array, or a table. You can generate code to repeat the operation on multiple similar files.
<code>readtable</code>	Import a worksheet or range to a table.
<code>xlsread</code>	Import a worksheet or range to numeric and cell arrays.


# Importing Tabular Data

---

## Text

Import Option	Description
Import Tool 	Import a file or range of data to column vectors, a matrix, a cell array, or a table. You can generate code to repeat the operation on multiple similar files.
<code>readtable</code>	Import column-oriented data into a table.
<code>csvread</code>	Import a file or range of comma-separated numeric data to a matrix.
<code>textscan</code>	Import a nonrectangular or arbitrarily formatted text file to a cell array.

## Spreadsheet

Import Option	Description
Import Tool 	Import a worksheet or range to column vectors, a matrix, a cell array, or a table. You can generate code to repeat the operation on multiple similar files.
<code>readtable</code>	Import a worksheet or range to a table.
<code>xlsread</code>	Import a worksheet or range to numeric and cell arrays.

**Unstructured data**

# Text Files

---

If you have **regularly formatted tabular** data in a single or a delimited text file, you can use the **edit** function to visualize it in a MATLAB built-in text editor and the **readtable** function to import it into MATLAB

>> doc **edit**

## edit

---

Edit or create file

## Syntax

---

```
edit
edit file
edit file1 ... fileN
```

>> doc **readtable**

## readtable

---

Create table from file

## Syntax

---

```
T = readtable(filename)
T = readtable(filename,Name,Value)
T = readtable(filename,opts)
T = readtable(filename,opts,Name,Value)
```

# Text Files

---

If you have **regularly formatted tabular** data in a single or a delimited text file, you can use the **edit** function to visualize it in a MATLAB built-in text editor and the **readtable** function to import it into MATLAB

```
>> edit 'hurricaneData1990s.txt'
```

```
Number, Timestamp, Country, Windspeed, Pressure  
1, 29-Jun-1991 12:00:00, "N/A", 23, 1012  
1, 29-Jun-1991 18:00:00, "N/A", 23, 1012  
1, 30-Jun-1991 00:00:00, "N/A", , 1012  
1, 30-Jun-1991 06:00:00, "United States", 23, 1012  
1, 30-Jun-1991 12:00:00, "United States", 23, 1012  
1, 30-Jun-1991 18:00:00, "N/A", 23, 1012  
1, 01-Jul-1991 00:00:00, "N/A", 23, 1012  
1, 01-Jul-1991 06:00:00, "United States", 23, 1012  
1, 01-Jul-1991 12:00:00, "United States", 23, 1012
```

# Text Files

---

If you have **regularly formatted tabular** data in a single or a delimited text file, you can use the **edit** function to visualize it in a MATLAB built-in text editor and the **readtable** function to import it into MATLAB

```
>> readtable('hurricaneData1990s.txt')
```

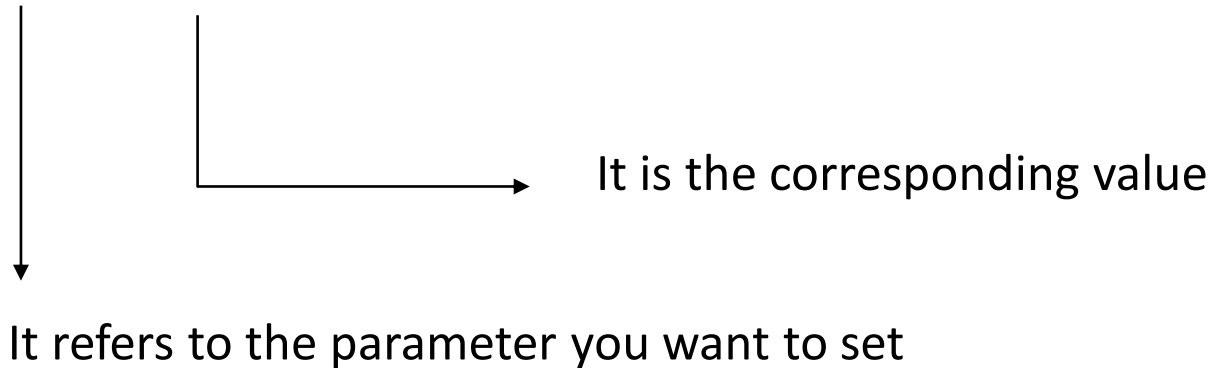
Number	Timestamp	Country	Windspeed	Pressure
1	29-Jun-1991 12:00:00	'N/A'	23	1012
1	29-Jun-1991 18:00:00	'N/A'	23	1012
1	30-Jun-1991 00:00:00	'N/A'	NaN	1012
1	30-Jun-1991 06:00:00	'United States'	23	1012
1	30-Jun-1991 12:00:00	'United States'	23	1012
1	30-Jun-1991 18:00:00	'N/A'	23	1012
1	01-Jul-1991 00:00:00	'N/A'	23	1012
1	01-Jul-1991 06:00:00	'United States'	23	1012
1	01-Jul-1991 12:00:00	'United States'	23	1012

