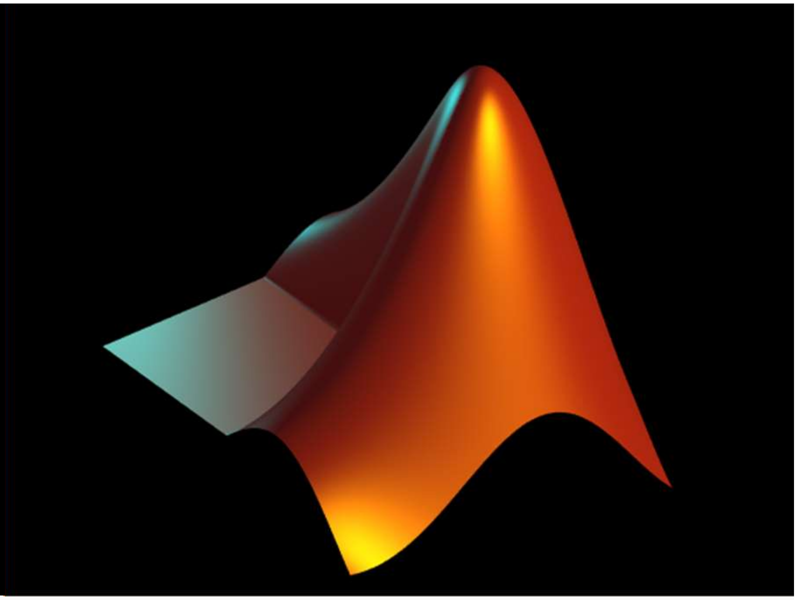


Week 1:
MATLAB and Matrices

MCDB-BCHM 4312-5312

2019-08-30

Expectations for MATLAB Fridays

- Have MATLAB running on your laptop
- Please interrupt and ask questions!
- Practice, practice, practice!

Office hours and MATLAB help

- Office hours:

NEXT TUESDAY (Sept 3)

2 – 3 pm

Room B331

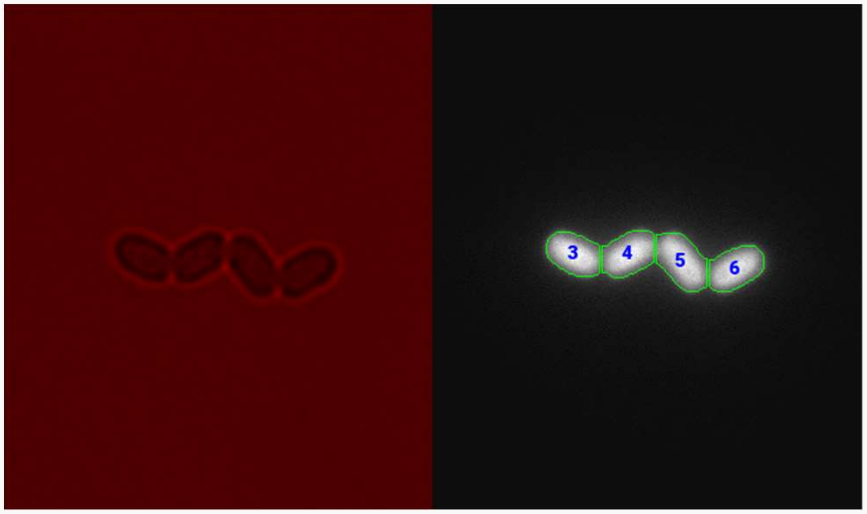
- Email help:

imageanalysis@colorado.edu

- Policy for homework related questions:

You must **make an attempt** before asking for help

Why do we have to learn how to program??



- Image analysis is the process of extracting meaningful data from images

- Examples:
  - Cell length vs time
  - Cell intensity vs time
  - Cell lineages (e.g. how do traits from the mother cell pass down to the daughters)

- Analyzing images by hand is tedious!

"Automate the boring stuff!"

