Lecture 38:

Data analysis tips

Lecturer: Jian Wei Tay

Date: 19 November 2021

Learning Objectives

- Review of core concepts from Problem Set 8
  - Analyzing multichannel images – marker and sensor
  - Adding data to a vector within a for loop
- Using debugging functions to inspect a for loop
- Measuring time-series statistics from tracked data

Problem Set 8

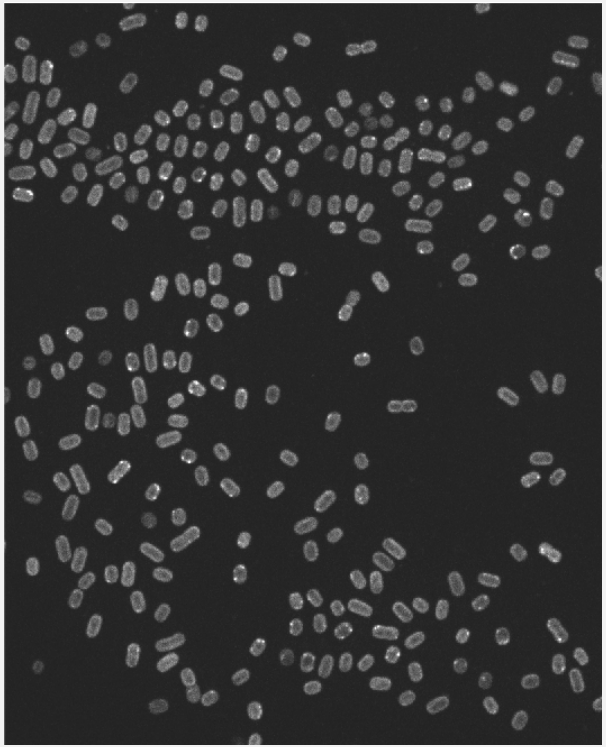
Question 1

- You were given two images:

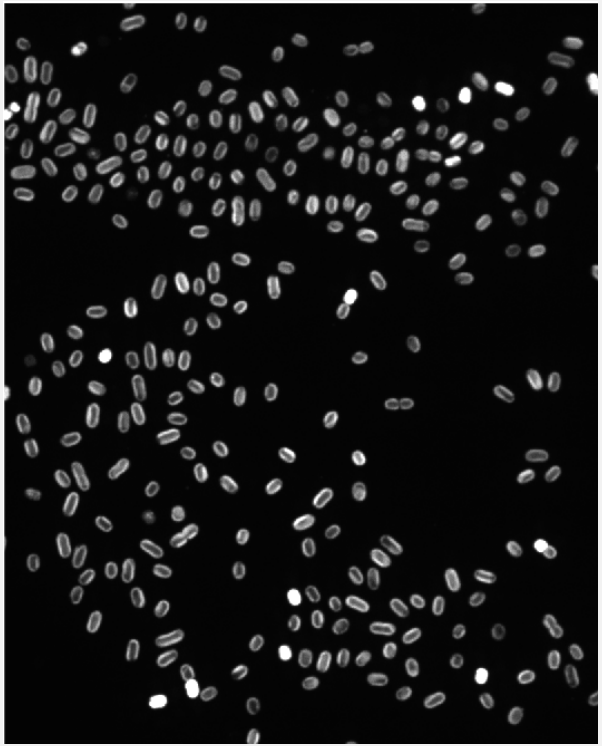
  The image `ps8_cyanobacteria_chl.tif` shows the fluorescence arising from natural fluorescence of the chlorophyll pigments, while the image `ps8_cyanobacteria_mNeonGreen.tif` shows the fluorescence from a cell marker that can be used to identify each cell.

Explanation of the images

mNeonGreen marker



Chlorophyll



Why can we not use the chlorophyll image to segment cells?

Question 1

- The chlorophyll intensity of a cell that is dying will be very bright. Cells that have low chlorophyll intensity are dead.