# Text Files

---

When using the **readtable** function, you can specify some additional parameters in order to control how data are read by using the comma-separated pairs of **Name, Value** arguments.

>> **readtable(filename, Name, Value)**



# Text Files

---

When using the **readtable** function, you can specify some additional parameters in order to control how data are read by using the comma-separated pairs of **Name**, **Value** arguments.

>> edit 'hurricaneData1990s\_v2.txt'

```
This file contains the wind speed and pressure data  
of hurricanes in the early 1990s.
```

```
Units - Number, Date and Time, Text, Number (mbar), Number (mph)
```

```
Number, Timestamp, Country, Windspeed, Pressure  
1, 29-Jun-1991 12:00:00, "N/A", 23, 1012  
1, 29-Jun-1991 18:00:00, "N/A", 23, 1012  
1, 30-Jun-1991 00:00:00, "N/A", , 1012  
1, 30-Jun-1991 06:00:00, "United States", 23, 1012  
1, 30-Jun-1991 12:00:00, "United States", 23, 1012  
1, 30-Jun-1991 18:00:00, "N/A", 23, 1012
```



# Text Files

---

When using the **readtable** function, you can specify some additional parameters in order to control how data are read by using the comma-separated pairs of **Name**, **Value** arguments.

```
>> readtable('hurricaneData1990s_v2.txt', 'HeaderLines', 5)
```

<b>Number</b>	<b>Timestamp</b>	<b>Country</b>	<b>Windspeed</b>	<b>Pressure</b>
1	29-Jun-1991 12:00:00	'N/A'	23	1012
1	29-Jun-1991 18:00:00	'N/A'	23	1012
1	30-Jun-1991 00:00:00	'N/A'	NaN	1012
1	30-Jun-1991 06:00:00	'United States'	23	1012
1	30-Jun-1991 12:00:00	'United States'	23	1012
1	30-Jun-1991 18:00:00	'N/A'	23	1012
1	01-Jul-1991 00:00:00	'N/A'	23	1012
1	01-Jul-1991 06:00:00	'United States'	23	1012
1	01-Jul-1991 12:00:00	'United States'	23	1012

## Main Name,Value pairs:

---

- **'HeaderLines', Value:** skip the first 'Value' rows of the file. 'Value' must be an integer.  
e.g.: `readtable('dataset.txt','HeaderLines',5)`
- **'Delimiter',Value:** specify the delimiter between different columns. 'Value' must be any valid character (see documentation).  
e.g.: `readtable('dataset.txt','Delimiter',',')`
- **'CommentStyle', Value:** specify the symbol designating text to ignore. 'Value' must be any valid character (see documentation).  
e.g.: `readtable('dataset.txt','CommentStyle','%')`
- **'EmptyValue', Value:** set the returned value for empty numeric fields (default is NaN).  
e.g.: `readtable('dataset.txt','EmptyValue',999)`

# Exercise 1

---

The file `'hurricaneData1990s_v3.txt'` contains the data of six hurricanes in the early 1990s.

1. Import this dataset as a table where each row must contain an **ID\_number** (data type: double), the **date** (data type: datetime), the **country** (data type: char), the **windspeed** (data type: double) and the **pressure** (data type: double). Missing numeric values must be replaced with **-1**.
2. Check that the data type of each column of the table is corrected by using the function **class**
3. Rename the column of the table as: ID, Date, Location, Wind\_Speed, Pressure.

