# Week 2:
# Matrices and programming in MATLAB

MCDB-BCHM 4312-5312

**2019-09-06**

Given

```
R = [1, 1, 2, 3, 5, 8, 13]
```

Which of the following commands will return

```
result = [3, 5, 8]
```

a) `result = R([3, 5, 8])`

b) `result = R(3, 5, 8)`

c) `result = R([4, 5, 6])`

d) `result = R(4, 5, 6)`

Given the following

```
A = [1; 2; 3]
B = [4; 5; 6]
```

Which of the following commands will return

```
C = [1; 2; 3; 4; 5; 6]?
```

```
a) C = [A B]

b) C = (A; B)

c) C = A + B

d) C = [A; B]
```

```
R = [1, 2, 3, 4]
```

Which TWO commands below will give an error?

```
a) R(1:3) = 1

b) R(1:3) = [5 6]

c) R(1:3) = [5 6 7 8]

d) R(1:3) = [5 6 7]

e) R(end + 1) = 8
```

Given the following

```
A = [1; 2; 3]
B = [4; 5; 6]
```

Which of the following commands will return

```
C = [1 4; 2 5; 3 6]?
```

```
a) C = [A B]

b) C = (A; B)

c) C = A + B

d) C = [A; B]
```

# Learning goals

- Matrices (continued from last week)
  - Size and number of elements
  - Indexing a sub-matrix
  - Performing matrix arithmetic operations
  - Performing element-wise operations

- Programming in MATLAB
  - Writing and running (“executing”) a script
  - Commenting your code

# Matrix size

Matrix size is defined as

Number of rows by Number of columns

To get the size of a matrix use function `size()`

Example:

$$M = \begin{bmatrix} 10 & 3 & 5 \\ 8 & 6 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

```
>> size(M)
```

= `[3, 2]` (“3-by-2 matrix”)

# Number of elements

- To count the number of elements in a matrix, use `numel()`

- Example:

```
>> numel(M)
```

numel = number of rows x number of columns

# Order of matrix dimensions in MATLAB

Columns
Dimension = 2

$$M = \begin{bmatrix} 1 & 4 \\ 3 & 6 \end{bmatrix}$$

Row
Dimension = 1

The order of dimensions is defined in MATLAB (a.k.a. you just need to know this)

## Subscripts

$$M = \begin{bmatrix} 1 & 4 \\ 3 & 6 \end{bmatrix}$$

$\mathtt{M}_{i,j}$ = $\mathtt{M}_{\mathtt{ij}}$ = `M(i,j)` = $i$-th row, $j$-th column

| Example: | Corresponding MATLAB syntax: |
|---|---|
| $M_{11}$ | >> M(1, 1) |
| $M_{21}$ | >> M(2, 1) |

# Modifying elements in a matrix

$$M = \begin{bmatrix} 10 & 3 & \boxed{5} \\ 8 & 6 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

How would you replace the element 5 above with the value 9?

a) `M = [10, 3, 8; 8, 6, 3; 1, 2, 1]`

b) `M(1, 3) = 9`

c) `M(3, 1) = 9`

# Indexing a range of elements

How would you retrieve just the first row of the matrix below?

$$M = \begin{bmatrix} 10 & 3 & 5 \\ 8 & 6 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

a) `M(1, 1:3)`

b) `M(1:3, 1)`

c) `M(1,1), M(1, 2), M(1, 3)`

# The colon operator = all elements in dimension

- The colon operator on its own specifies **<u>ALL ELEMENTS</u>** in that dimension

An alternative way of getting just the first row

$$M = \begin{bmatrix} 10 & 3 & 5 \\ 8 & 6 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

```
M(1, :)
```

# More examples of indexing

$$M = \begin{bmatrix} 10 & 3 & 5 \\ 8 & 6 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

- What is the command to retrieve the first column instead?

```
M(:, 1)
```

- What is the command to retrieve the first TWO rows?

```
M(1:2, :)
```

- What is the command to retrieve the LAST two rows?

