

# MATLAB: Session 1

---

Diego Jimenez   Ciaran Rogers

# Why MATLAB?

1. Easy to use scripting language for linear algebra
2. Built-in Numerical Computation
  - Matrix decompositions
  - Numerical integration
  - Optimization etc.
3. Plotting in tractable and appealing way

# MATLAB Interface

The screenshot displays the MATLAB R2016b interface with the following components:

- Top Bar:** Includes tabs for HOME, PLOTS, APPS, EDITOR, PUBLISH, and VIEW. A search bar for documentation is on the right.
- Toolbox:** A row of icons for file operations (New, Open, Save, Print), navigation (Go To, Find), and execution (Run, Run and Advance, Advance, Run Section, Run and Time).
- Current Folder:** A tree view on the left showing the file structure of the current project, including folders like 'Assets.xlsx', 'Attempt.m', and 'bond\_price.m'.
- Editor - Main.m:** The central workspace for writing code. The current file is 'Main.m', which contains MATLAB code for parameter assignment and data processing. The code includes comments and variable assignments such as `phi_beq = 1.0;`, `beta = exp(-0.025*years);`, and `theta_bar_E = data(:,4)./sum(data(:,4));`.
- Command Window:** A window at the bottom left showing the execution of commands. It displays the same code as the Editor, with the prompt `fj >>` at the end.
- Workspace:** A window on the right showing the variables currently in memory. It lists variables like `ageprofile`, `beta`, `bond_opt`, `bonds_bar`, `bonds_new`, `c_opt`, `captax`, `captaxh`, `data`, `data_Germany_1`, `data_Germany_2`, `data_Italy_1`, `data_Italy_2`, `data_new`, `death`, `equity_opt`, `expmean`, `expmean_net`, `expmean_net_obj`, `factors`, `finfl`, `fstock`, `gamma`, `hidfac`, `housing_opt`, `idcorrel`, and `idmean`.

# Getting Help

If you don't know, Google it! (someone very likely to have had same problem)

Help!

```
>> help < command >
```

```
>> doc < command >
```

```
>> lookfor (somewhat inferior to google)
```

Demos and examples

# Basic Syntax (1)

- **Semicolon**

```
>> 4 + 5;
```

- **% for comments**

```
>> 4 * 5% - 6
```

- **Assignment**

```
>> y = 4/5
```

- **Matrix Assignment**

```
>> y = [1, 2; 3, 4]
```

- **Single Element Assignment**

```
>> y(2, 2) = 4/10
```

## Basic Syntax (2)

- Transpose

```
>> y = y'
```

- Logical

```
>> y = (y == 0)
```

- Colon Operator

```
>> x = -pi : 0.1 : pi;
```

- Linspace Command

```
>> z = linspace(-pi, pi, 100)
```

- Call Functions

```
>> y = sin(x)
```

- Using Logical Statements

```
>> y(y > 0.5) = 0.5
```

## Basic Syntax (3)

- **Matrix Multiplication**

```
>> x * x
```

```
>> x.*x
```

- **Strings**

```
>> a = ['two' 'words']  
a(2)
```

- **Size**

```
>> size(x)
```

```
>> size(x, 2)
```

- **Zeros**

```
>> zeros(5, 3)
```

```
>> zeros(size(x))
```

- **Clear Command/Variables/Figures**

```
clc; clear all; close all
```

## Stopping Code Running

ctrl + c

## Running the Script

- **"Run" Tab** - Found in Editor tab - runs full script
- **"Run Section" Tab** - Found in Editor tab - runs section
- **Cmd+Enter** - Runs whatever is highlighted



# Basic Plotting

Create a figure

```
>> figure;
```

Line plot

```
>> plot(x, y);
```

title;xlabel;ylabel;

```
>> title('myplot'); xlabel('x'); ylabel('y');
```

Scatter plot

```
>> holdon; scatter(x, y);
```

## Saving Data

```
save('filename',' var1',' var2',' var3')
```

## Loading Data

- MATLAB files

```
load('filename')
```

- Excel Files

```
[data textdata] = xlsread('Random.xlsx')
```

- Add paths to other files

```
addpath(< pathname >)
```

## Example: TripAdvisor Dataset

Load the data

```
[data textdata] = xlsread('TripAdvisor.xlsx','Raw')
```

Use *unique* to identify set of possible prices

```
Prices = unique(textdata(2:end,5))
```

Then, use *strcmp* to generate logicals from strings

```
PriceLow = strcmp(textdata(2:end,5),Prices(1))
```

```
PriceMed = strcmp(textdata(2:end,5),Prices(1))
```

```
PriceHig = strcmp(textdata(2:end,5),Prices(1))
```

Clean data by removing:

- Observations where there is no Price indication
- Observations where merge was unsuccessful

# Regressions (1)

Assignment of Variable Names

```
placeId = datamatrix(:,1);
```

Generate X variable

```
X = [ones(size(userRating)) Height];
```

Run regression

$$b = X \text{ over } Y$$

Generate t-statistics

## Regressions (2)

Run Regression with More Dummies

$X = [\text{ones}(\text{size}(\text{userRating})) \text{ isGreat isFresh isOk PriceHig PriceMed}]$ ;

EXERCISE: What is the effect of price on ratings?

- Start with basic single-variable
- Add variables you think are relevant
- Evaluate t-statistics to examine significance

Generation of functions

function < outputvars > = < function-name > (< inputvars >)

Functions saved as m. files under the function-name.

Example  $y = \min(x, 2)$

## **"for" Loops**

```
for  $i = 1 : 10$   
    statements  
end
```

## **"If" Statements**

```
if  $x == 0$   
    statements  
else    statements  
end
```

## **"while" Statements**

```
while  $d > 0.01$   
    statements  
end
```

**Note:** If you can, write in matrix form, not loops

## Example 1: Fibonacci

$$f(1) = 1; f(2) = 1$$

$$f(n) = f(n-1) + f(n-2)$$

**Write code that returns the nth element**

1. Recursion
2. Loop
3. Practice
  - Write code that returns first n elements (in loop and recursion)
  - Use tic/toc to time recursion and loop functions; plot the differences up to n=30
  - Question: Why is one faster than the other?



## Example 2: Numerical Integration/Pareto Distribution

$$F_X(x) = \begin{cases} 1 - \frac{x_m^\alpha}{x^\alpha} & \text{if } x \geq x_m \\ 0 & \text{if } x < x_m \end{cases}$$

$$f_X(x) = \begin{cases} \frac{\alpha x_m^\alpha}{x^{\alpha+1}} & \text{if } x \geq x_m \\ 0 & \text{if } x < x_m \end{cases}$$

### PRACTICE 1

1. Write a function for the Pareto PDF ( $\alpha = 2$ ;  $x_m = 1$ )
2. Write a function for the Pareto CDF

### PRACTICE 2: Derive Probability of Drawing

1. Define a grid  $x$  that covers (3,4), with step size  $d$
2. Evaluate Pareto density on  $x$ , store as  $f_x$
3. Find the sum of  $f_x$  times the step size

# Function Handles and Expected Values

- Convenient alternative to writing an alternative function .m file
  - particularly useful if function is simple
  - `funcname = @(x)(f(x))`

## Examples

- Example 1: `addone = @(x)(x + 1)`
- Example 2: `addition = @(x,y)(x + y)`
- Example 3: `multiply = @(x,y)(x * y)`

**EXERCISE:** Expected Values Using Function Handles