## TIPS:

- You can use dot indexing to access table columns. e.g.: `table.NameOfCol1`
- If you don't remember how to access the 'VariableNames' property of a table, check into the documentation (`>> doc table`)

# Spreadsheet files



---

There are **three main approaches** by which you can import data from spreadsheet files into MATLAB®: **interactively, programmatically or by pasting data from the clipboard.**

Import option	Description
Import Tool	Import a worksheet or range to column vectors, a matrix, a cell array, a string array, or a table. You can generate code to repeat the operation on multiple similar files.
readtable	Import a worksheet or range to a table.
Copy and Paste	You can paste spreadsheet data from the clipboard into MATLAB.

# Import Tool

---


- On the **Home** tab, in the **Variable** section, click **Import data**  .
- Select the spreadsheet file you want to import. The **Import Tool** opens.
- Select the data you want to import. You can edit the variables names and other options.
- Click the **Import Selection** button  to import the data in your workspace.

## Exercise 2

Import the first ten rows of data from the file '[dati\\_consumo\\_Italia.xlsx](#)' (from the sheet 'dati 2010-2012') using the Import Tool. The data must be imported as a table with two columns of numeric values named 'Date' and 'Load'.

# Copy and Paste

---

- Select and copy your spreadsheet data in Microsoft Excel, then use one of the following methods:
  1. On the **Workspace browser** title bar, click  and select **Paste**.
  2. Open an existing variable in the Variables editor, right-click and then select **Paste Excel** (Note: the data types must be the same!).
  3. Use the command `'uiimport-pastespecial'`.

# readtable

---

## `readtable(filename, Name, Value)`

- The syntax is the same as the text file case, but here we need to specify new **Name, Value** pair arguments to correctly select the data that we want to import.

Main **Name,Value** pairs for spreadsheet files:

- **'Sheet', Value:** worksheet to read. 'Value' must be a string containing the worksheet name.  
e.g.: `readtable('dataset.xlsx','Sheet','Foglio1')`
- **'Range', Value:** portion of the worksheet to read, indicated as a rectangular area. 'Value' must be specify using the syntax 'Corner1:Corner2', where Corner1 and Corner2 are two opposing corners that define a rectangular region.  
e.g.: `readtable('dataset.xlsx','Range','B3:H45')`

## Exercise 3

---

1. Import all the data of the file `'dati_consumo_Italia.xlsx'` from the sheet `'dati 2010-2012'` in a table called `"tab1"` using the `readtable` command.
2. Then import all the data of the same file but from the sheet `"dati 2013-2014"` in a table called `"tab2"`.
3. Rename the columns of the two table as `'Date'` and `'Load'`.
4. Create a table called `"tab_dati"` which is the result of the vertical concatenation of the two previous tables.