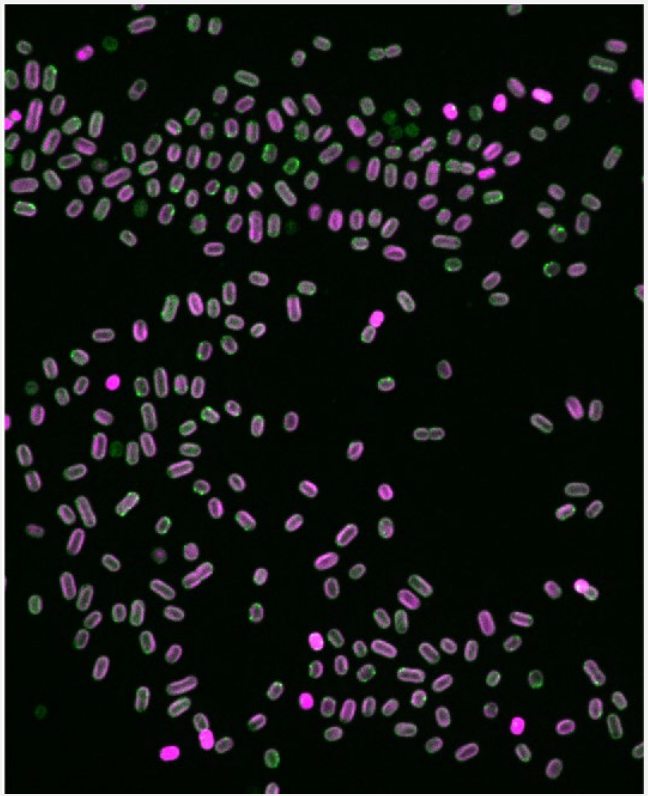
Why can we not use the chlorophyll image to segment cells?

The dead cells will not show up in the chlorophyll channel.

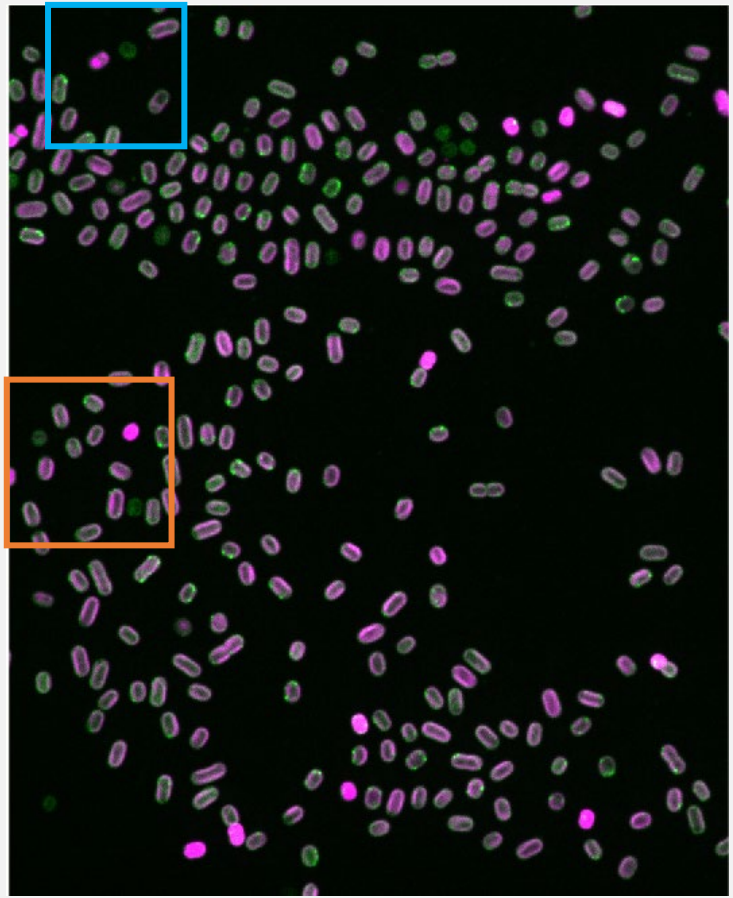
This is the same concept for any application where you are trying to measure the intensity of a sensor