`M(2:3, :)` or `M(end - 1:end, :)`

*The second command works no matter matrix size

# Modifying multiple elements in a matrix

$$M = \begin{bmatrix} 10 & 3 & 5 \\ 8 & 6 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

What is the command to replace row 1 with the following vector?

$$r = [5, 4, 3]$$

```
>> M(1, :) = r
```

- Note: The sizes must match during reassignment

`r = [5, 4, 3, 2]` will cause an error

# Indexing values using linear indices

$$M = \begin{bmatrix} 10 & 3 & 5 \\ 8 & 6 & 3 \\ 1 & 2 & 1 \end{bmatrix} \longrightarrow$$

Elements arranged linearly

$$\begin{matrix} 10 \\ 8 \\ 1 \\ 3 \\ 6 \\ 2 \\ 5 \\ 3 \\ 1 \end{matrix}$$

## Examples of linear indexing

$$M = \begin{bmatrix} 10 & 3 & 5 \\ 8 & 6 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

What is the linear index of M(2, 2)?

a) 5

b) 4

c) 6

d) 1

## Examples of linear indexing

$$M = \begin{bmatrix} 10 & 3 & 5 \\ 8 & 6 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

What is the value of `M(8)`?

a) 2

b) 3

c) 10

d) 5

## Examples of linear indexing

$$M = \begin{bmatrix} 10 & 3 & 5 \\ 8 & 6 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

What is the output of the command `M(end)`?

a) 1

b) 5

c) 3

d) Nothing. An error occurs

# Examples of linear indexing

$$M = \begin{bmatrix} 10 & 3 & 5 \\ 8 & 6 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

What is the value of `M(1:4)`?

```
[10, 8, 1, 3]
```

Indexing multiple elements of a matrix linearly combines the output into a row vector

# Mean of matrix values

$$M = \begin{bmatrix} 10 & 3 & 5 \\ 8 & 6 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

Compute the mean of all numbers in M using the function `mean()`

```
>> mean(M)
ans =
    6.3333    3.6667    3.0000
```

Computes the mean of **<u>each row</u>**

```
>> mean(mean(M))
ans =
    4.3333
```

# Mean of matrix values

$$M = \begin{bmatrix} 10 & 3 & 5 \\ 8 & 6 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

Compute the mean of all numbers in M using the function `mean()`

```
>> mean(M(:))
ans =
    4.3333
```

Computes the mean of **<u>every element</u>** in M

# Summary

- Declaring matrices
- The order of dimensions
- Getting the size of a matrix
- Retrieving elements using matrix notation and linear indexing

# Arithmetic operations on matrices

Two types of operations:

1. Matrix and scalar (number) – scalar operation
2. Matrix and matrix – matrix operation

We will look at examples for both for each operator

# Addition and subtraction

Matrix and scalar

$$C = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + 2 = \begin{bmatrix} 3 & 4 \\ 5 & 6 \end{bmatrix}$$

```
>> C = [1, 2; 3, 4] + 2
```

# Addition and subtraction

Matrix and matrix

$$C = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 3 & 3 \\ 4 & 6 \end{bmatrix}$$

```
>> A = [1, 2; 3, 4]
>> B = [2, 1; 1, 2]
>> C = A + B
```

# Quick caveat

Matrix and matrix

```
>> A = [1, 2; 3, 4]
>> B = [2, 1; 1, 2; 3, 4]
>> C = A + B
  Matrix dimensions must agree.
```

Matrices must have the **<u>same size</u>**

$$C = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 2 & 1 \\ 1 & 2 \\ 3 & 4 \end{bmatrix} = \text{operation undefined}$$

# Multiplication

- Matrix and scalar

$$C = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times 2 = \begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix}$$

```
>> A = [1, 2; 3, 4]
>> B = 2
>> C = A * B
```

# Multiplication

Matrix and matrix

$$C = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$$

# Multiplication

Matrix and matrix

$$C = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} = \begin{bmatrix} 6 & 6 \\ 14 & 14 \end{bmatrix}$$

```
>> A = [1, 2; 3, 4]
>> B = [2, 2; 2, 2]
>> C = A * B
```

Matrix multiplication (Linear algebra)

# Element-wise multiplication

- To multiply the elements from each matrix, use the element-wise multiplication operator (`.*`)

```
>> C = A .* B
```

$$C = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} .* \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix}$$

Also known as array multiplication

# Division

Matrix and scalar

```
>> A = [1, 2; 3, 4]
>> B = 2
>> C = A / B
```

$$C = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \div 2 = \begin{bmatrix} 0.5 & 1 \\ 1.5 & 2 \end{bmatrix}$$