# Outline

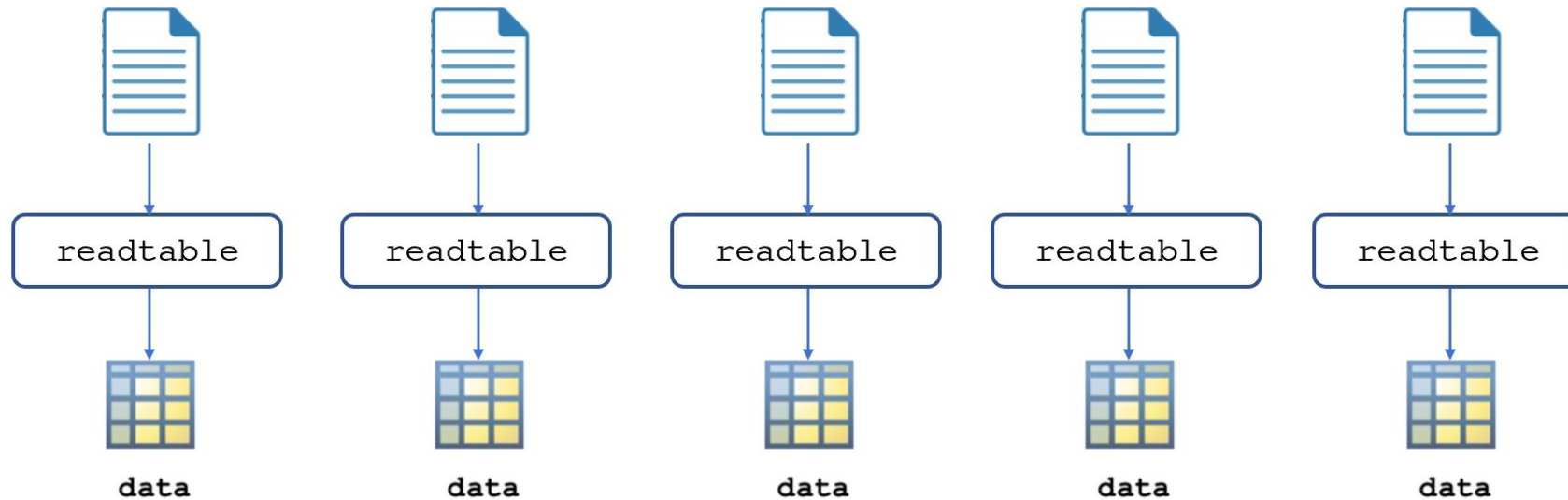
---

1. Introduction
2. Importing Tabular Data
- 3. Handling Large Dataset**
4. Handling Missing Data

# Handling Large Dataset

---

Until now...



**N data sheets = N Tables**

# Handling Large Dataset

---

To easily manage multiple data files:



# Handling Large Dataset

---

To easily manage multiple data files:



## What is a datastore?

- an **object useful** for reading a single file or a collection of files or data **too large to fit in memory**
- a **repository** for multiple files and folders with the same structure and formatting

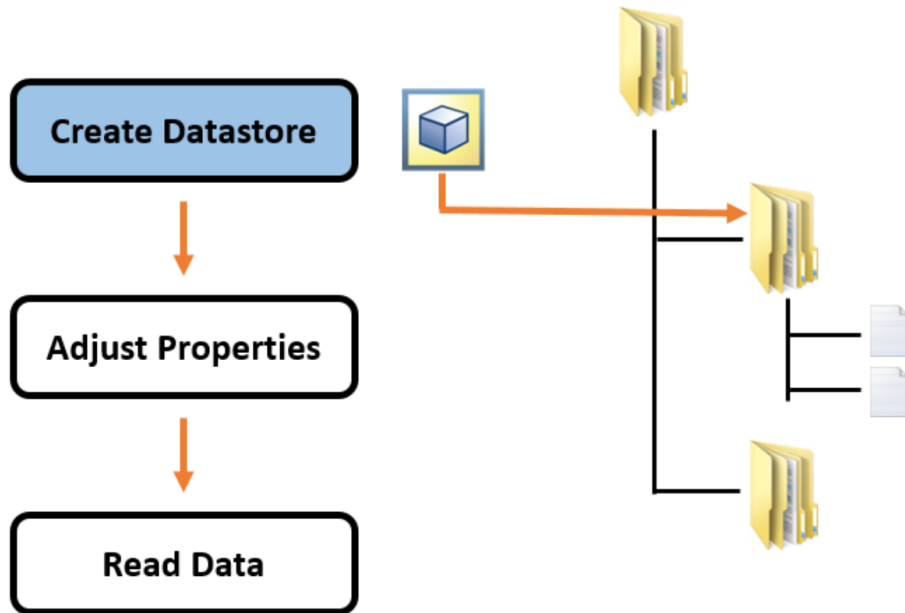
# Handling Large Dataset

---

Steps to read data using a datastore:

1. **Creating a datastore** that refers to a single file or a set of files in a directory
2. **Adjusting the properties** of the datastore according to the data contents
3. **Reading the data** from the file
  - Read Selected Columns of Data
  - Read Subsets of Data
  - Read One File at time

# Creating the datastore



- To create a datastore use the datastore function with the file or folder location as input

```
ds = datastore(location)
```