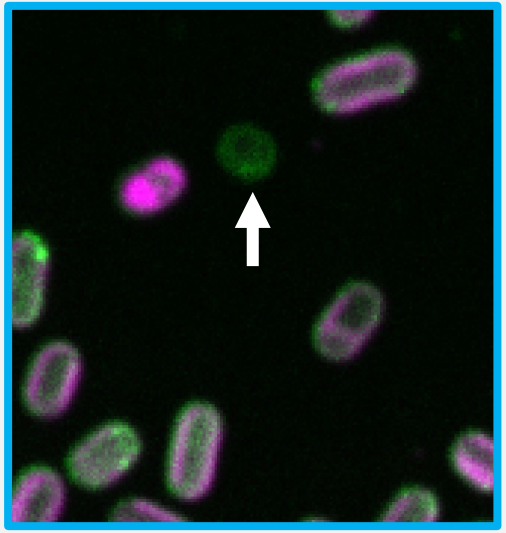
You need to have a way of segmenting a cell even when the sensor is dim. Therefore, you cannot segment cells using the sensor image.

You can use imshowpair to display a "merged" image

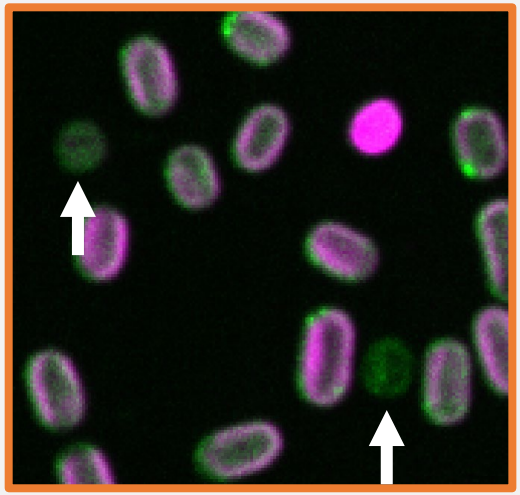


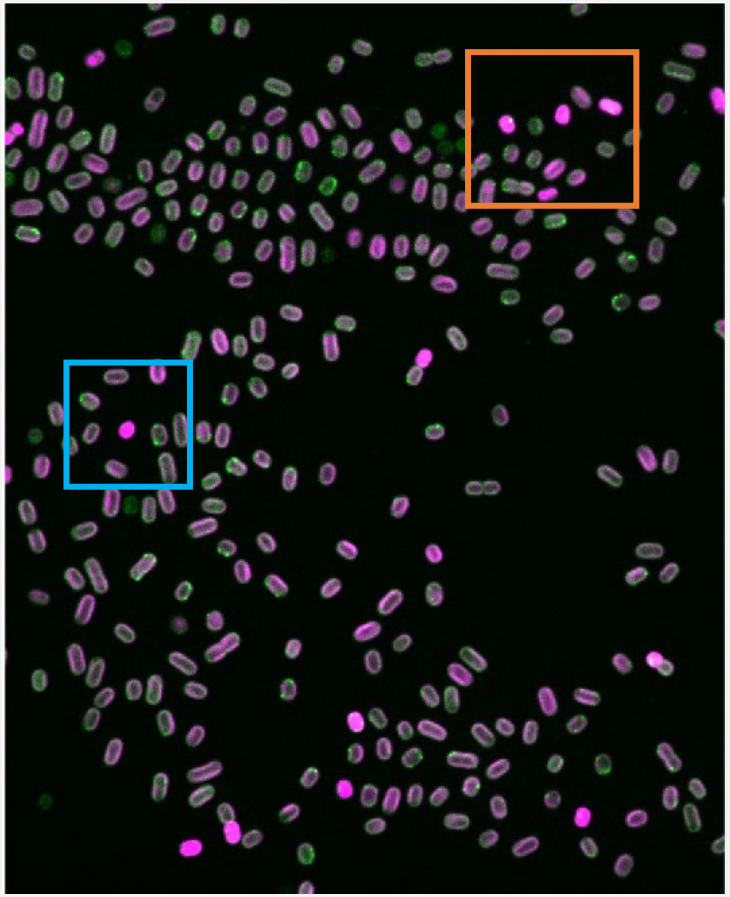
```
Igreen = ...
imread('ps8_cyanobacteria_mNeonGreen.tif');
Ichl = ...
imread('imread('ps8_cyanobacteria_mNeonGreen.
tif');


imshowpair(Igreen, Ichl)
```

