ENGR 101 – Chatper 6

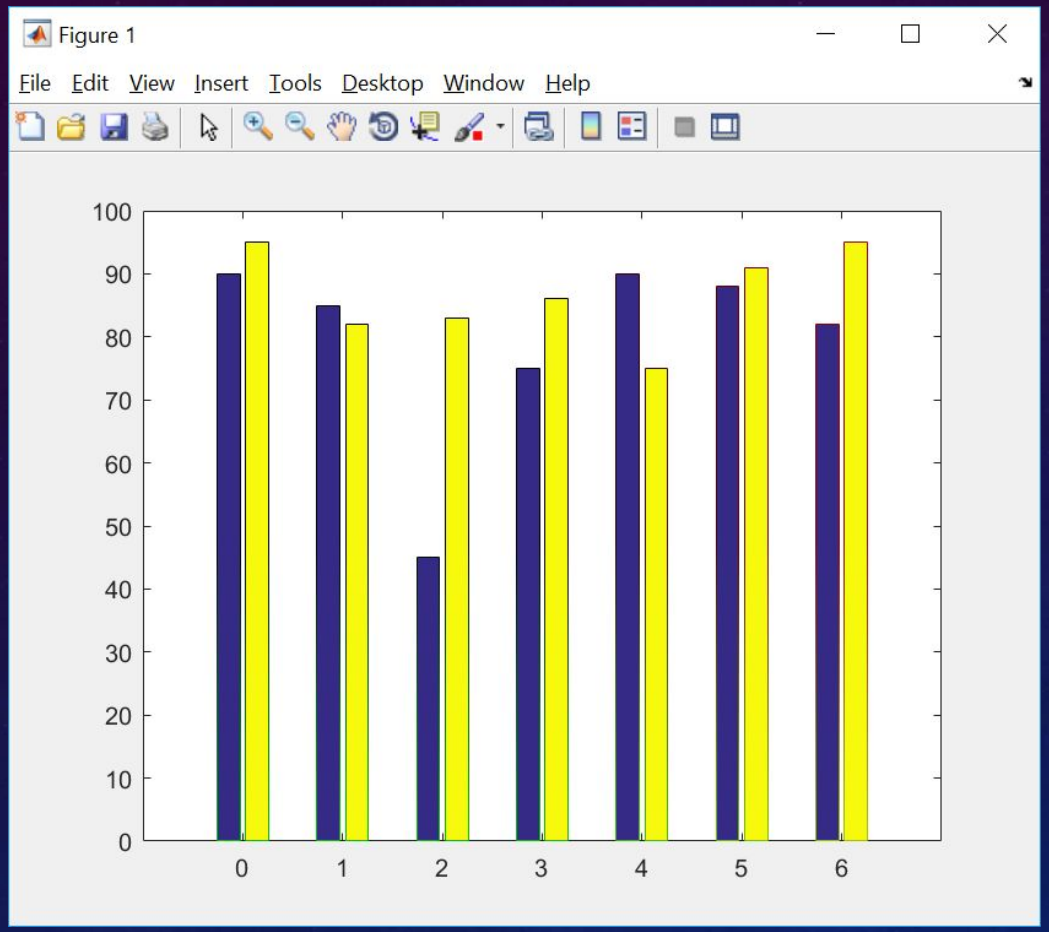
Plotting and Figures

Laura Alford, James Juett, Rick Niciejewski 9/19/2020

Plotting

No, not that kind of plotting.

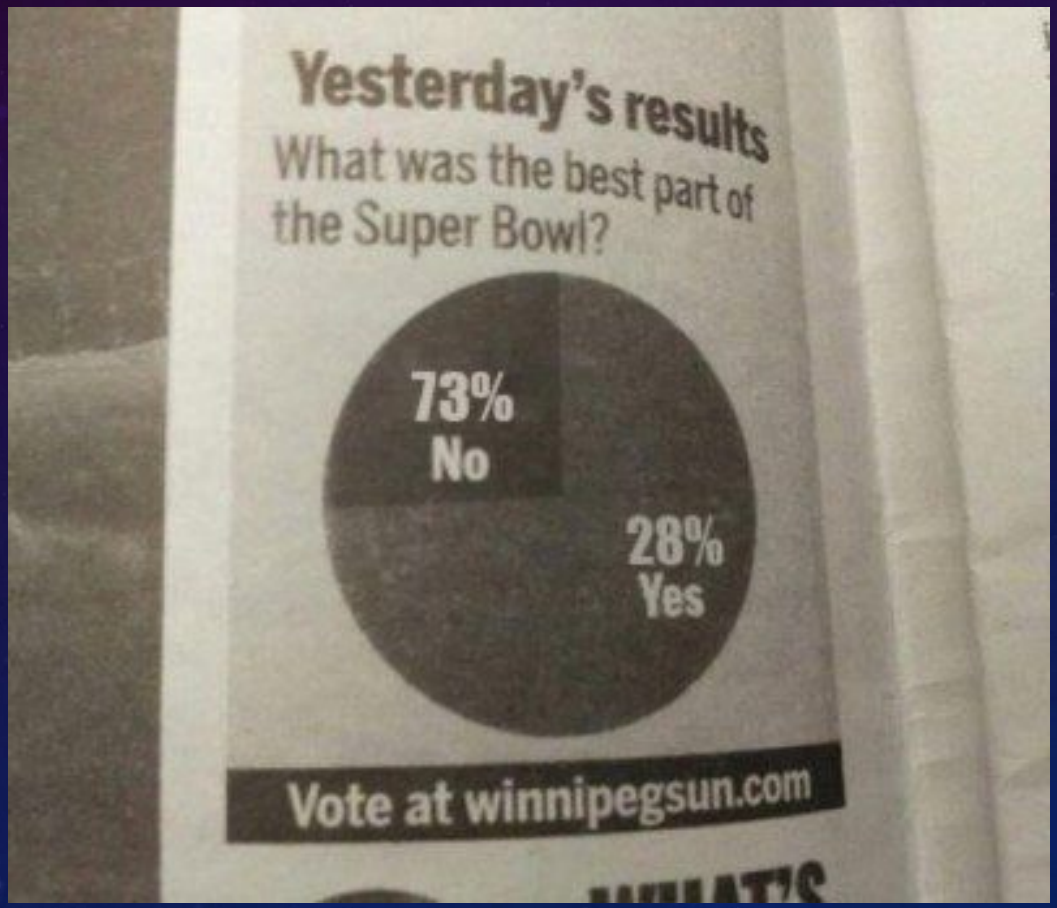
What's wrong with this plot?



How is anyone going to know what this plot is supposed to be?

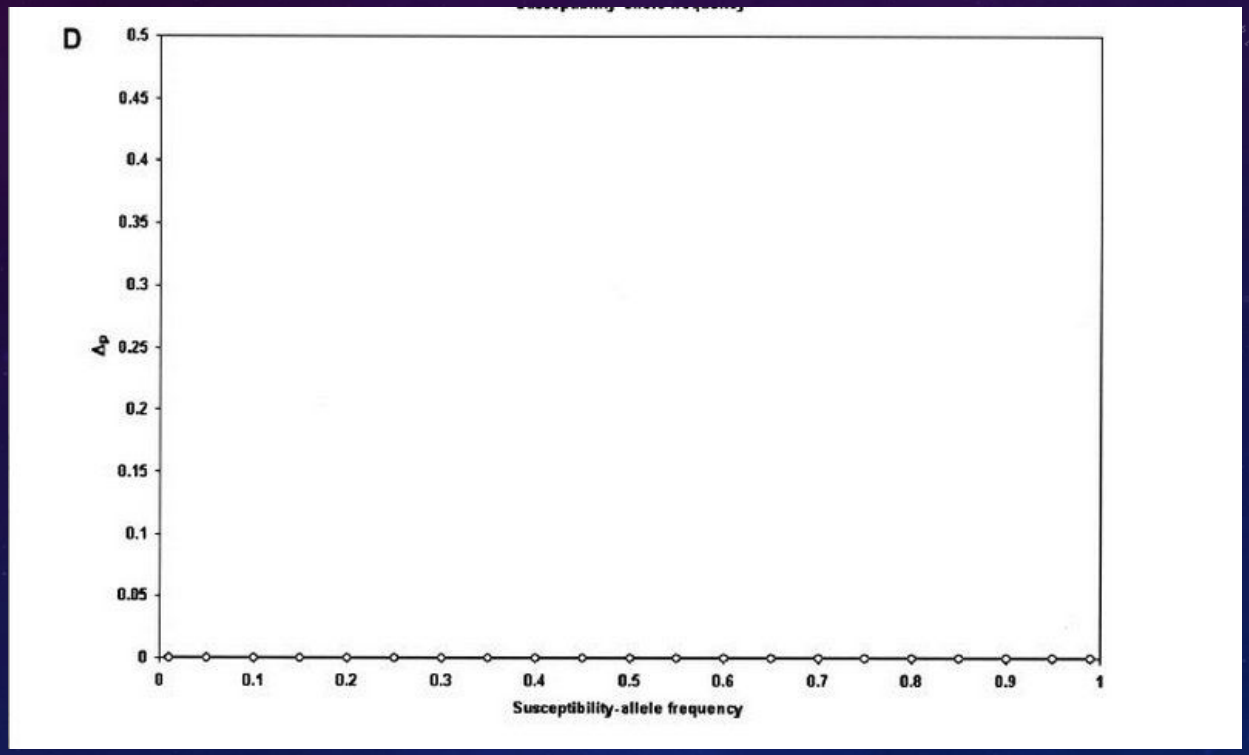
Motivation for Making Good Plots

- Because someone did this



Motivation for Making Good Plots

- Because someone did this, and this



Wittke-Thompson JK, Pluzhnikov A, Cox NJ (2005) Rational inferences about departures from Hardy-Weinberg equilibrium. *American Journal of Human Genetics* 76:967-986