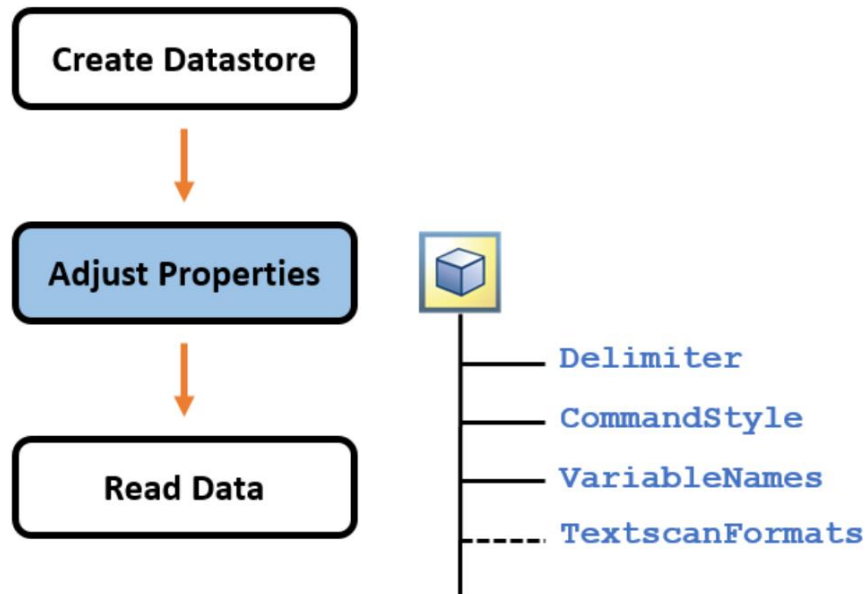
- To visualize how the data will be read, the preview function returns a subset of data from datastore  

```
data = preview(ds)
```



**The datastore variables does not contain any data but meta informations about the data!**

# Modifying Datastore Properties



Adjusting the datastore properties is possible to customize the data import procedure

Examples:

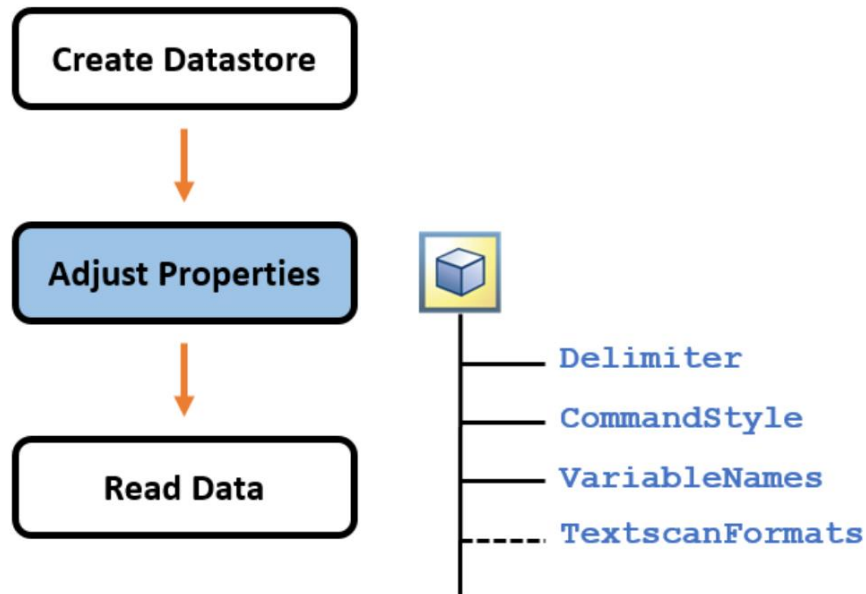
- To ignore lines of data that begins with the character sequence '//':

```
ds.CommentStyle = '//'
```

- To disable the automatic variable name detection:

```
ds.ReadVariableNames = false
```

# Modifying Datastore Properties



Adjusting the datastore properties is possible to customize the data import procedure