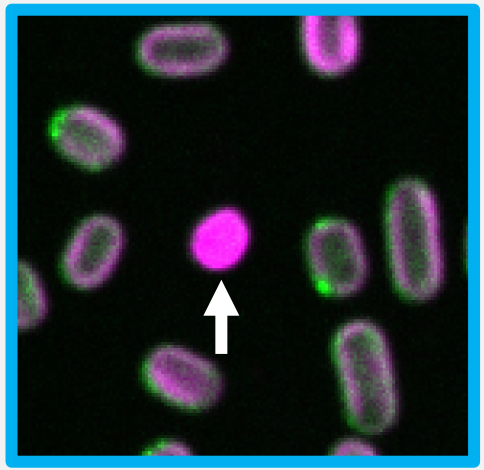


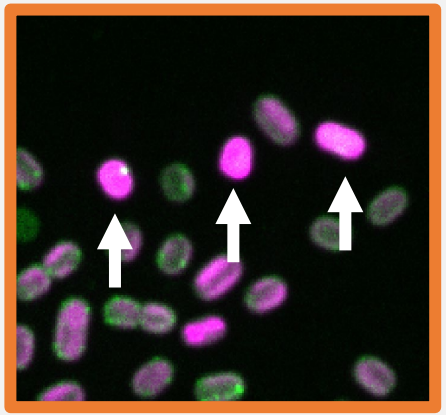


These cells are "dead"









These cells are "dying"

Measuring cell intensities

- The idea is to make a mask using the mNeonGreen image, then use the mask to measure the cell intensities in the chlorophyll channel

MATLAB Code

```
%Read in images
Igreen = imread('ps8_cyanobacteria_mNeonGreen.tif');
Ichl = imread('ps8_cyanobacteria_chl.tif');

%Make a mask using the mNeonGreen image
mask = Igreen > 700;


celldata = regionprops(mask, Ichl, 'MeanIntensity');
```