Motivation for Making Good Plots

- Because someone did this, and this, and this

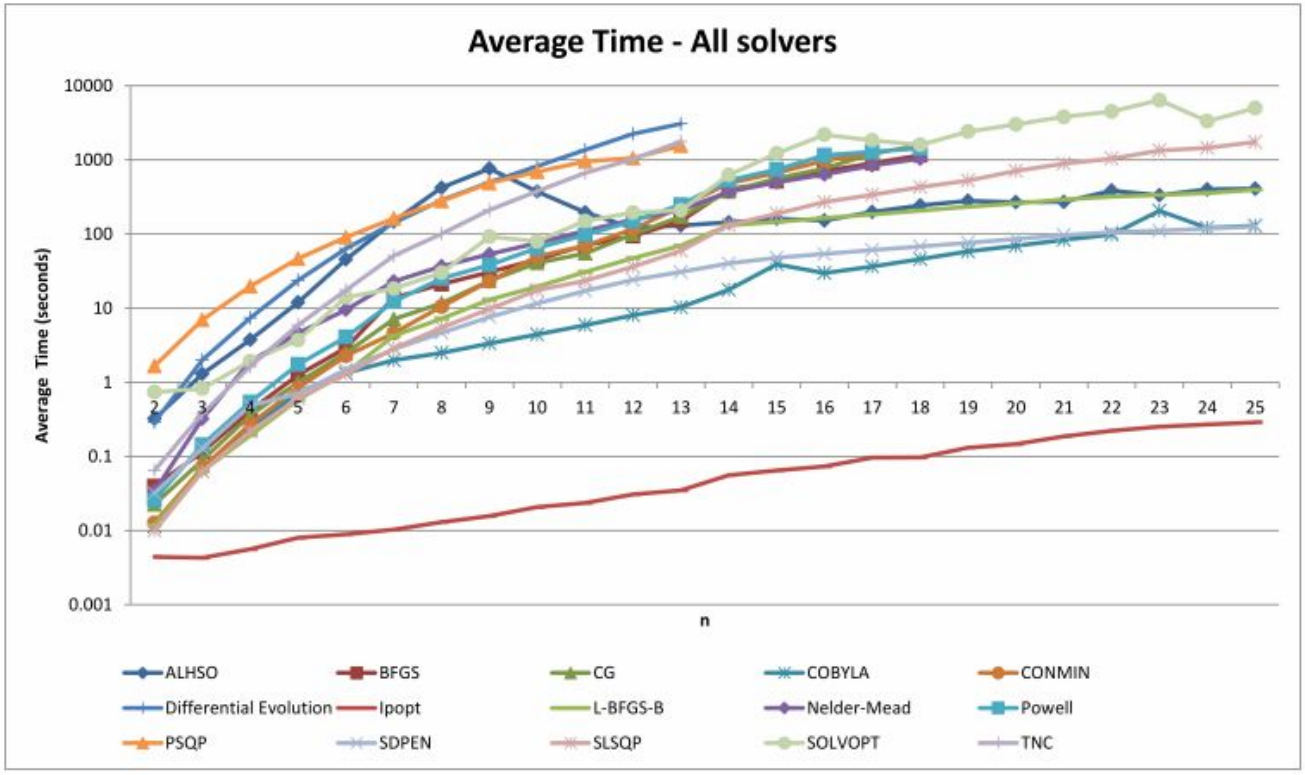
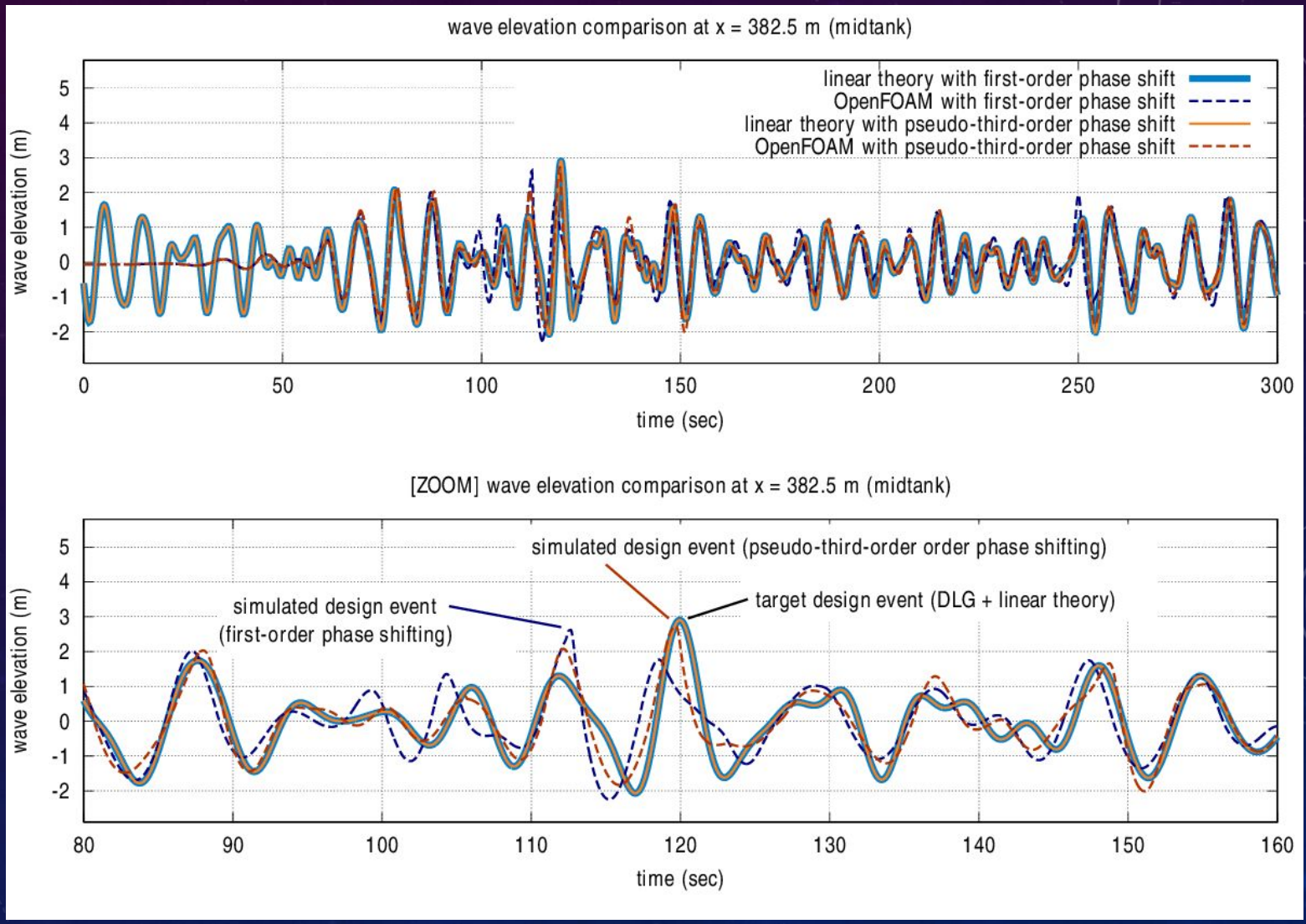


Figure 17: All solver times

Not Just Plots -- Professional Plots!



Recall: Calculating ESP From Data Vectors

- Our measurements of chemicals in the soil are encoded into column vectors, which are passed into the ESP function.

```
Na = [10.9; 13.7; 14.3; 14.1; 12.3; 12.6; 14.1; 12.0; 14.5; 12.1];
K = [68.2; 66.3; 67.0; 72.2; 72.3; 67.9; 71.5; 72.1; 71.4; 73.5];
Ca = [25.4; 26.4; 25.4; 26.7; 25.5; 26.8; 26.5; 26.9; 26.7; 25.7];
Mg = [13.8; 13.2; 14.1; 13; 17.3; 13.1; 17.7; 13; 15.6; 15];

display(ESP(Na, K, Ca, Mg));
```

- A further improvement might be to read these data vectors out of a file or some other source. Let's do this!

Reading Data from a CSV File

- CSV stands for "Comma Separated Values"
- These files contain one line for each row of data, with the entries of each row separated by commas.
- The file may contain headers and/or labels for the data.
- These files generally have the extension `.csv`

```
Day,Sodium,Potassium,Calcium,Magnesium
1,13.417,71.909,25.758,9.5634
2,12.62,72.405,28.038,13.537
3,13.627,71.302,23.048,9.3256
4,9.6751,65.028,23.947,9.9358
5,11.527,68.825,23.165,8.579
...
```

soil_samples_2019.csv

File I/O: csvread

```
Day,Sodium,Potassium,Calcium,Magnesium
1,13.417,71.909,25.758,9.5634
2,12.62,72.405,28.038,13.537
3,13.627,71.302,23.048,9.3256
4,9.6751,65.028,23.947,9.9358
5,11.527,68.825,23.165,8.579
...
```

- The `csvread` function reads data from a CSV file into a matrix.
- `csvread` only works for numeric data (not strings/text)!
- Optional parameters allow you to skip a certain number of rows/columns (i.e. header data).

Don't skip any columns for this file

Skip 1 row to start reading data at row 2 in the file

```
samples = csvread('soil_samples_2019.csv', 1, 0);



whos samples;

  Name          Size                Bytes  Class     Attributes
  samples       155x5                6200  double
```

CAUTION
These numbers are 0-indexed, instead of 1-indexed!

Extracting Parallel Vectors

- Download the `soil_samples_2019.csv` file from the Google Drive.
- Create a new script to extract the data into parallel vectors.

```
% first, clear any existing data from the workspace
clear

% read the data, skipping 1 row (the header)
samples = csvread('soil_samples_2019.csv', 1, 0);

% extract individual columns
days = samples(:,1);
sodium = samples(:,2);
potassium = samples(:,3);
calcium = samples(:,4);
magnesium = samples(:,5);
```

Recall: A Function to Calculate ESP

ESP.m

- Let's write a function that calculates the Exchangeable Sodium Percentage (ESP) from the practice project. Here's the interface:

The result is in e, so its value should be returned.

Name the function.

Our function takes several parameters for each of the chemicals found in the soil.

```
function [ e ] = ESP( Na, K, Ca, Mg )

  e = Na ./ (K + Ca + Mg + Na);

end
```

The first line is often called the **function header**.

The implementation computes the result from the parameters.

- Now, in the implementation of the function, we write code that uses the parameters Na, K, Ca, and Mg to calculate e.

1 min

Minute Exercise: Calculate ESP for All Samples

add code here to call the ESP( ) function and create a parallel vector of ESP values for each soil sample

```
% first, clear any existing data from the workspace
clear

% read the data, skipping 1 row (the header)
samples = csvread('soil_samples_2019.csv', 1, 0);

% extract individual columns
days = samples(:,1);
sodium = samples(:,2);
potassium = samples(:,3);
calcium = samples(:,4);
magnesium = samples(:,5);

% calculate ESP
```

Remember that the ESP function is vectorized! It works for vectors the same way as for scalars.

Minute Solution: Calculate ESP for All Samples

```
% first, clear any existing data from the workspace
clear

% read the data, skipping 1 row (the header)
samples = csvread('soil_samples_2019.csv', 1, 0);

% extract individual columns
days = samples(:,1);
sodium = samples(:,2);
potassium = samples(:,3);
calcium = samples(:,4);
magnesium = samples(:,5);

% calculate ESP
ESP_values = ESP(sodium, potassium, calcium, magnesium);
```

Remember that the ESP function is vectorized! It works for vectors the same way as for scalars.

Recall: The `plot` Function

- The `plot` function takes two sets of (x,y) ordinates and connects the data points, creating a line plot.

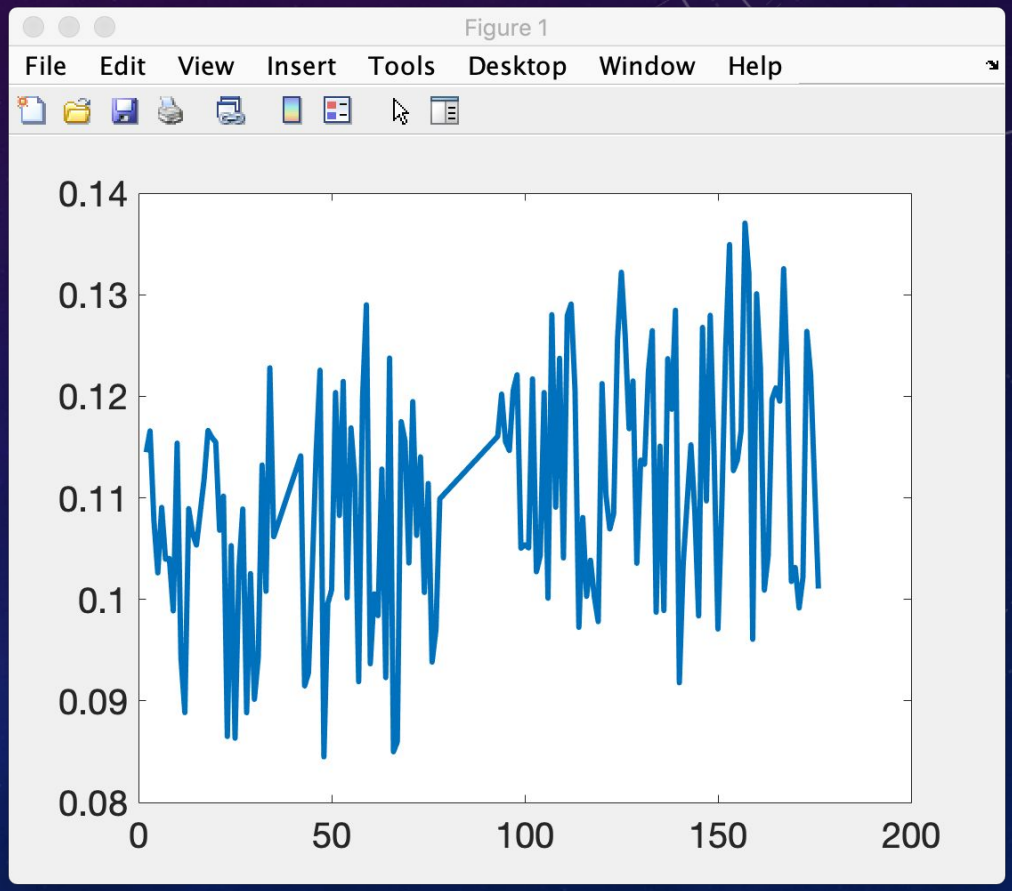
```
plot(x,y);
```

horizontal axis values

vertical axis values

- The (x,y) data can represent anything you want to see plotted… but make sure the vectors have the same length!

```
plot(days,ESP_values);
```