# Division

Matrix and matrix

```
>> A = [1, 2; 3, 4]
>> B = [2, 2; 2, 2]
>> C = A / B
```

Linear algebra matrix division:

$AB = C$

$A = CB^{-1}$

$B^{-1}$ = inverse matrix

Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND = 4.432558e-17.

```
C =
  1.0e+15 *
  -2.8200    2.8200
  -2.8200    2.8200
```

# Element-wise division

$$C = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} ./ \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} = \begin{bmatrix} 0.5 & 1 \\ 1.5 & 2 \end{bmatrix}$$

```
>> C = A ./ B
```

# Power

```
>> A = [1, 2, 3, 4]
>> A ^ 2
```

What is the output of the command above?

`A ^ 2 = A * A` (A matrix operation)

What is the command if we want the square of each element in A instead?

```
A .^ 2
```

# Programming in MATLAB

- Code in MATLAB are written in plain text and saved with the extension **.m** (they are also often called m-files)

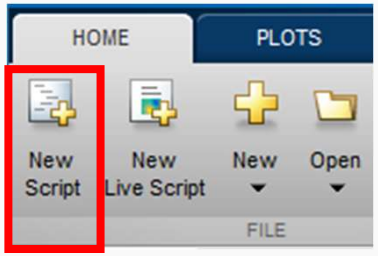
- Two types of m-files:
  - Scripts
  - Functions – we will look at this in another lecture

- Scripts contain lines of commands (basically, copy and paste from the command window)

# Creating a script

Two ways to create a script:

1. Click on **New Script**




2. Using the function `edit <scriptname>`

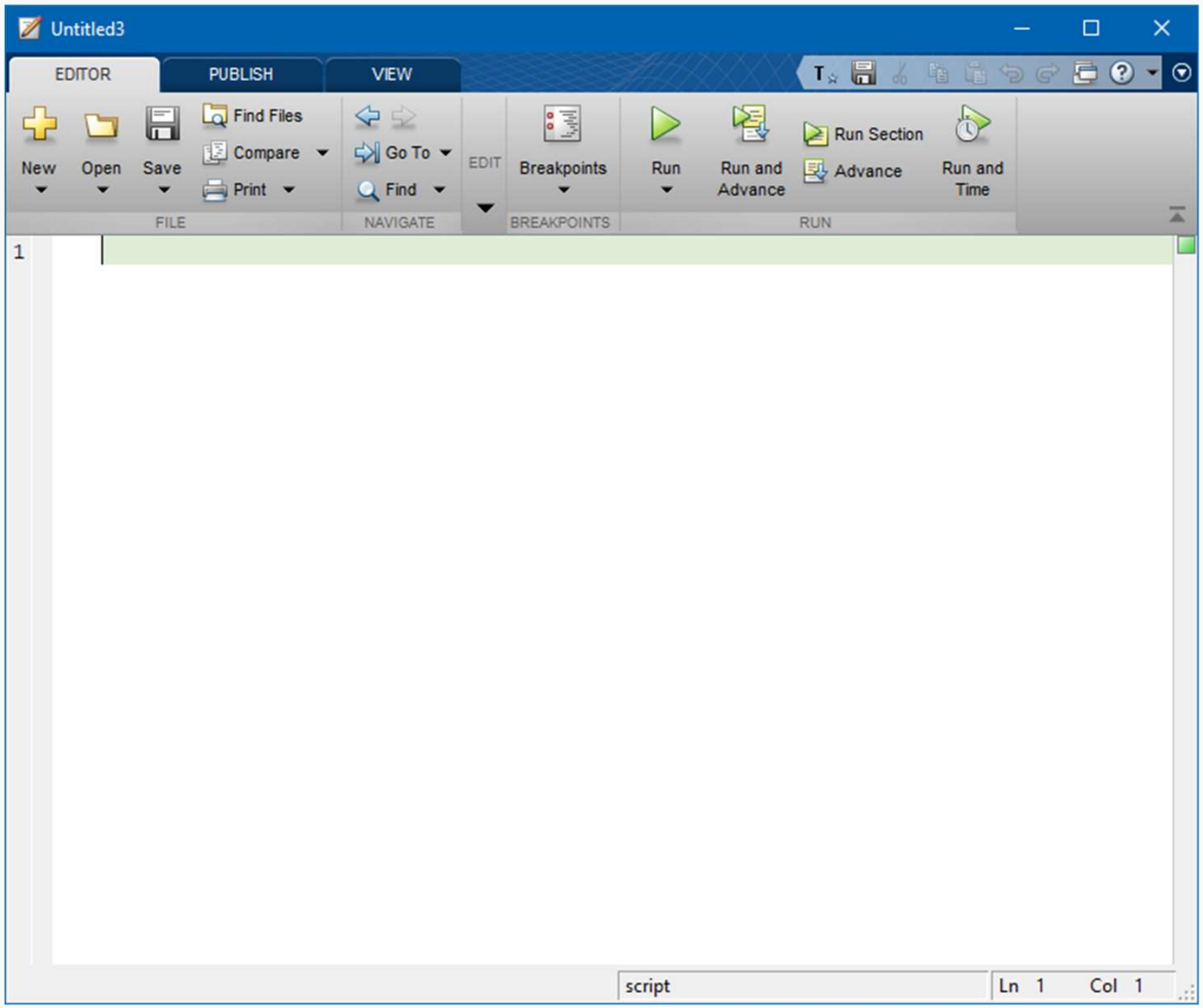
Example:

```
>> edit myNewScript.m
```

The script will be created in the current directory

- Restrictions on script names are similar to variable names
  - Filenames must start with a letter
  - Cannot have operators (letters, numbers and underscores only)

# The Editor window



# Two ways to run (“call”) a script

1. Click the **Run** button in the Editor window


2. Type the script name (without the `.m`) in the Command Window

```
>> calcarea
```

- Your file must be in the MATLAB search path – the easiest way is to have it in the current working directory

# Commenting your code

- Comments are lines of code that are not executed by MATLAB
- Comments in MATLAB start with a percent sign (%)
- Examples:

```
height = 10;  % Everything after is ignored
width = 20;   % meters

%Compute the area
area = height * width;
```

Whitespace is free – please use it!

# Suppressing the output from assignments

- Notice that whenever you assign a variable, MATLAB will print its value

```
>> width = 10
width =
     3
```

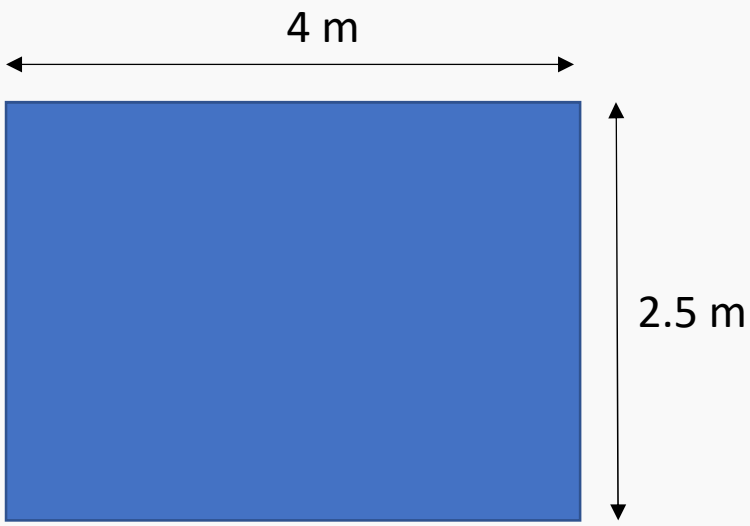
- You can suppress this output by adding a semicolon (`;`) at the end of the command

```
>> width = 10;
```

- This becomes useful when running multiple lines of code

# Exercise

Write a script called `calcarea.m` to compute the area of the rectangle below.




- Defining values (e.g. `height` and `width`) as variables will make it easier to remember what they are
- It is usually a good idea to start your script by clearing existing variables
- Also generally want to suppress output after each line

# Summary

- Matrices (continued from last week)
  - Size and number of elements
  - Indexing a sub-matrix
  - Performing matrix arithmetic operations
  - Performing element-wise operations

- Programming in MATLAB
  - Writing and running (“executing”) a script
  - Commenting your code