Counting dead (dim) and dying (very bright) cells

Plotting a histogram

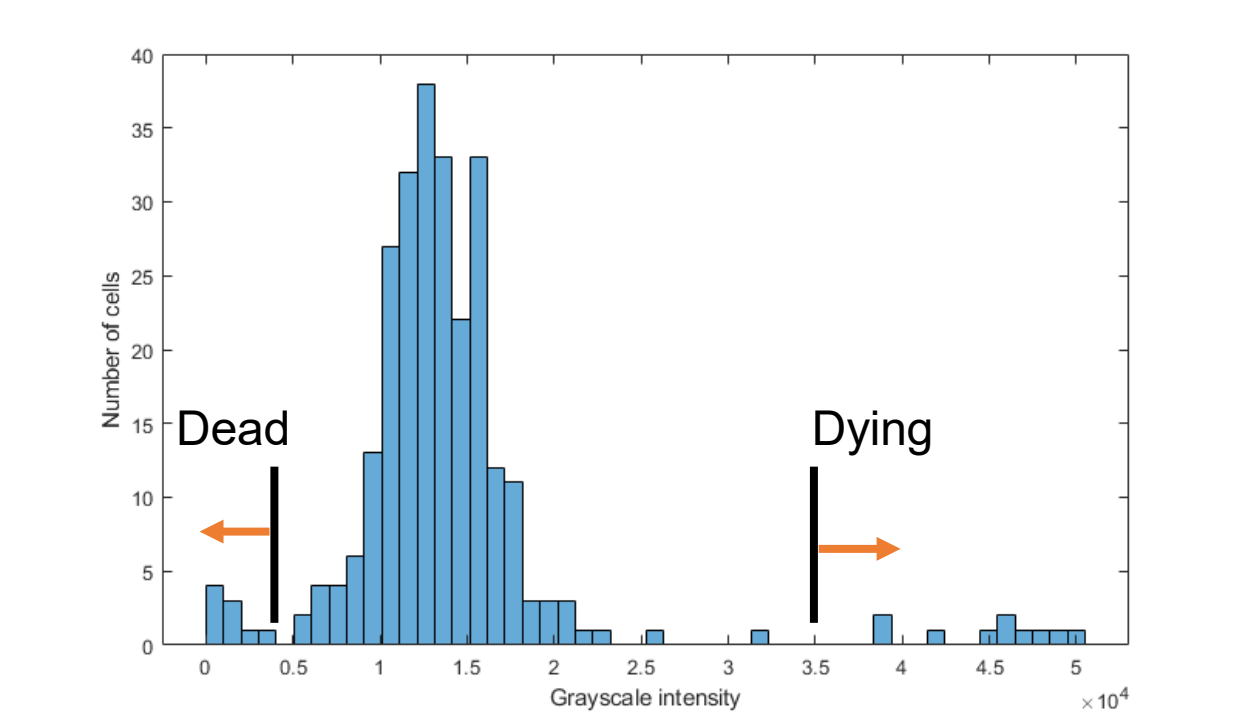
- Syntax:

```
histogram(data, numBins)
```

MATLAB code

```
data = regionprops(mask, Ichl, 'MeanIntensity');

intensities = cat(1, data.MeanIntensity)
histogram(intensities, 50)
```

Plotting a histogram



Using logical indexing to count the number of cells

```
deadCells = intensities(intensities < 0.5e4);

numDeadCells = numel(intensities);
```

Question 2

- In Question 2, you were asked to count the number of cells over time in the time-lapse dataset `PS8_MCF7.tif`

- There are two tips I want to share for analyzing a time-lapse dataset:
  - How to view a timelapse video
  - Storing the number of cells in a vector within the for loop

Viewing a timelapse video (Always good to get an idea of what data to expect)

Viewing a timelapse video

```
%Get number of frames in the movie
nFrames = numel(imfinfo('ps8_MCF7.tif'));

%Identify cells in each frame of the movie

for iT = 1:nFrames
    I = imread('ps8_MCF7.tif', iT);
    imshow(I, [])
    drawnow
end
```

Note: `drawnow` forces MATLAB to update the figure

Viewing a timelapse video

- Fiji/ImageJ: https://imagej.net/software/fiji/

Storing data in a vector within a for loop

Example code

```
nFrames = numel(imfinfo('ps8_MCF7.tif'));


for iT = 1:nFrames
  I = imread('ps8_MCF7.tif', iT);
  mask = imbinarize(I);

  data = regionprops(mask, 'Area');


end
```

Note: How to calculate the number of cells in the mask?

Example code

```
nFrames = numel(imfinfo('ps8_MCF7.tif'));


for iT = 1:nFrames
  I = imread('ps8_MCF7.tif', iT);
  mask = imbinarize(I);

  data = regionprops(mask, 'Area');
  numCells = numel(data)


end
```

Example code

```
nFrames = numel(imfinfo('ps8_MCF7.tif'));

for iT = 1:nFrames
  I = imread('ps8_MCF7.tif', iT);
  mask = imbinarize(I);

  data = regionprops(mask, 'Area');
  numCells(iT) = numel(data)

end
```

Note: You can grow a vector by indexing, e.g.:

```
>> A(1) = 1; % A = [1]
>> A(2) = 10;  %A = [1, 10]
>> A(3) = 23;  %A = [1, 10, 23]
```