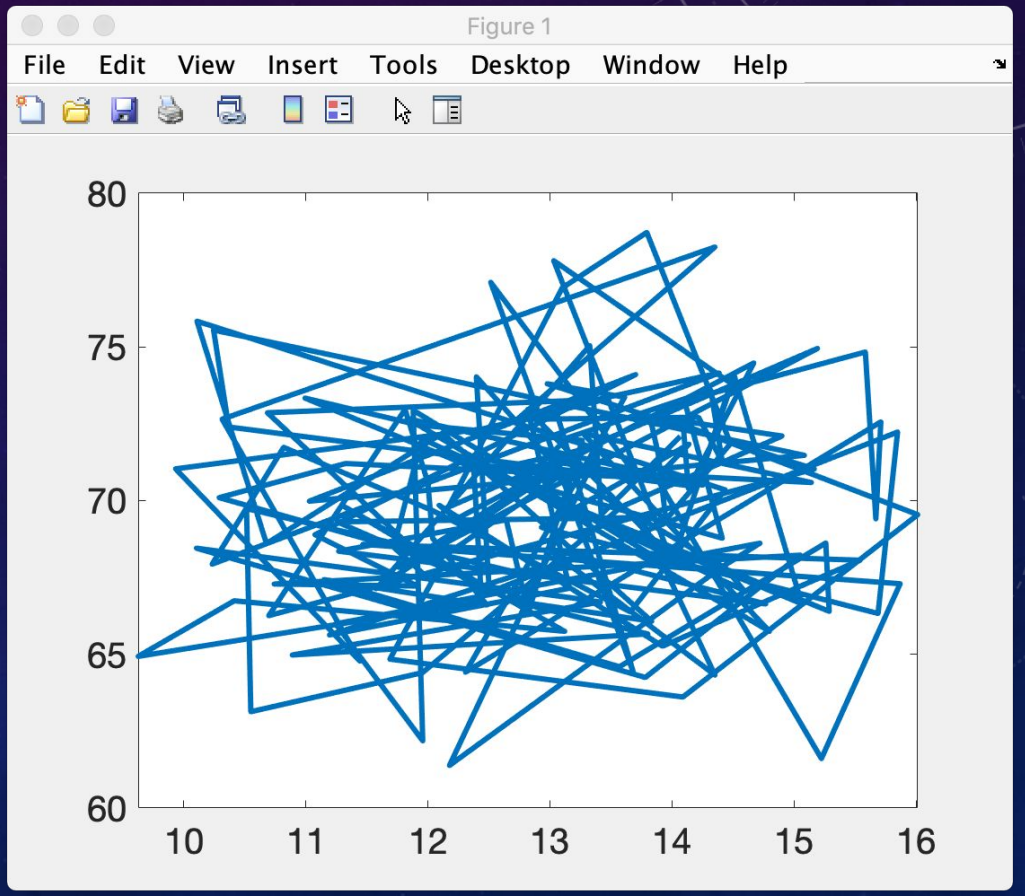
plot - Order Matters!

- When calling the `plot` function, the ordering of the (x,y) pairs matters!

sodium	potassium
13.417	71.909
12.62	72.405
13.627	71.302
9.6751	65.028
11.527	68.825
...	...

these data pairs are ordered by increasing *day*, not increasing sodium ☹

```
plot(sodium, potassium);
```



Creating a Scatterplot

- Sometimes, there is no inherent order to the data set – you just want to see all the data pairs. To create a scatterplot, use the `scatter` function.

optional parameter: size (in pixels)

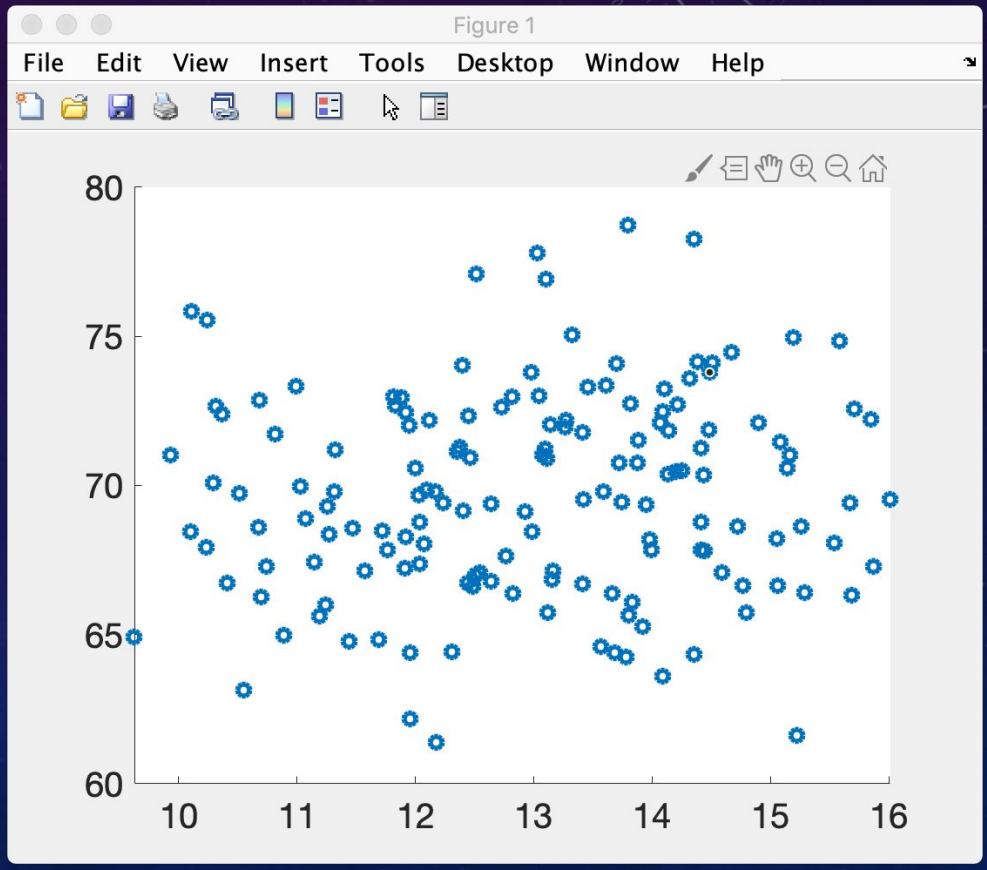
```
scatter(x,y,50);
```

horizontal axis values

vertical axis values

The two arguments are interpreted as coordinates of data pairs to be plotted.

```
scatter(calcium, potassium, 50);
```



Figures

- In MATLAB, **figures** are used to display graphics (e.g. plots, charts, images, etc.) in a separate window.
- You can have several figures at once.
  - Each figure has a unique number (e.g. figure 1, figure 2, …)
  - The **current figure** will be the target of any display operations.
  - Initially, figure 1 is the current figure.
- Use the `figure` function to manage figures.
  - `figure();` — Creates a new figure, which becomes the current figure.
  - `figure(n);` — Sets figure n to be the current figure. (It is created if it doesn't exist already.)

1 min

Minute Exercise: Using Multiple Figures

- Add these lines to your script that analyzes the soil samples:

```
figure();
plot(days, ESP_values);

figure();
scatter(calcium, potassium);

figure(42);
scatter(sodium, ESP_values);
```

- Run your script.
- Now type in the command window:

```
figure(2);
plot(days, magnesium);
```

Solution: Using Multiple Figures

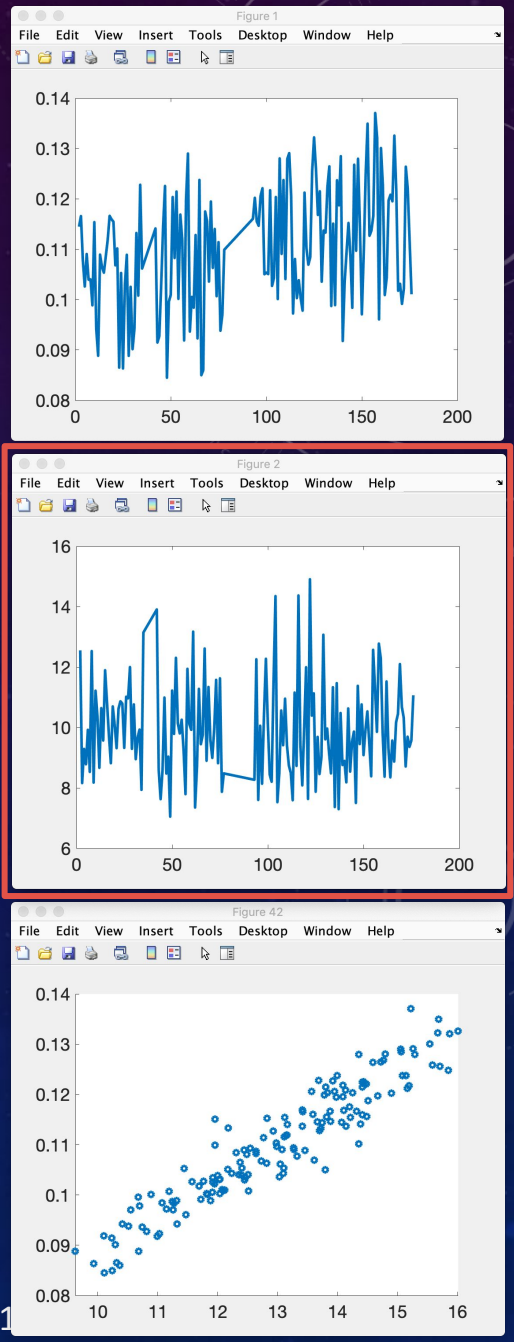
- Add these lines to your script that analyzes the soil samples:

```
figure();
plot(days, ESP_values);

figure();
scatter(calcium, potassium);

figure(42);
scatter(sodium, ESP_values);
```

- Run your script.
- Now type in the command window:

```
figure(2);
plot(days, magnesium);
```

- The original plot in Figure 2 disappeared ☹



Closing Figures

- To close the current figure:

```
close
```

- To close all figures:

```
close all
```

- To close a particular figure:

```
close n
```

Break Time

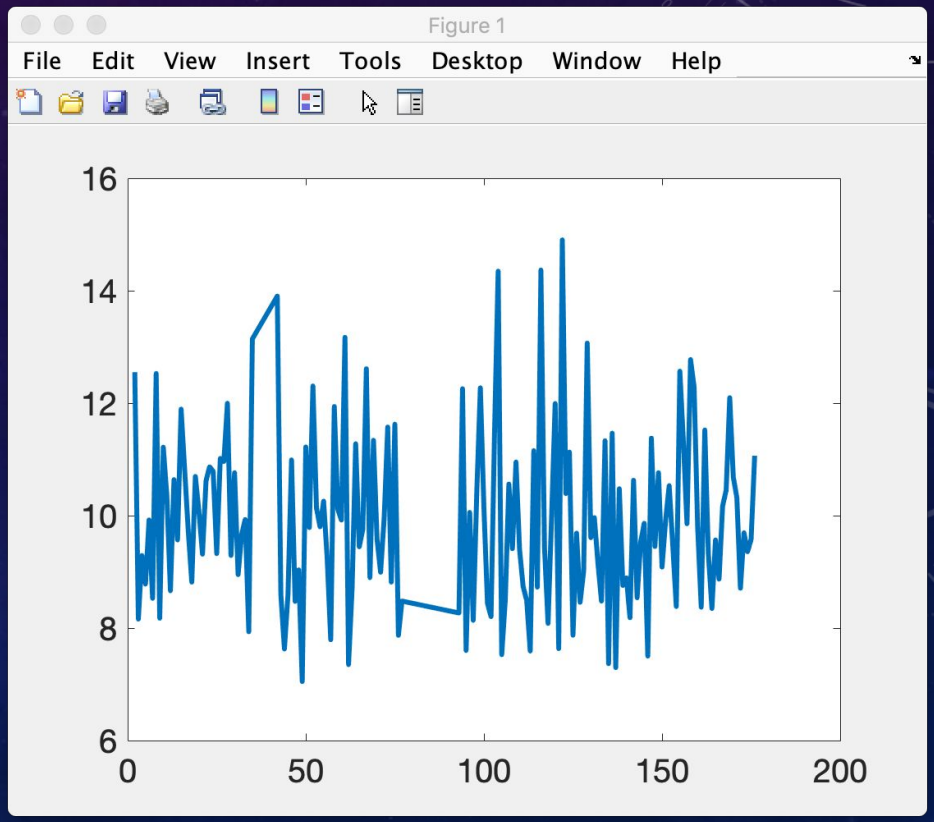
We'll start again in 5 minutes.