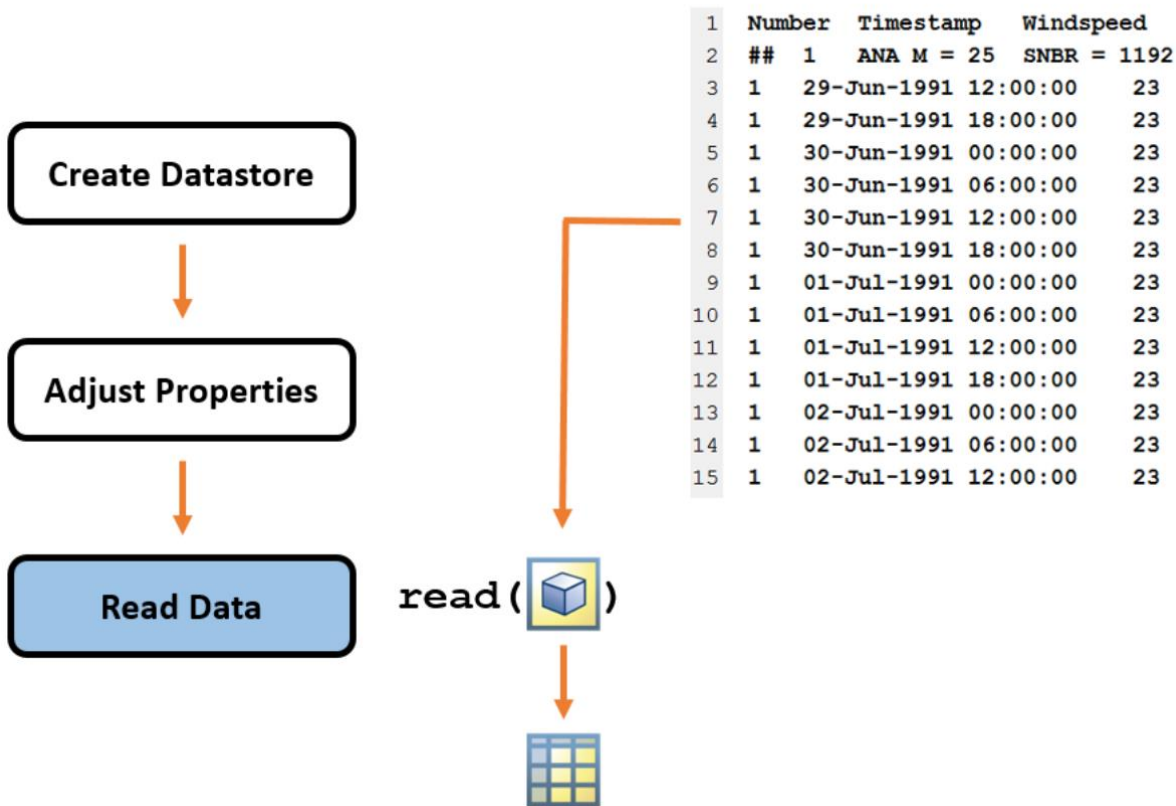
Examples:

- To skip a number of lines at the beginning of the file:  
`ds.numHeaderLines = 3`
- To set the amount of data to read in a call to the **read** function

```
ds.ReadSize = 'file'  
ds.ReadSize = '15000'
```



# Reading the data



- The readall function allows to import the data from **all files** referenced by the datastore

data = readall(ds)

- The read function allows to import the data from the **first file/fixed amount of lines** referenced by the datastore

data = read(ds)

# Reading the data

---

Three different ways to use the datastore to work with a dataset that does not entirely fit in the memory of your machine. Example: `ds = datastore('airlinesmall.csv', 'TreatAsMissing', 'NA')`

## 1. Read Selected Columns of Data

```
>> ds.SelectedVariableNames = 'ArrDelay';
```

```
>> data = readall(ds);
```

# Reading the data

---

Three different ways to use the datastore to work with a dataset that does not entirely fit in the memory of your machine

## 2. Read Subsets of Data

```
>> ds.ReadSize = 15000;  
>> sumElapsedTime = 0;  
>> while ds.hasdata  
>>   T = read(ds);  
>>   sumElapsedTime = sumElapsedTime + sum(T.ActualElapsedTime);  
>> end  
  
>> sumElapsedTime
```

# Reading the data

---

Three different ways to use the datastore to work with a dataset that does not entirely fit in the memory of your machine

## 3. Read One File at Time

```
>> ds.ReadSize = 'file';  
>> sumElapsedTime = 0;  
>> while ds.hasdata  
>>   T = read(ds);  
>>   sumElapsedTime = sumElapsedTime + sum(T.ActualElapsedTime);  
>> end  
  
>> sumElapsedTime
```

## Exercise 4 (part 1)

---

1. Create a datastore named `dat` from the `.txt` files contained in the folder `hurricaneData`
2. Preview the files contents using the `dat` datastore to check and compare the results with the data in the editor
3. Ignore lines that begins with the character sequence `'##'`
4. Change the variable names in the datastore with:  
{'Time','Latitude','Longitude','WindSpeed','Pressure'}
5. Save the path of the files included in datastore to a variable names `fname`
6. Read the data from the first file referenced by the datastore `dat` and store the imported data in `hurrs1`.