Learning goals

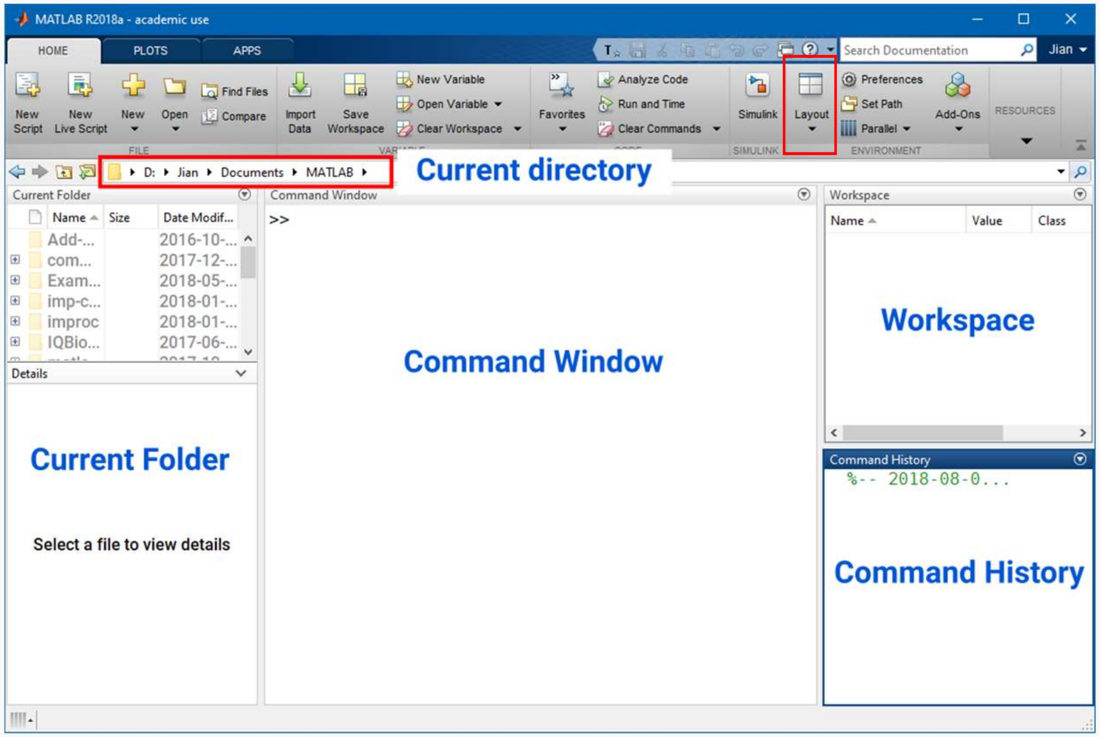
- Getting to know MATLAB
  - Identifying the important components of the interface
  - Performing basic arithmetic
  - Understanding how to declare variables and how to retrieve their values
  - Getting help in MATLAB

- Matrices
  - Declaring matrices
  - Retrieving individual and a range of values by indexing
  - Modifying and deleting matrix elements

The MATLAB interface

Important elements to know:

- **Command Window**
- Current Folder
- Working directory
- **Workspace**
- Command History



The arrangement of your window could be different – each panel can be moved/hidden

If a panel is hidden, you can display it by ticking its checkbox under **Layout** (highlighted above)

Entering commands

- Commands are typed into the **Command Window** and are executed (run) by pressing **Enter**

- Example of arithmetic commands:

```
a) 10 / 2

b) 5 + 3

c) 1.5 * 2

d) 2 ^ 2
```

Arithmetic operators

Operation	Operator
Add	+
Subtract	-
Multiply	*
Divide	/
Power	^

What about calculating the root (e.g. square root of 4)?

```
>> 4^(1/2)
```

```
>> sqrt(4)
```

Entering text data (character arrays/strings)

- Start and end with apostrophes (`'`)

- Example:

```
>> 'Hello World'
```

- Often used to enter optional settings (more on this in a later lecture)

Note: For those familiar with programming, *character arrays* are technically not the same as *strings* (which were introduced 2.5 years ago)

Exponents

- You can also enter numbers with exponents using scientific notation

`1e10` means 1 x 10^{10}

Which of the following commands equals 730 nm?

```
(A)  730e-6

(B)  730e-3

(C)  730e-9

(D)  730^-6
```

Precedence of arithmetic operators

What do you think is the output of the following command?

```
>> 5 + 2 / 2
```

(A)   3.5

(B)   6

(C)   20

(D)   Don't know

Precedence of arithmetic operators

- The order in which operations are carried out is defined by **precedence**

Power > division and multiplication > addition and subtraction

```
>> 5 + 2 / 2
```

In this case, MATLAB will compute the division first:

```
5 + 1
```

Then the addition:

```
6
```

Controlling the order of evaluation

- You can control the order of evaluation using parentheses ( )

- For example:

```
>> (5 + 2) / 2
```

You can use as many of these you need but remember to match them - there must be as many open (as close)

```
((5 + 2) / 2) + 1
```

Quick Summary

- Looked at different parts of the MATLAB interface
- Carried out simple arithmetic commands
- Learnt about operator precedence

Questions?