Multiple Plots in One Figure

- We can plot different things in different figures, but what if we want to plot different things on the same axes?

```
figure();
plot(days, ESP_values);
plot(days, magnesium);
```

MATLAB's default behavior is to replace the old plot with the new one.



Option 1

Call `plot()` with multiple sets of data

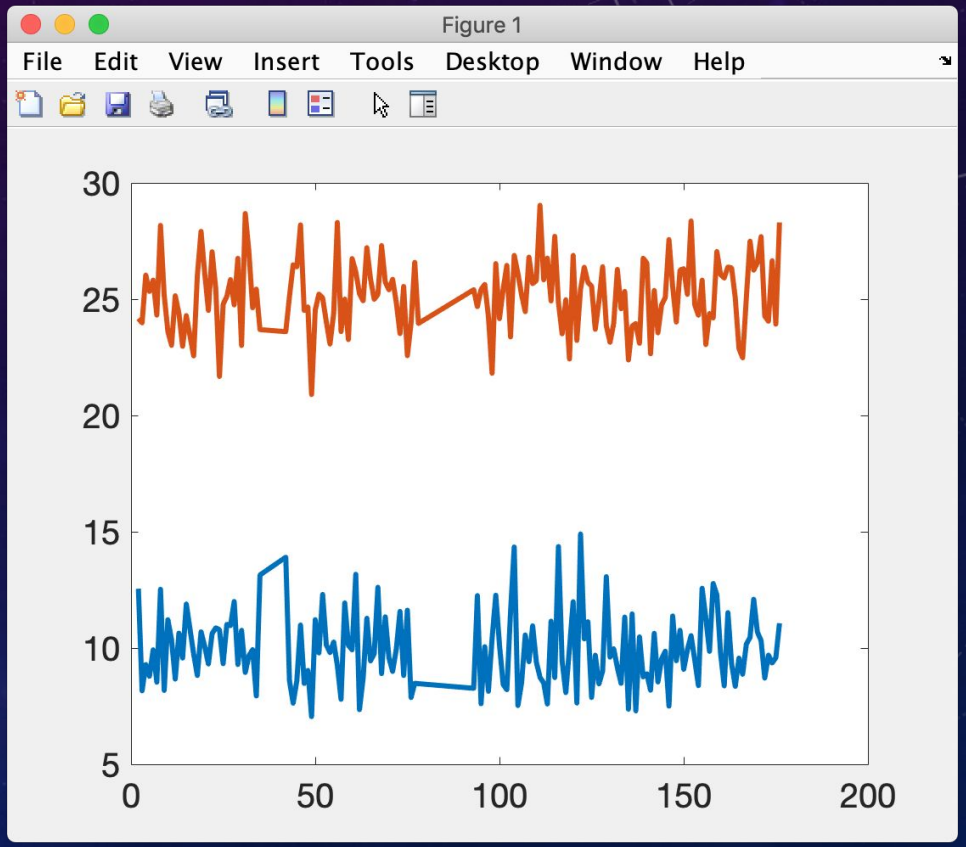
Have more lines to plot? Just keep passing in more data sets!

```
plot(x1, y1, x2, y2, x3, y3,...);
```

first set of data

second set of data

```
plot(days, magnesium, days, calcium);
```



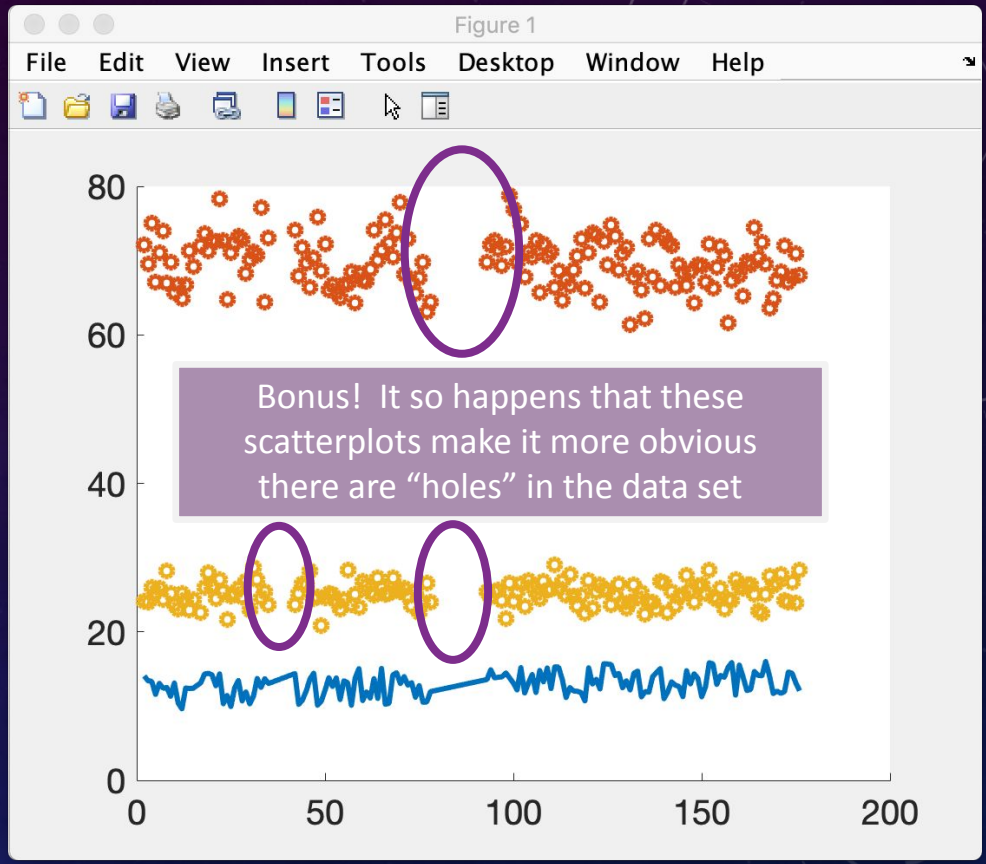
Option 2

The `hold` command tells MATLAB to add new plots to the same set of axes instead of replacing the old plot.

MATLAB will now add new plots without replacing the old ones.

```
figure();
hold on;
plot(days, sodium);
scatter(days, potassium);
scatter(days, calcium);
hold off;
```

After `hold` is turned off, new plots will once again replace old ones.



Creating a Pie Chart

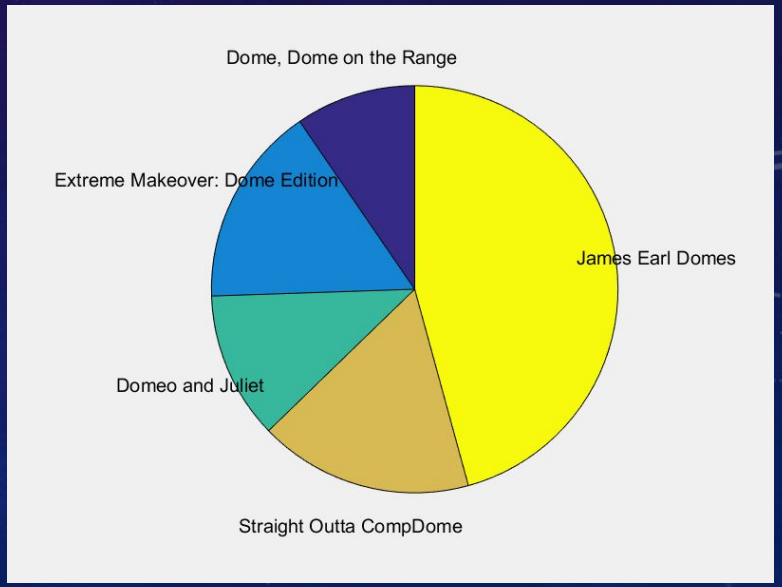
- To show a pie chart, use the `pie` function.

```
votes = [9, 15, 11, 16, 43];
names = {'Dome, Dome on the Range', 'Extreme Makeover:
Dome Edition', … , 'James Earl Domes'};
pie(votes, names);
```

The first input is a vector of either counts or percentages.

The next input contains the labels for each category.

You can change the font size and add lines in the figure window.



- The `pie3` function works similarly but produces a 3D pie chart instead.

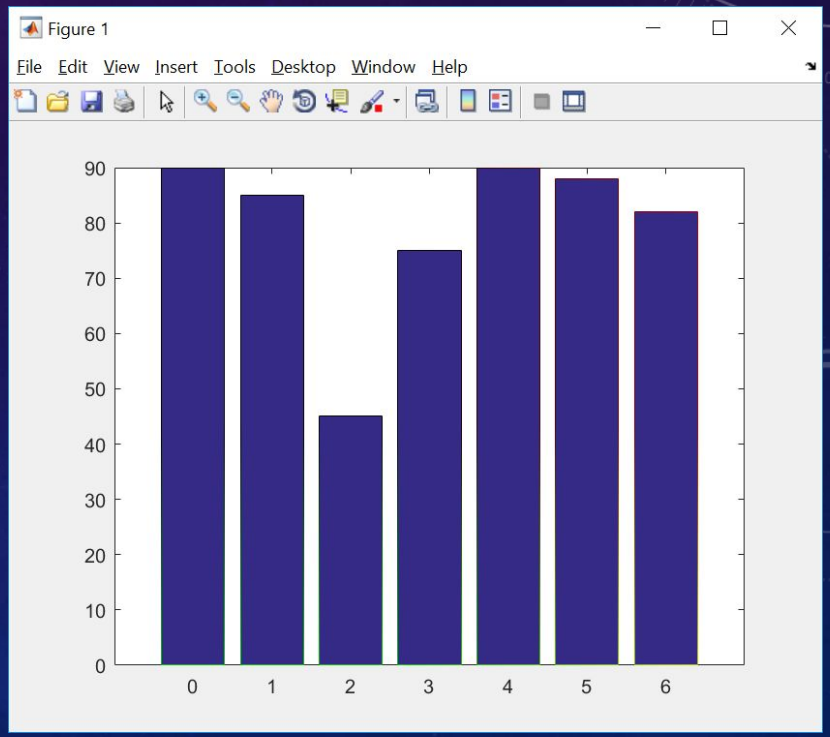
Creating a Bar Chart

- To show a bar chart, use the `bar` function.

```
projects = [0,1,2,3,4,5,6];
scores = [90,85,45,75,90,88,82];
bar(projects, scores);
```

The first input controls default labels and bar positions.

The next input contains the amount for each bar.