Declaring a vector ahead of time

```
nFrames = numel(imfinfo('ps8_MCF7.tif'));

numCells = zeros(1, nFrames);

for iT = 1:nFrames
  I = imread('ps8_MCF7.tif', iT);
  mask = imbinarize(I);

  data = regionprops(mask, 'Area');
  numCells(iT) = numel(data)
end
```

Note: This function creates a row of zeros

Tips for writing code in a for loop

- Make sure that the code works for one iteration before moving on
- Change the number of iterations to process the first 5 frames
- Set breakpoints to inspect how the code runs line by line within the for loop

Saving variables in MATLAB

- To save your variables in MATLAB, use the function save:

`save(filename)` – saves every variable in the Workspace to a MAT-file

`save(filename, var1, var2, …)` – saves the variables var1, var2 to a MAT-file

Examples of using save

- `save('lec38data.mat')`

- `save('lec38data.mat', 'linker')`

Loading saved data

- Double-click on the .mat file to load the variables back to your workspace

- Alternatively, you can use the function load:

```
load(matFilename)
```

Note: When you double-click the mat file, MATLAB will run the `load` function for you, as shown in the Command Window.

Analyzing tracked data

For this section, we will use data in `unperturbed.mat`. This file contains tracked data from the control dataset.

Question

- The sensor in this data becomes bright when cells are not actively dividing. To compare the different datasets, we want to know how the average cell intensity changes over time.

Clarification of terminology

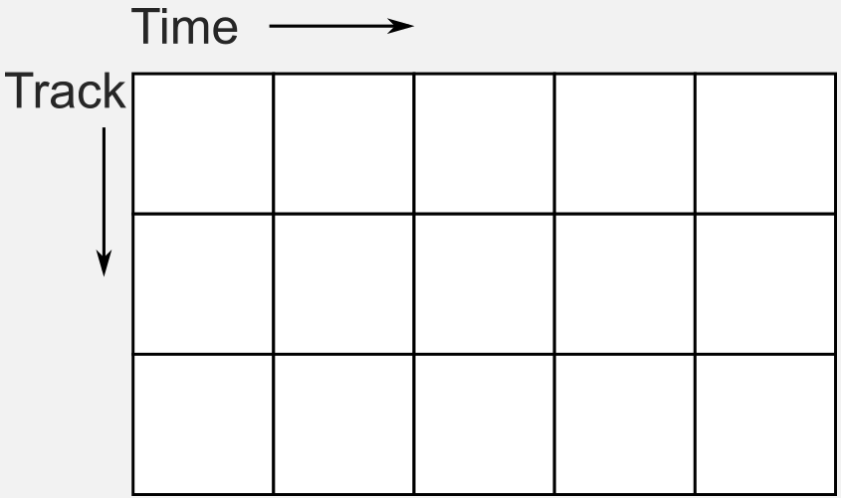
- Using `regionprops`, we measure the `MeanIntensity` property which is the average intensity of a cell
- Here we are trying to measure the mean `MeanIntensity` of every cell in a frame, for every single frame of the movie
- The challenge is that each track starts and ends at different frames…

Track properties

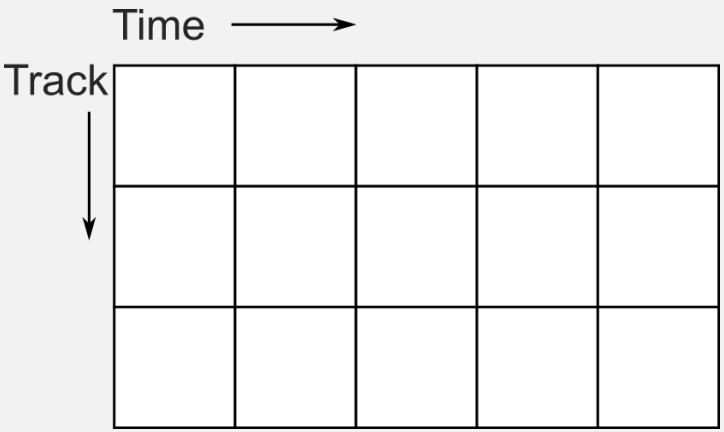
```
>> currTrack = getTrack(linker, 1)
currTrack =
  struct with fields:

                ID: 1
          MotherID: NaN
        DaughterID: NaN
            Frames: [1×30 double]
              Area: [30×1 double]
          Centroid: [30×2 double]
     MeanIntensity: [30×1 double]
```