## Exercise 4 (part 1)

---

7. Read the data from the second file referenced by the datastore `dat` and store the imported data in `hurrs2`
8. Select the variable 'WindSpeed' and save the all the data in the table `wind_speed`

# Outline

---

1. Introduction
2. Importing Tabular Data
3. Handling Large Dataset
4. Handling Missing Data

# Working with Missing Data

---

- When you import data into MATLAB, missing numerical values are replaced with NaN, which stands for “not a number”
- If a numerical arrays contains NaN values, many function that operate on such an array will return NaN

Timestamp	Country	Windspeed	Pressure
'29-Jun-1991 12:00:00'	'N/A'	23	1012
'29-Jun-1991 18:00:00'	'N/A'	23	1012
'30-Jun-1991 00:00:00'	'N/A'	23	NaN
'30-Jun-1991 06:00:00'	'United States'	109	957
'30-Jun-1991 12:00:00'	'United States'	109	980
'1-Aug-1991 06:00:00'	'Canada'	70	1050

mean (data . Pressure)

NaN



# How to ignore NaNs in the data?

- using the **omitnan** flag when you are working with matlab function

```
>> m = mean(tab.Windspeed,'omitnan')
```

- deleting the observations with missing data using `>> rmmissing(tab)`

Timestamp	Country	Windspeed	Pressure
'29-Jun-1991 12:00:00'	'N/A'	23	1012
'29-Jun-1991 18:00:00'	'N/A'	23	1012
'30-Jun-1991 00:00:00'	'N/A'	23	NaN
'30-Jun-1991 06:00:00'	'United States'	109	957
'30-Jun-1991 12:00:00'	'United States'	109	980
'1-Aug-1991 06:00:00'	'Canada'	70	1050



Timestamp	Country	Windspeed	Pressure
'29-Jun-1991 12:00:00'	'N/A'	23	1012
'29-Jun-1991 18:00:00'	'N/A'	23	1012
'30-Jun-1991 06:00:00'	'United States'	109	957
'30-Jun-1991 12:00:00'	'United States'	109	980
'1-Aug-1991 06:00:00'	'Canada'	70	1050

# Exercises

---

1. Use the mean function to create a variable named xAvg that contains the average value of  $x = [0.32 \ 0.95 \ \text{NaN} \ 0.87 \ 0.71 \ 0.42]$ ;

## Solution:

```
>> xAvg = mean(x,'omitnan')
```

2. Create a variable named xMin that contains the smallest value in x

## Solution:

```
>> xMin = min(x)
```



**In some Matlab functions  
'omitnan' is the default option**

# Exercises

---

3. Remove NaNs from the vector x

```
x = [0.32 0.95 NaN 0.87 0.71 0.42]
```

**Solution:**

```
>> !x = isnan(x)
```

```
>> !x = [0 0 1 0 0 0]
```

# Exercises

---

3. Remove NaNs from the vector x

x = [0.32 0.95 NaN 0.87 0.71 0.42]

## Solution:

```
>> lx = isnan(x)
```

```
>> lx = [0 0 1 0 0 0]
```

➤ **isnan** checks each element of the input array and returns a logical array in which the true values (displayed as 1) indicate locations of NaNs

# Exercises

---

3. Remove NaNs from the vector x

x = [0.32 0.95 NaN 0.87 0.71 0.42]

## Solution:

```
>> lx = isnan(x)
```

```
lx = [0 0 1 0 0 0]
```

```
>> z = x(not(lx))
```

```
z = [0.32 0.95 0.87 0.71 0.42];
```

➤ **isnan** checks each element of the input array and returns a logical array in which the true values (displayed as 1) indicate locations of NaNs

# Exercises

---

4. Import the data from the file `hurricaneData1990s_v2.txt` using the function `readtable` and delete the rows of the table that contain missing values

## Solution:

```
>> T = readtable('hurricaneData1990s_v2.txt','HeaderLines',5)
```

```
>> im = ismissing(T)
```

```
>> ridx = any(im,2)
```

```
>> T(ridx,:) = []
```

or

```
>> T = rmmissing(T)
```

## Exercise 4 (part 2)

---

1. Check if the table hurrs1 and hurrs2 contains NaN values
2. Remove the NaN values if present in table hurrs1 and hurrs2

## Reference Documentation:

---

- <https://it.mathworks.com/>
- <http://sisdin.unipv.it/labsisdin/teaching/courses/imadlt/esercitazioni>