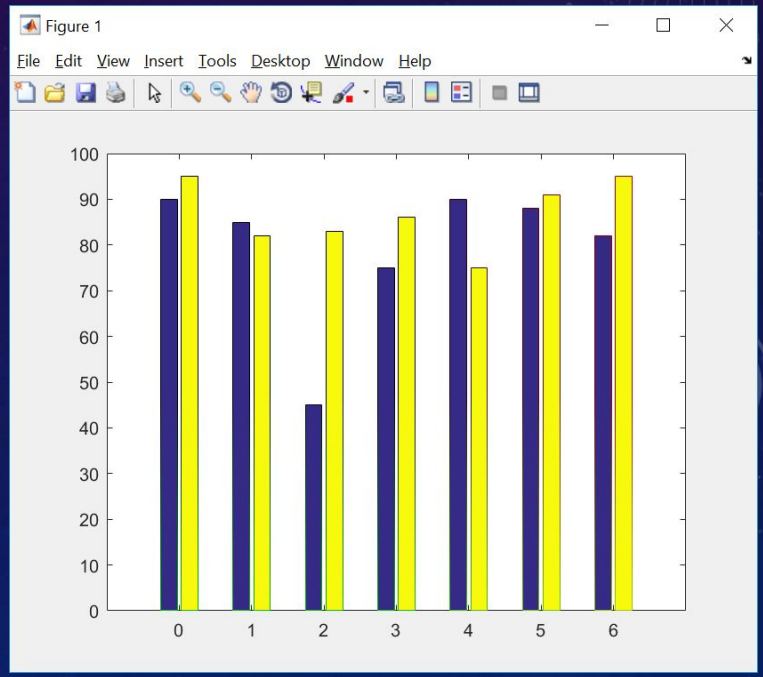
- The `barh` and `bar3` functions works similarly but produce horizontal and 3D charts, respectively.

Creating a Bar Chart

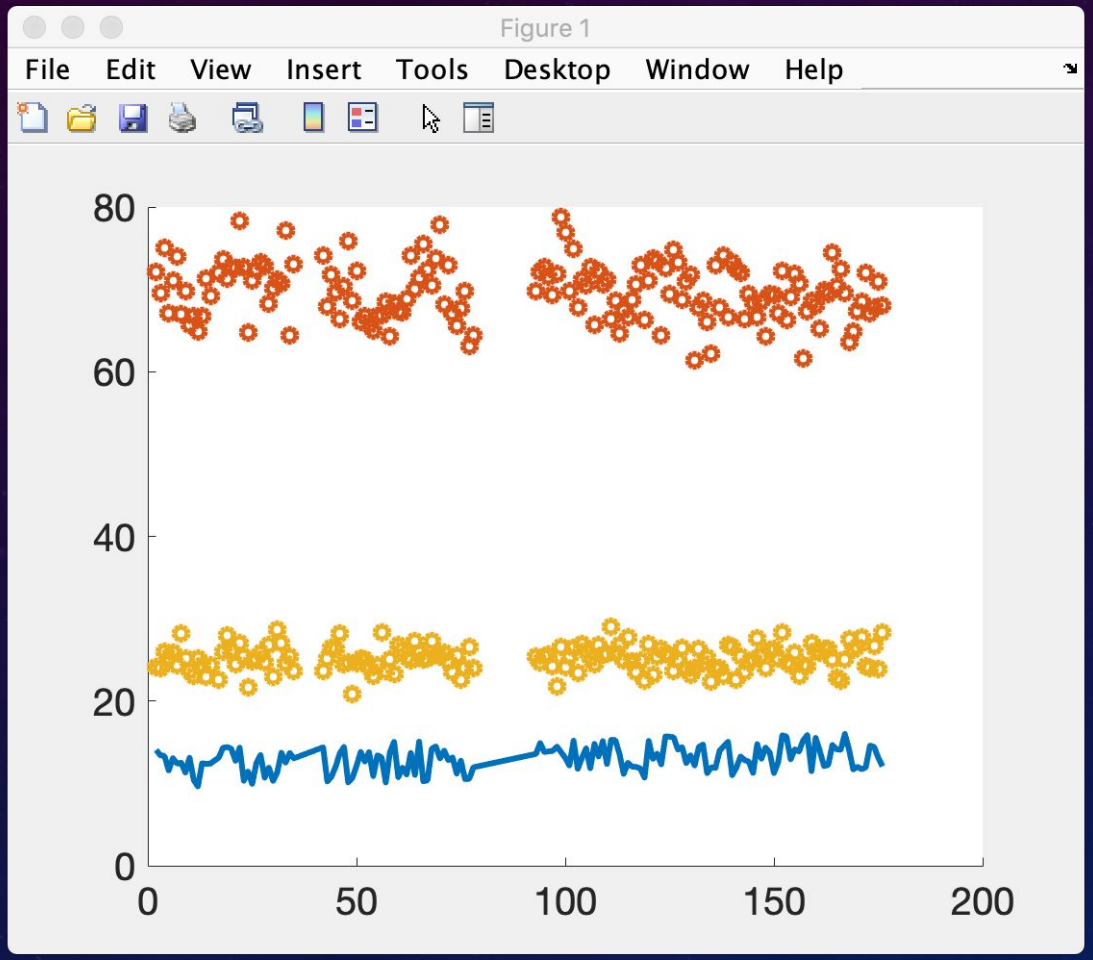
- To show grouped bars, use a matrix where each row corresponds to a single group of bars.

```
projects = [0,1,2,3,4,5,6];
autograder = [90;85;45;75;90;88;82];
style = [95;82;83;86;75;91;95];
bar(projects, [autograder,style]);
```

Combine so that each row has one score for both autograder and style.



What's wrong with this plot?



How is anyone going to know what this plot is supposed to be?

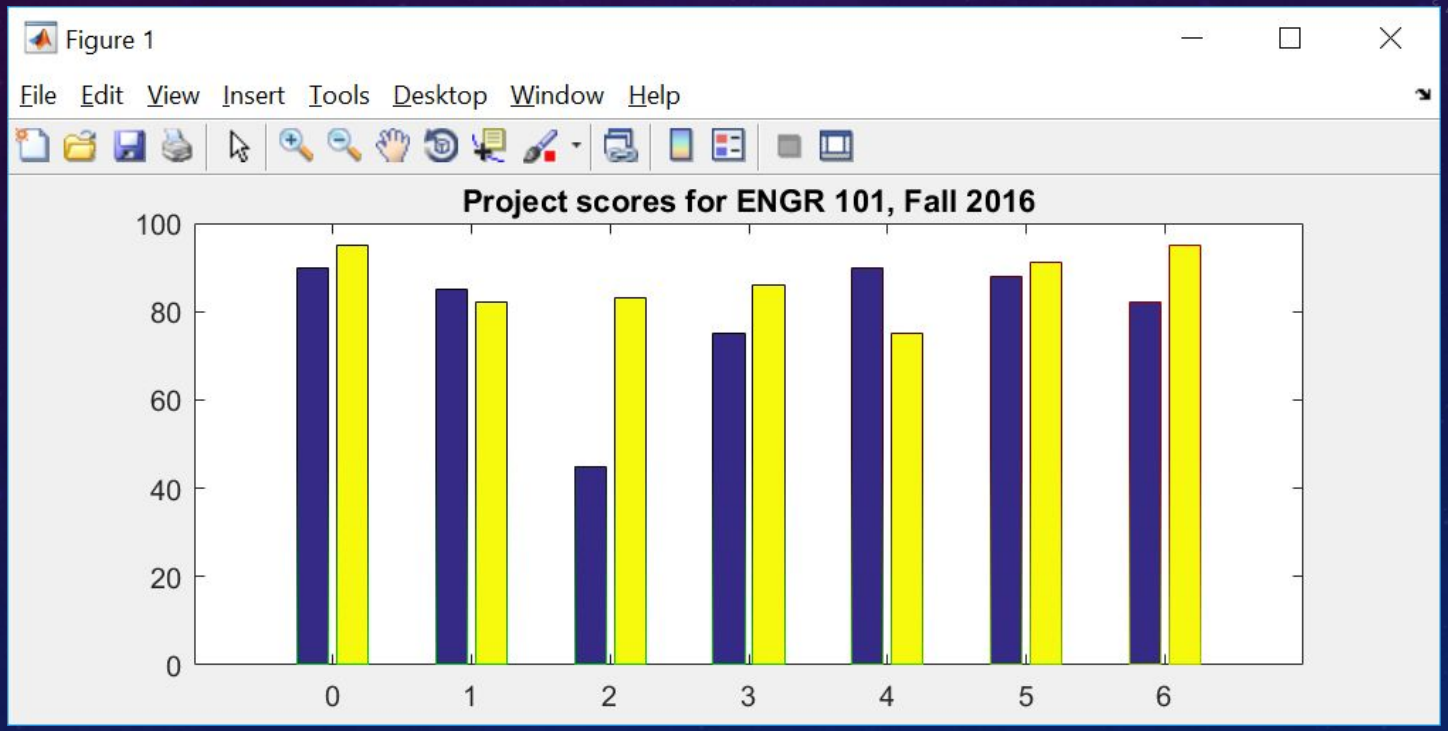
Customizing Plots

- We'll look at a few methods for customizing the appearance of plots in MATLAB.
- This lecture only covers the basics – see the MATLAB documentation linked in the footnotes for all the details!

Adding a Title

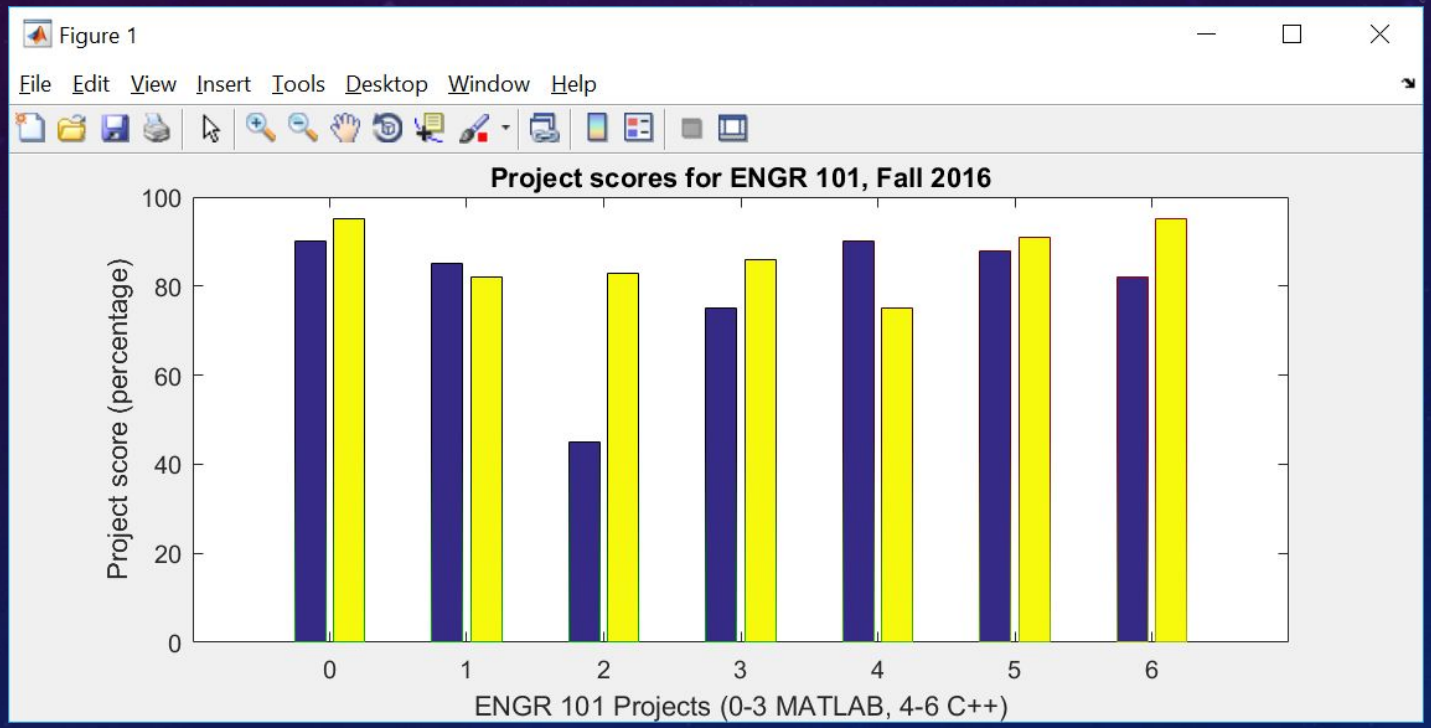
- The `title` function adds a title to the current figure.

```
title('Project scores for ENGR 101, Fall 2016');
```