Note: The `Frames` property tells you when this track existed.

Basic idea is to collect data into a matrix so we can use the mean function



Basic idea is to collect data into a matrix so we can use the mean function

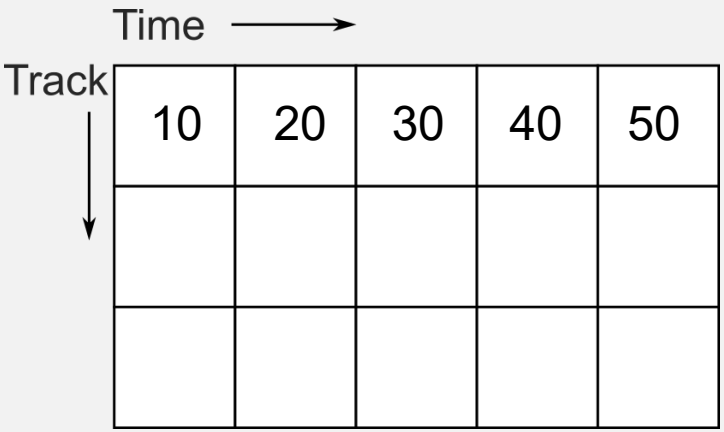


Example:

```
track1.Frames = [1 2 3 4 5];
track1.MeanIntensity = [10 20 30 40
50];
```

How to add this data into this matrix?

Basic idea is to collect data into a matrix so we can use the mean function



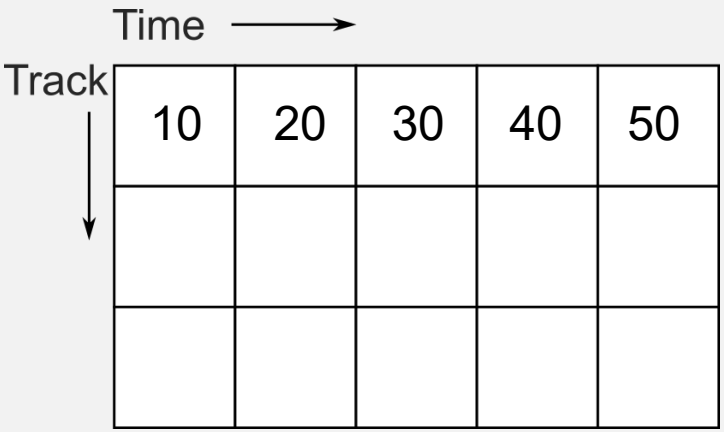
Example:

```
track1.Frames = [1 2 3 4 5];
track1.MeanIntensity = [10 20 30 40
50];
```

How to add this data into this matrix?

```
M(1, track1.Frames) = ...
        track1.MeanIntensity
```

