Introduction to Programming for Biological Research
I study cell division in bacteria:
I use MATLAB a lot:

Particle Tracking:

Image Analysis:

Simulation:
The Story So Far…

• What programming is, and what MATLAB does
• How to write algorithms for computational tasks
• Fundamentals of MATLAB syntax: indexing, conditional statements, loops
• Data structures, variable types, arrays, cell arrays
• Writing scripts and functions
Today’s Topics:

What should your program do?

• You want to be able to give your program an **input** (your data), have it process that input in a specific way, and then return a useful **output**.

• You want to be able to use it **repeatedly** on many inputs without having to change the code itself.

• You want it to be **user friendly**: it should be well documented, and throw useful error messages.
Today’s Topics:

- Inputs and Outputs
- Making Functions Reusable
- Projects
Opening Files in MATLAB

To open individual files, just click and drag:
Opening Files in MATLAB

To open individual files, just click and drag:
Opening Files in MATLAB

This can be very useful for Excel files:
Opening Files in MATLAB

This can be very useful for Excel files:
Opening Files in MATLAB

This can be very useful for Excel files:
Opening Files in MATLAB

There are also ways to open files *programmatically*

\[ v = \text{open}('\text{ParticleVelocities.xlsx}') \]
Opening Files in MATLAB

There are also ways to open files programmatically

```matlab
v = open('ParticleVelocities.xlsx')
```

1. Loads the file ParticleVelocities.xlsx into MATLAB

2. Assigns the results to a variable named v.
Opening Files in MATLAB

There are also ways to open files programmatically:

```matlab
v = open(‘ParticleVelocities.xlsx’)
```

There are more specialized ways of opening specific types of files:

```matlab
img = imread(‘Cells1.jpg’)

v = xlsread(‘ParticleVelocities.xlsx’)

uiopen
uigetfile
```
Practice!

Create an Excel spreadsheet:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

Save it in your working directory:

First, write a script that:
1. Opens your Excel spreadsheet in MATLAB (using \texttt{xlsread})
2. Plots column A vs column B (using the \texttt{plot} function)
Practice!

First, write a script that:
1. Opens your Excel spreadsheet in MATLAB (using `xlsread`)
2. Plots column A vs column B (using the `plot` function)

My Solution:

```matlab
myfile = xlsread('myspreadsheet1.xlsx');
plot(myfile(:,1),myfile(:,2));
```
Practice!

Now, turn your script into a function that:
1. Takes the name of a spreadsheet file and a plot color as an input (‘r’, ‘g’, ‘b’, ‘k’ etc.)
2. Plots column A vs column B of the file
3. Returns the average value of column B and the number of rows in the array

Your function should work for spreadsheets of different sizes and with different names.
Practice!

Now, turn your script into a function that:
1. Takes the name of a spreadsheet file and a plot color as an input (‘r’,’g’,’b’,’k’ etc.)
2. Plots column A vs column B of the file
3. Returns the average value of column B and the number of rows in the array

My Solution:

```matlab
function [sAvg,sLength] = myfun(filename,graphcolor)
    myfile = xlsread(filename);
    plot(myfile(:,1),myfile(:,2),'Color',graphcolor);
    sAvg = mean(myfile(:,2));
    sLength = size(myfile,1);
end
```

```
>> [myAverage1, myLength1] = myfun('myspreadsheet1.xlsx','r')
```
Practice!

Finally, write a script that:
1. Creates a list of file names of 3 Excel spreadsheets in your directory (I suggest that you use a cell array)
2. Runs your function for each file
3. Makes a list of the averages and lengths of each spreadsheet
Practice!

Finally, write a script that:
1. Creates a list of file names of 3 Excel spreadsheets in your directory (I suggest that you use a cell array)
2. Runs your function for each file
3. Makes a list of the averages and lengths of each spreadsheet

My Solution:

```matlab
filenames = {'myspreadsheet1.xlsx','myspreadsheet2.xlsx','myspreadsheet3.xlsx'};
allAvgs = [];
allLengths = [];
for i = 1:length(filenames)
    [allAvgs(end+1), allLengths(end+1)] = myfun(filenames{i},'r');
end
```
Final Projects

Goal: design a program for biology research!

Four weeks from now, you will have:

• Written, commented code addressing a question or solving a problem in biology research

• A README for users that explains how to run your code

• A 5-10 minute presentation about your project
Topics

We encourage you to be creative and draw inspiration from your own research when designing a project.
Topics

We encourage you to be creative and draw inspiration from your own research when designing a project.

Sample project:

**Image Classifier:** It is common in microscopy labs to want to classify images. For instance, you might want to look through a data set of images and classify them as “labeled” or “unlabeled”. Write code in MATLAB that takes as input a directory full of images and shows them to the user one at a time. The user can then type in their classification for each image. MATLAB should save the user’s input and, at the end, show some helpful information (for instance, what percent of images were “labeled”.)

- Week 1: Outline algorithm, write code to read images from a directory
- Week 2: Code to show images to the user and get user classification
- Week 3: Output code, error handling
- Week 4: README and presentation
Final Projects

This week: Form groups (2-3 people), choose your topics, and outline your project plan

By next Wednesday, e-mail Georgia with your group members and a description of your proposed project. I will approve your proposal (or suggest revisions if needed), and assign an instructor to supervise your project.
Final Projects

Weekly Check-Ins:

Each Wednesday, you should submit your (commented!!!) code for your project, as well as a short description of your progress for the week, along with any questions you have.

Your code should be submitted to your supervising instructor, who will respond with advice and answers to any of your questions.
Final Projects

Got questions along the way?

- **Monday review sessions** are a great place to bring your questions and work together with members of your group.

- **Your supervising instructor** can answer your questions by e-mail, and will give weekly feedback.

- **MATLAB’s online documentation** has very detailed function descriptions and the answers to many common questions.
Final Projects

Final Presentations

• At the end of your project, you will submit your commented program, as well as a README telling us how to run it
• Your group will also give a 5-10 minute presentation to the class about your project
Final Projects

Final Presentations

• At the end of your project, you will submit your commented program, as well as a README telling us how to run it.
• Your group will also give a 5-10 minute presentation to the class about your project.

• Things to discuss in the presentation:
  • The inspiration for your project
  • How your program works
  • Your experience writing the program
  • Improvements you would make in the future