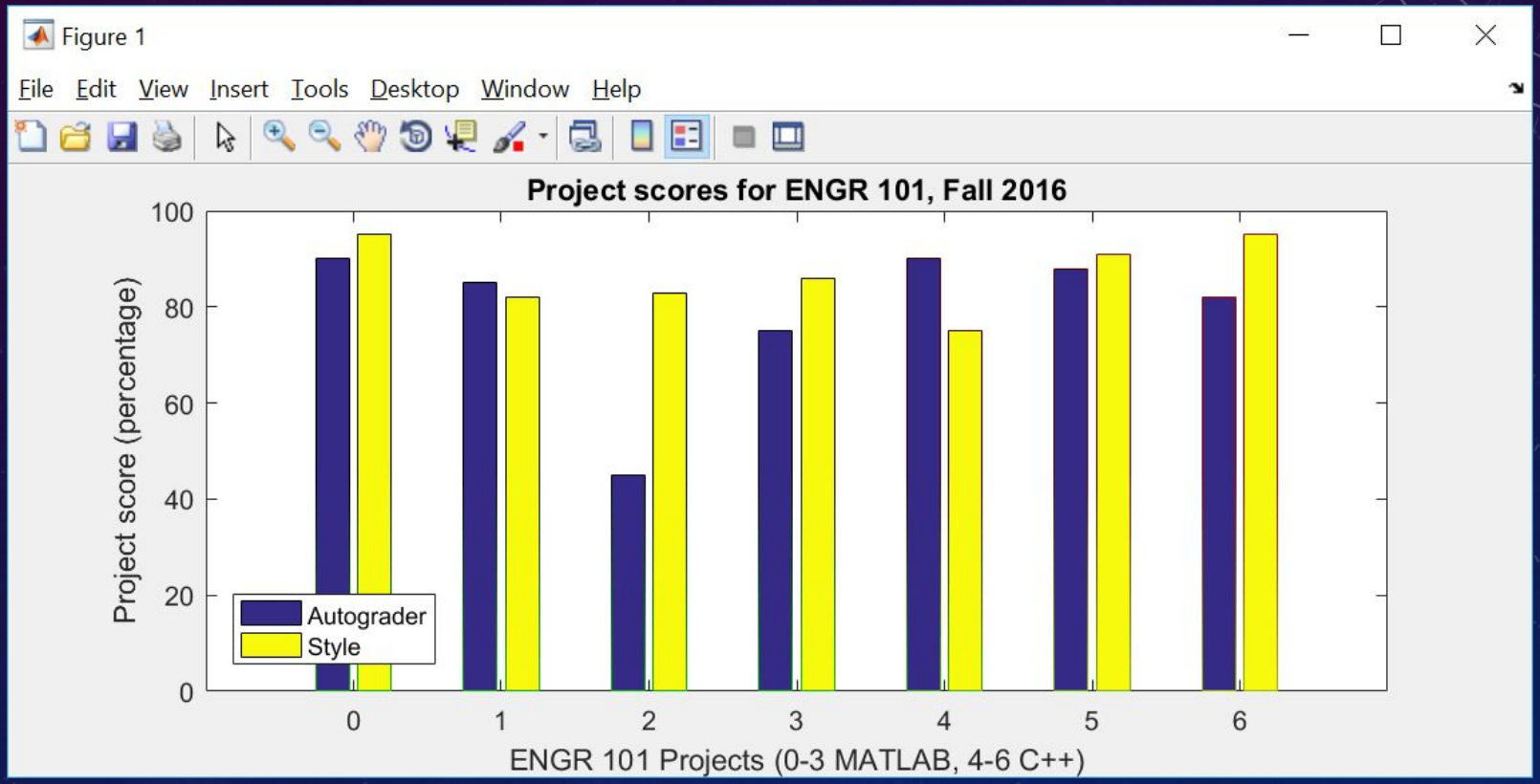
Adding Axis Labels

```
xlabel('ENGR 101 Projects (0-3 MATLAB, 4-6 C++)');
ylabel('Project score (percentage)');
```



Adding a Legend

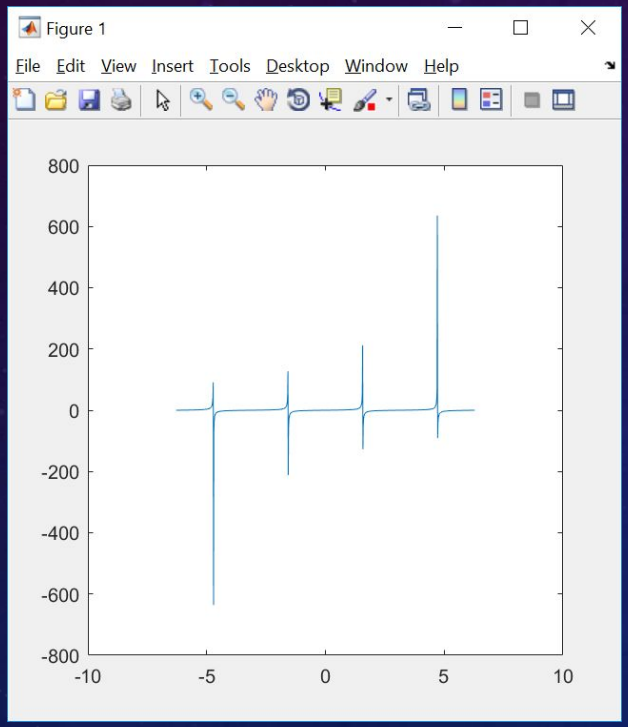
```
legend('Autograder', 'Style', 'Location', 'Southwest');
```

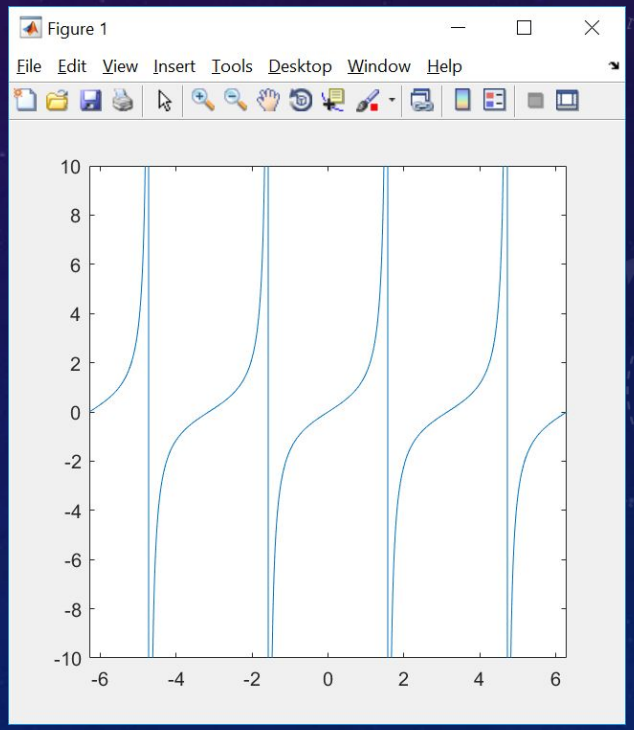


Customizing Axes

- MATLAB tries to guess at appropriate axes ranges, but sometimes it just takes a human touch.

```
xlim([-2.*pi, 2.*pi]);
ylim([-10, 10]);
```





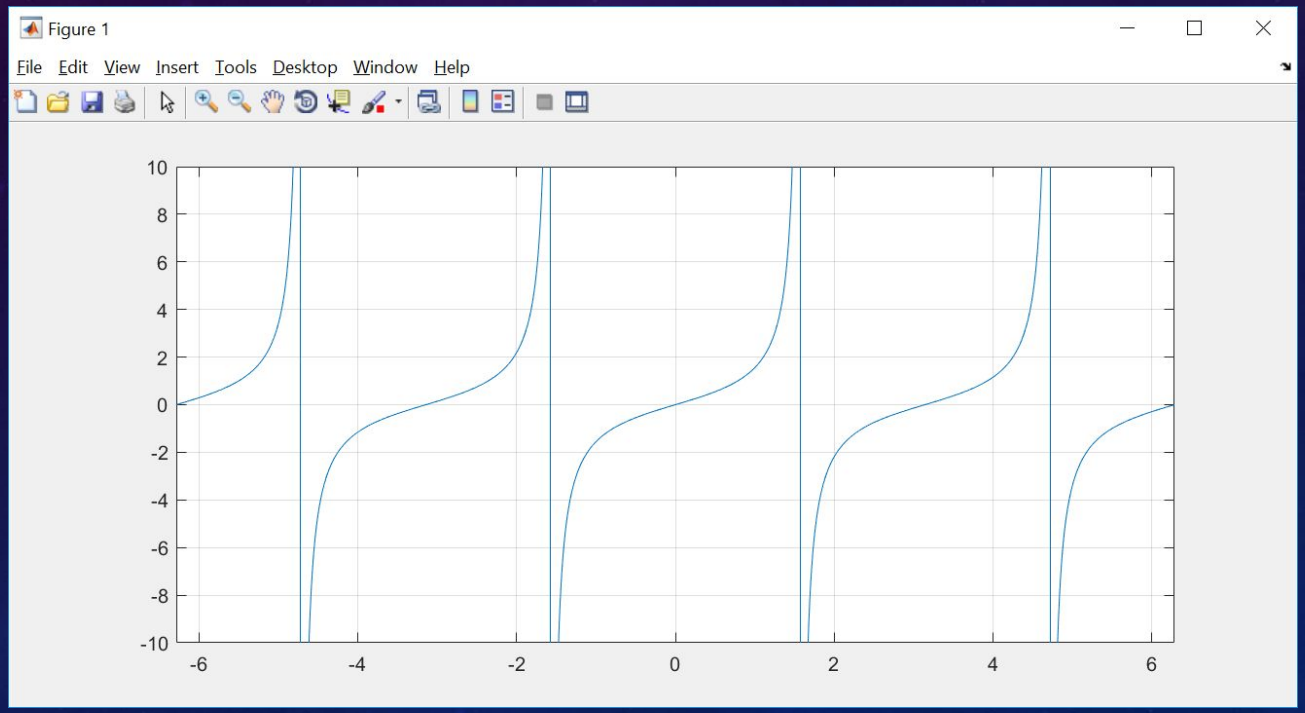
Gridlines

- To turn on gridlines:

```
grid on
```

- To turn off gridlines:

```
grid off
```



Shortcuts for Line Plots

```
x = 0:0.1:10;
plot(x, sin(x), '--sg', x, cos(x), ':or');
```