Basic idea is to collect data into a matrix so we can use the mean function

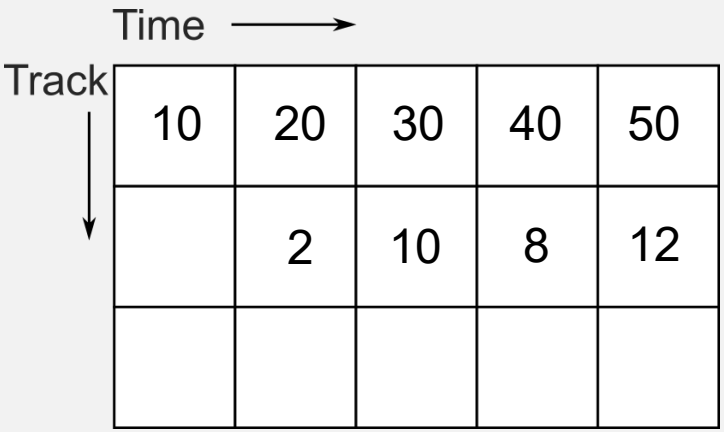


Example:

```
track2.Frames = [2 3 4 5];
track2.MeanIntensity = [2 10 8 12];
```

How to add this data into this matrix?

Basic idea is to collect data into a matrix so we can use the mean function



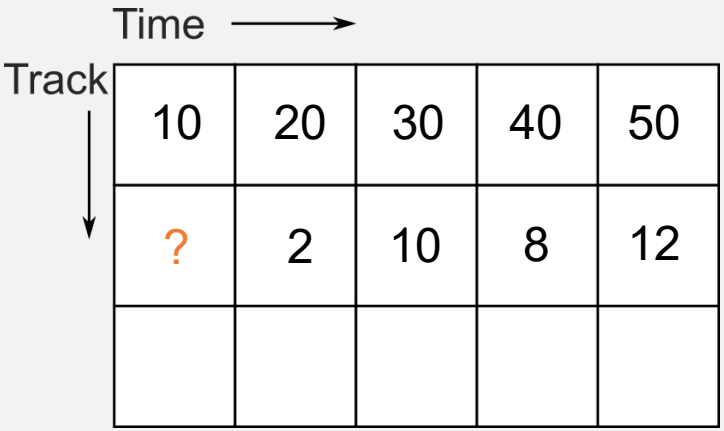
Example:

```
track2.Frames = [2 3 4 5];
track2.MeanIntensity = [2 10 8 12];
```

How to add this data into this matrix?

```
M(2, track2.Frames) = ...
       track2.MeanIntensity
```

Basic idea is to collect data into a matrix so we can use the mean function



But… a matrix MUST be rectangular.

So what value should be put in M(2, 1)?

Not-a-Number (NaN)

- NaN stands for Not-a-Number – this is an internal MATLAB designation for a special value that is not a number.

- Essentially any arithmetic operation involving a NaN returns a NaN:

```
>> 10 - NaN
ans =
   NaN
```

MATLAB's statistics functions can omit NaNs

```
>> X = [1 2 3; NaN 3 6; 4 6 NaN; 4 7 7];
```

1	2	3
NaN	3	6
4	6	NaN
4	7	7

MATLAB's statistics functions can omit NaNs

```
>> mean(X, 1, 'omitnan')
ans =
    3.0000    4.5000    5.3333
```

Mean of elements in first column:

(1 + 4 + 4)/3 = 3

1	2	3
NaN	3	6
4	6	NaN
4	7	7

Why not use zeros?

```
>> X = [1 2 3; 0 3 6; 4 6 0; 4 7 7]
X =
     1     2     3
     0     3     6
     4     6     0
     4     7     7
>> mean(X, 1)
ans =
    2.2500    4.5000    4.0000
```

Note: A zero would affect the calculation of the mean., e.g. the mean of the first column = (1 + 0 + 4 + 4) /4

Declaring a matrix of NaNs

- M = nan(numRows, numCols)

- Example: `M = nan(10, 3)` generates a matrix of NaNs with 10 rows and 3 columns.

Example code

- For the example we looked at with the two tracks:

```
%Declare a matrix of NaNs
allIntensities = nan(2, 5);

%Index the intensity data using the Frames property to
%specify the columns
allIntensities(1, track1.Frames) = track1.MeanIntensity
allIntensities(2, track2.Frames) = track2.MeanIntensity
```

Problem Set 10

- The challenge for you to solve in PS10 is to use this method to plot the average intensity of each cell in a frame over the entire movie
- Hint: You should use a for loop