Variables are used to store data

- Variables are **objects that store data**

  Note: Not the same meaning as variables in equations

- Variables are created ("declared") using the assignment operator (=)

```
 >> width = 10
>> height = 2 * 3
```

- Declared variables appear in the **Workspace**
  - Double-clicking on a variable will open a spreadsheet (Don't do this with large variables)

Variable names

- MUST start with a letter

- Can only contain letters, numbers and underscores (_)

Valid	Not valid
`var123`	`123var`
`var13_2`	`var12+3`

- Capitalization matters

`Variable ≠ variable ≠ VARiabLE`

Using variables

- To get the value, just **type in the variable name:**

```
>> width

>> height
```

- You can use variables in expressions

```
>> area = width * height
```

Changing variable values

- Overwrite a variable by reassigning its value

```
>> width = 10
>> width = 50
```

A special variable `ans`

What happens when you do not explicitly declare a variable?

```
>> 5 + 2
```

Hint: Look in the Workspace

Answer: MATLAB will create a variable called `ans` (short for answer)

Important!
Never use `ans` as a variable name as it could be overwritten

Functions and clearing variables

- Use the function `clearvars()` to **clear** **var**iable**s** from the Workspace

```
>> clearvars
```

Getting help in MATLAB

- MATLAB has a very extensive documentation system that you can access using `doc()` or `help()`

- Syntax (a.k.a. how to call the function):

```
doc <function name> or help <function name>
```

Example:

```
doc clearvars
```

* Tip: Get familiar with these functions, you will be using them a lot

Quick Summary

- Declared variables
- Reassigned variable values
- Cleared variables from the Workspace using the function `clearvars()`
- Accessing MATLAB documentation using `doc()` and `help()`
- Never use `ans` as a variable name!

Questions?

Matrices and vectors

- MATLAB stands for **MAT**rix **LAB**oratory - originally written to solve linear equations

- Matrices are arrays of numerical data

$$M = \begin{bmatrix} 1 & 53 & 20 & 39 & 7 \\ 12 & 2 & 39 & 8 & 5 \\ 2 & 38 & 1 & 92 & 0 \\ 3 & 9 & 48 & 7 & 6 \end{bmatrix}$$

- Start by looking at a special case – **1D matrices** or **vectors**

Row vectors

- Row vectors consist of numbers ("elements") in a single row

$$R = \begin{bmatrix} 1 & 5 & 8 & 3 \end{bmatrix}$$

- To declare a row vector:

1. Start with an open square bracket `[`
2. Enter elements separated by a **space** and/or a **comma** (`,`)
3. End with a close square bracket `]`

```
>> R = [1 5 8 3]
```

or

```
>> R = [1, 5, 8, 3]
```

Column vectors

- Column vectors consist of numbers in a single column

$$C = \begin{bmatrix} 1 \\ 5 \\ 8 \\ 3 \end{bmatrix}$$

To declare a row vector:

1. Start with an open square bracket `[`
2. Enter elements separated by a **semicolon** (`;`)
3. End with a close square bracket `]`

```
>> C = [1; 5; 8; 3]
```

Declaring matrices

- To declare a matrix:
  - Enter one row at a time
  - End each row, except the last, with a semicolon (;)

Example:

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

```
>> M = [1 4; 3 6]
```

Arithmetic operations on vectors

```
>> R = [1 5 8 3]

>> R + 2

>> R * 3
```

- (Most) operators will work on each element in the vector individually
- The power operator ^ will give you an error – more on this next week)

Adding elements to a vector

- Growing a row vector:

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

- Growing a column vector:

```
>> B = [1; 2; 3]
>> B = [B; 4]
```

Adding elements to a vector

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

Which of the following will produce

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

```
a) C = A + B

b) C = [A B]

c) C = A B

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

Retrieving elements

- Each element has an **index** that indicates its position

$$\begin{array}{lcccc} R = \big[& 1 & 5 & 8 & 3 \big] \\ \text{Index} & 1 & 2 & 3 & 4 \end{array}$$

- Use the index to refer to the element
- Example: To retrieve the second element in vector `R`

```
>> R(2)
```

- Indices start from 1 (not 0 like other programming languages)

Retrieving multiple elements

- Declare the following vector:

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

- To index the first five elements:

```
>> firstFive = F([1, 2, 3, 4, 5])
```

Retrieving multiple elements

- Declare the following vector:

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

- To index the first five elements:

```
>> firstFive = F(1:5)
```

The colon operator (:) generates a **number range**

```
1:5 = [1, 2, 3, 4, 5]
```

Modifying elements

- Modify individual elements by reassigning to a different value

$$R = \begin{bmatrix} 1 & 5 & 8 & 3 \end{bmatrix}$$

```
>> R = [1, 5, 8, 3];

>> R(3) = 10;
```

```
R = [1, 5, 10, 3]
```

Deleting an element

- Delete an element in a matrix by assigning it to an empty matrix [ ]

$$R = \begin{bmatrix} 1 & 5 & 8 & 3 \end{bmatrix}$$

```
>> R(3) = []
```

What is the value of R(3)?

```
R(3) = 3
```

Why?

```
R = [1 5 3];
```

Special keyword `end`

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

end refers to the element in the last index

```
>> F(end)
```

- You can use end to index as well:

```
>> F(5:end)
```

Quick Summary

- Declaring vectors
- Modifying and deleting elements in vectors
- Indexing elements in vectors

Questions?