o
circle
marker

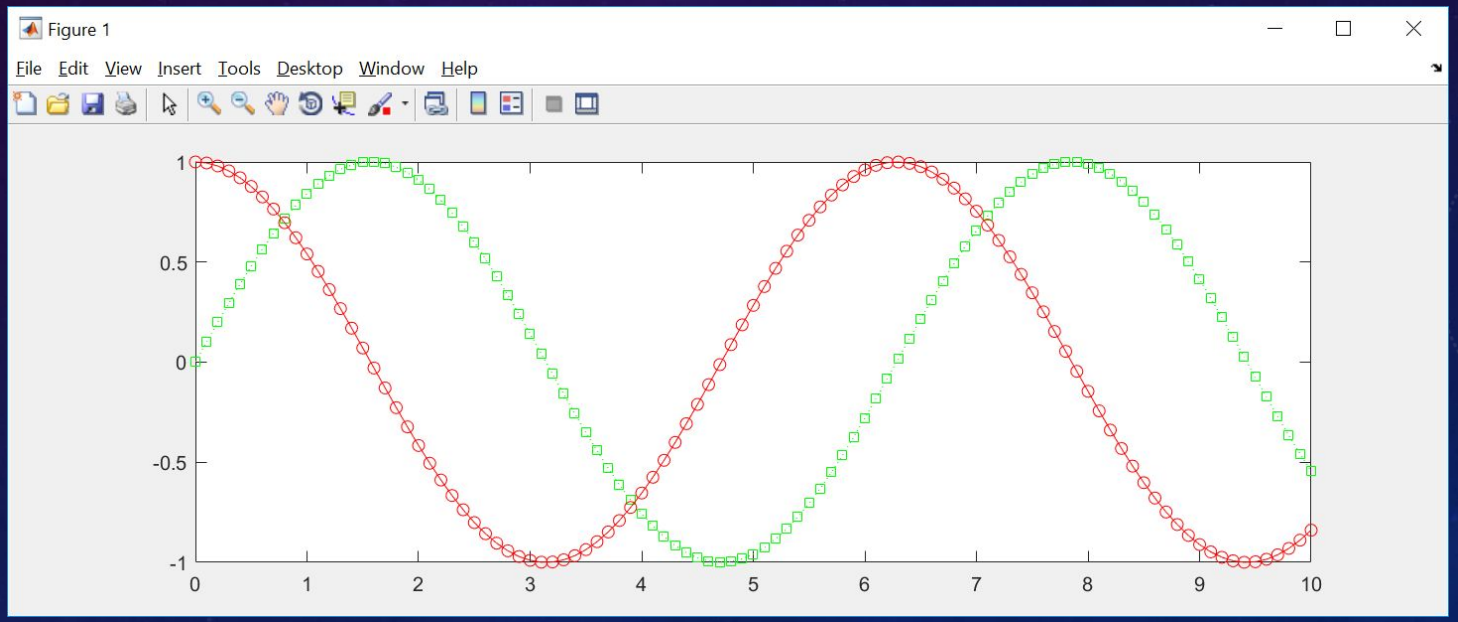
r
red color

--
dashed line

s
square
marker

g
green color

:
dotted line



Plotting and Scripts/Functions

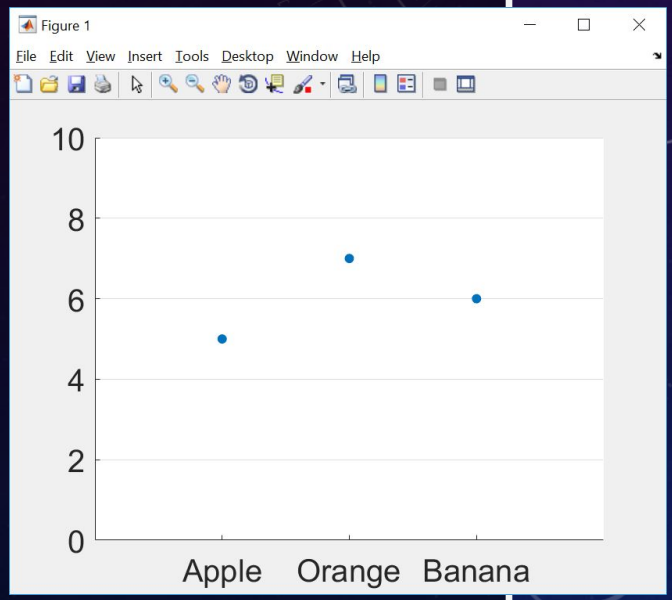
- Scripts and/or functions can be helpful as a way to collect together all the individual commands to customize a plot.
- You can change just a bit and then rerun the whole thing.

```
function [ ] = plot101Grades(projects, autograder, style)

   figure();
   bar(projects, [autograder,style]);
   title('Project scores for ENGR 101, Fall 2016');
   xlabel('ENGR 101 Projects (0-3 MATLAB, 4-6 C++)');
   ylabel('Project score (percentage)');
   legend('Autograder', 'Style', 'Location', 'Southwest');

end
```

gca

- You can use gca ("get current axes") to modify properties of the axes for the current figure.

```
% create a simple scatterplot
scatter([1,2,3], [5,7,6], 'filled');

% get current axes in the variable ax
ax = gca;

% modify properties through ax
ax.FontSize = 20;
ax.YLim = [0,10];
ax.XLim = [0,4];
ax.XTick = [1,2,3];
ax.XTickLabel = {'Apple', 'Orange', 'Banana'};
ax.YTick = [0:2:10];
ax.YGrid = 'on';
```



https://www.mathworks.com/help/matlab/ref/axes-properties.html

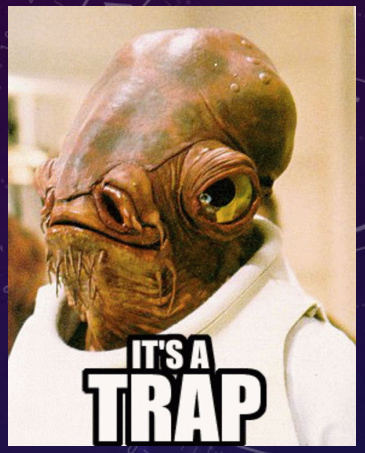
The set Function

- You can also use the set function instead of the dot notation.

```
% create a simple scatterplot
scatter([1,2,3], [5,7,6], 'filled');

% get current axes in the variable ax
ax = gca;

% set the font size using dot notation
ax.FontSize = 20;

% set the font size using the set function
set(ax,'FontSize',20);
```

- In rare cases (or older versions of MATLAB), only the approach using the set function will work.

Caution!



- It's tempting to try something like this:

```
% create a simple scatterplot
scatter([1,2,3], [5,7,6], 'filled');

% modify a property directly through gca
gca.FontSize = 20;
```

It doesn't work![1]

- You always need to "store" the current axes in a variable first:

```
% create a simple scatterplot
scatter([1,2,3], [5,7,6], 'filled');

% get current axes in the variable ax
ax = gca;

% modify properties through ax
ax.FontSize = 20;
```

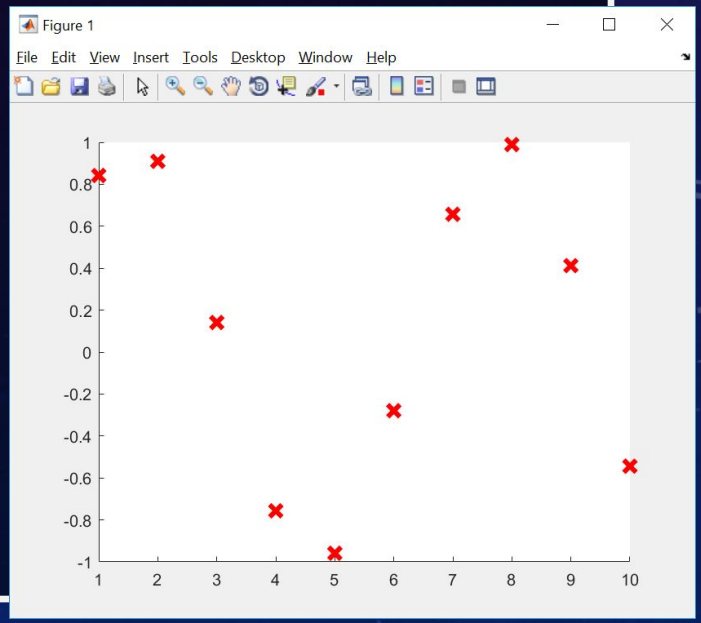
Do it this way.

[1] It actually creates a variable called `gca` that interferes with the regular `gca`. You'll also need to run "`clear gca;`" before it will work again.

Graphics Object Properties

- Plotting functions return graphics objects that can be used to customize the appearance of the plot.

```
% create a scatterplot
% store the returned graphics object in s
x = 1:1:10;
s = scatter(x, sin(x), 100);

% modify properties through s
s.Marker = 'x';
s.LineWidth = 3;
s.MarkerEdgeColor = 'red';
```

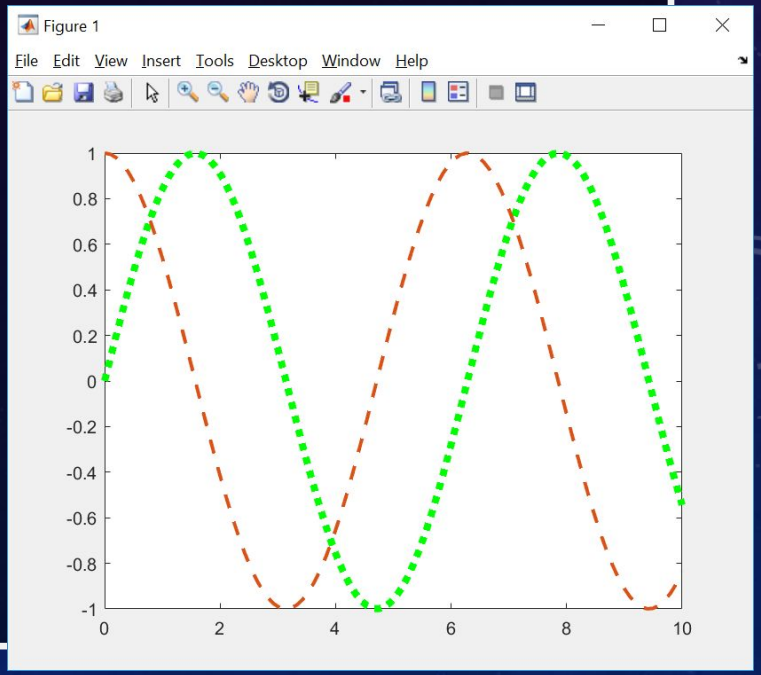


https://www.mathworks.com/help/matlab/graphics-object-properties.html

Graphics Object Properties

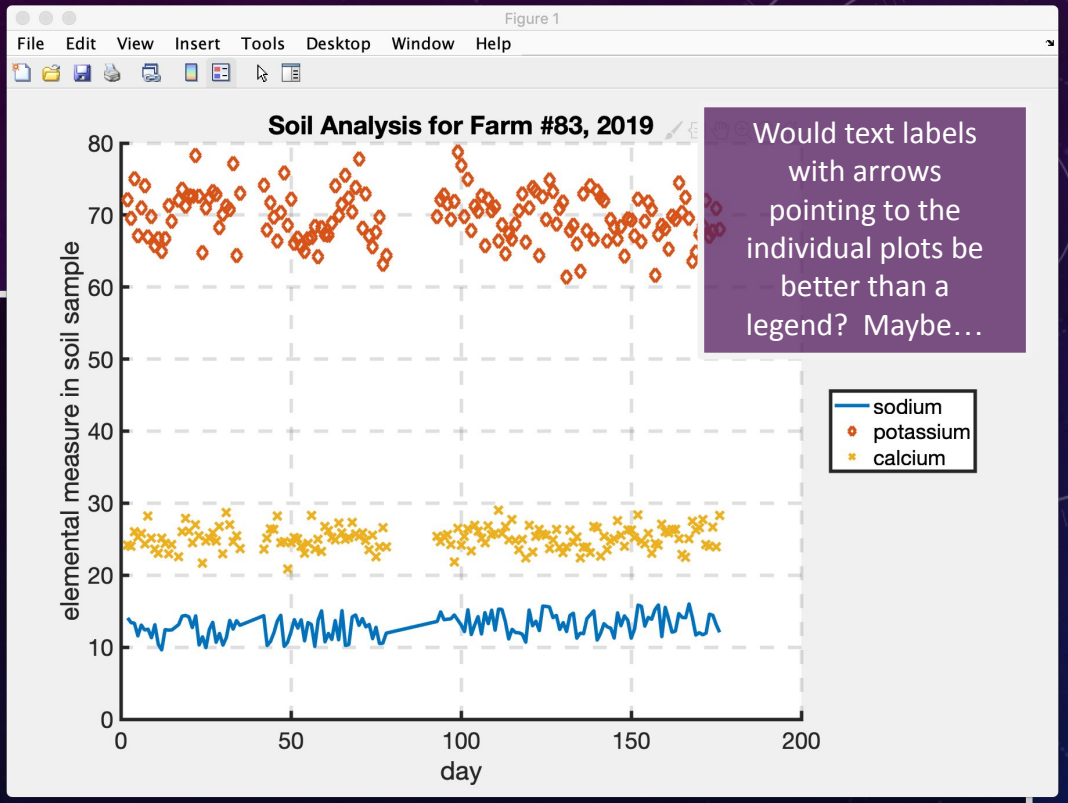
- If you plot more than one thing at a time, you'll get a vector of graphics objects. Index into it to modify properties.

```
% plot multiple functions
% store the returned graphics objects in p
x = linspace(0,10,100);
p = plot(x, sin(x), x, cos(x));

% modify properties through p
% index to select which plot
p(1).LineStyle = ':';
p(1).Color = 'green';
p(1).LineWidth = 4;

p(2).LineStyle = '--';
P(2).Color = 'red';
p(2).LineWidth = 2;
```



https://www.mathworks.com/help/matlab/graphics-object-properties.html

A Better Plot!




```
figure();
hold on;
p1 = plot(days, sodium);
p1.LineWidth = 3;
s1 = scatter(days, potassium, 80);
s1.LineWidth = 3;
s1.Marker = 'd';
s2 = scatter(days, calcium, 80);
s2.LineWidth = 3;
s2.Marker = 'x';
hold off;
grid on;
title('Soil Analysis for Farm #83, 2019');
legend('sodium','potassium','calcium','Location', 'Eastoutside');
ax = gca;
ax.FontSize = 20;
ax.LineWidth = 3;
ax.GridLineStyle = '--';
ax.XLabel.String = 'day';
ax.YLabel.String = 'elemental measure in soil sample';
```

This looks like so much code, right?! But now we have complete control over how MATLAB plots our data, and we can re-use this code any time we want!

Refer to the MATLAB help documentation for more examples of things you can control when plotting!

Example: Reading Rainfall Data

- Download the files `rainfall.csv` and `AnalyzeRainfall.m` from the Google Drive.

```
% read the data, skipping 1 row
data = csvread('rainfall.csv',1,0);

% extract individual columns
days = data(:,1);
dailyRain = data(:,2);
totalRain = data(:,3);

% Now do something with the data...
```

```
day,rainfall,cumulative
1,0,0
2,0,0
3,0,0
4,0,0
5,0,0
6,0,0
7,0,0
8,0.14,0.14
9,0.13,0.27
10,0.18,0.45
11,0,0.45
12,0.04,0.49
13,0.01,0.5
14,0,0.5
```

`rainfall.csv`

http://www3.a2gov.org/RainGauges/
`rainfall.csv` contains data for the first 300 or so days of 2016.

3 min

Exercise: Rainfall Data

- In the `AnalyzeRainfall.m` script...

- Use `csvread` to load the dataset, skipping the header row.
- Extract variables for each column of data:
  - `days` – Numbered days 1-276 (October 2nd is day 276)
  - `dailyRain` – Amounts of rainfall on each day.
  - `totalRain` – A cumulative sum of rainfall up to that day.

- Create charts that show both the daily rainfall amounts and the cumulative total as a function of the day.
  - Think about which kind of plot is appropriate to use for each case!
  - Practice adding titles, axes labels, legends, etc.

Plotting in General

- So far, we've only scratched the surface of what you can do with plots in MATLAB. There's so many more options!
- Here's the truth:

  Nobody memorizes all the different kinds of plots and the ways you can customize them.

- Refer to online documentation for general guidance.
- Search online if there's something specific you're looking for.
  - https://www.mathworks.com/help/matlab/creating_plots/types-of-matlab-plots.html

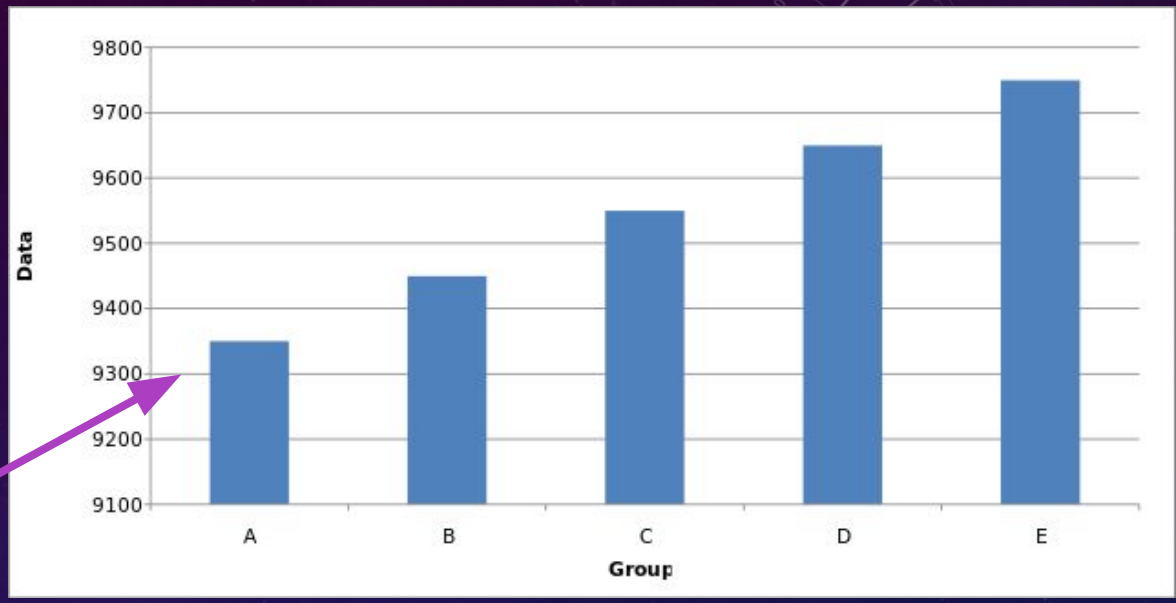
Your Graphs Will Out-Live You

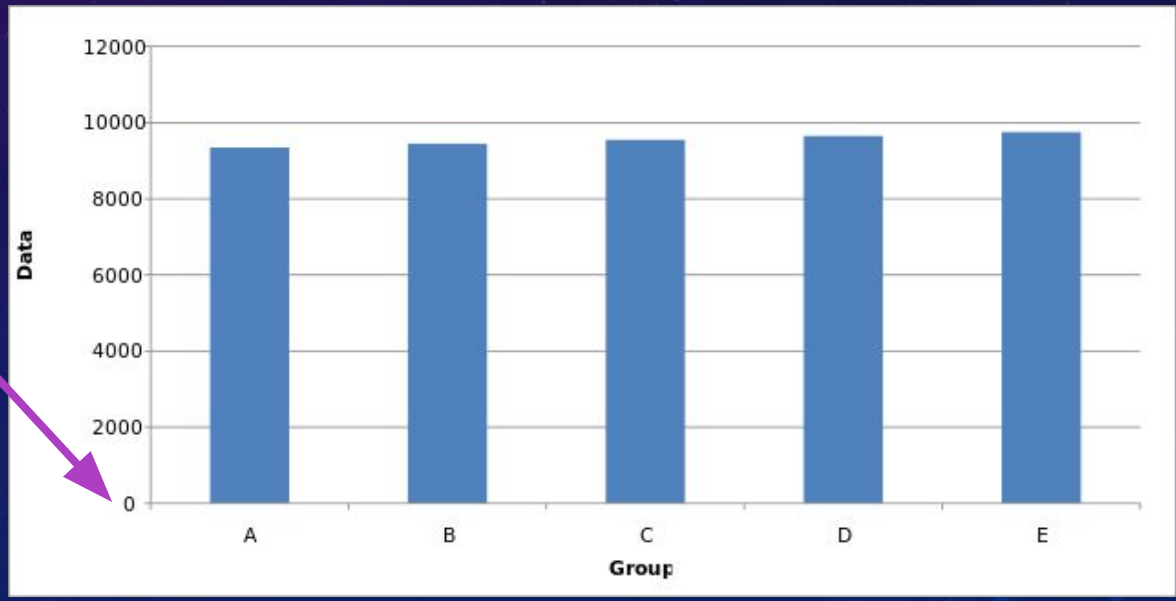
- The graphs and figures you make will go on to have a life of their own
- As a responsible engineer, you need to make clear graphs that can “stand on their own” – i.e. it is obvious what they are trying to convey
- Graphs and figures can show relationships between data that are difficult, if not impossible, to explain with words
- Graphs and figures can help you (and your audience) grasp what an equation actually *means*

Don't Be Misleading in Your Plots

Wow! Look at the differences between the groups!

Oh wait… When the y-axis starts at zero, there's almost no difference.





for more information:
https://en.wikipedia.org/wiki/Misleading_graph

Do Maximize your "Signal-Noise Ratio"

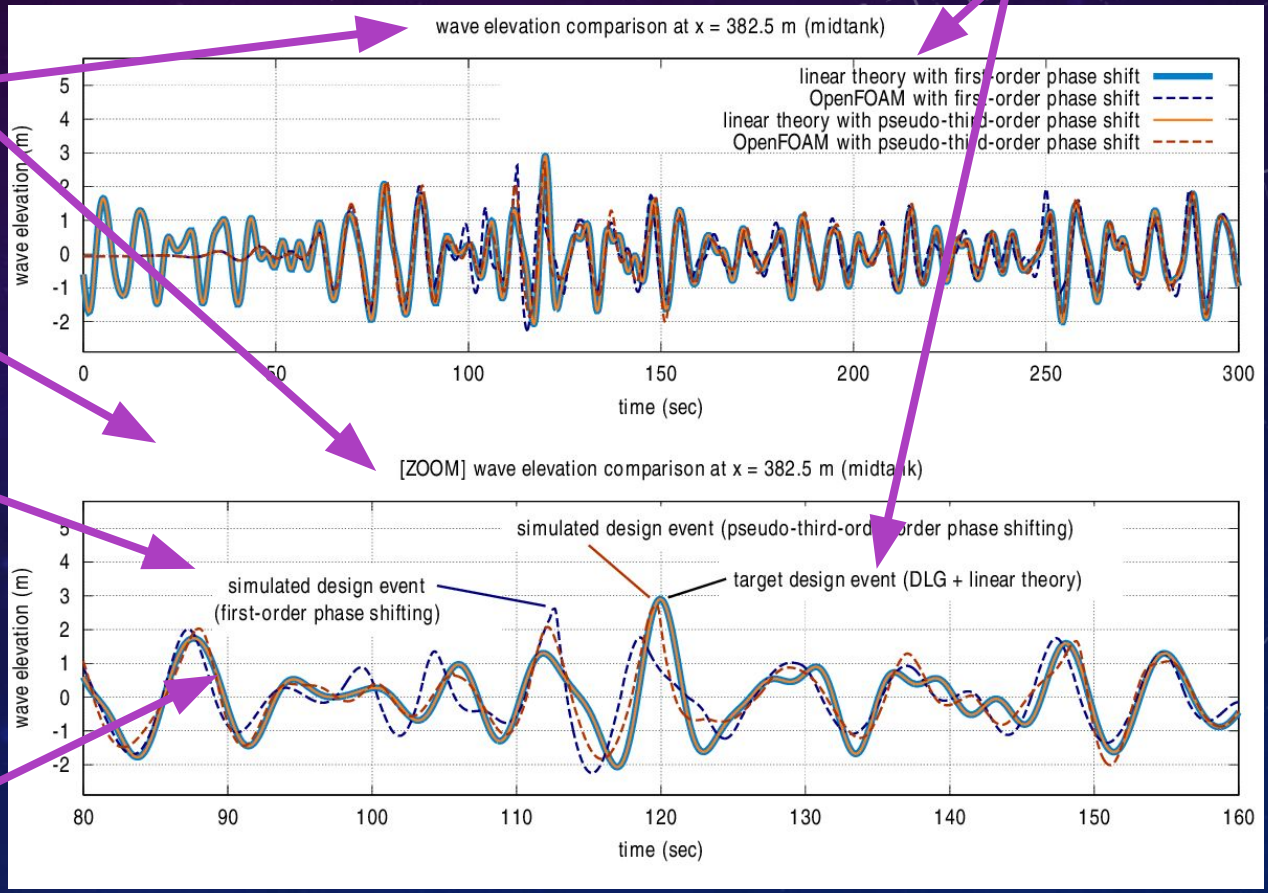
use legends vs. labels appropriately

if you HAVE to have a title, it should describe what your audience should learn from the graph

plot on white background

include horizontal and vertical grid lines (unless you have a REALLY good reason not to)

choose high contrast colors (watch out for color blindness!)



More inspiration: reddit.com/r/dataisbeautiful