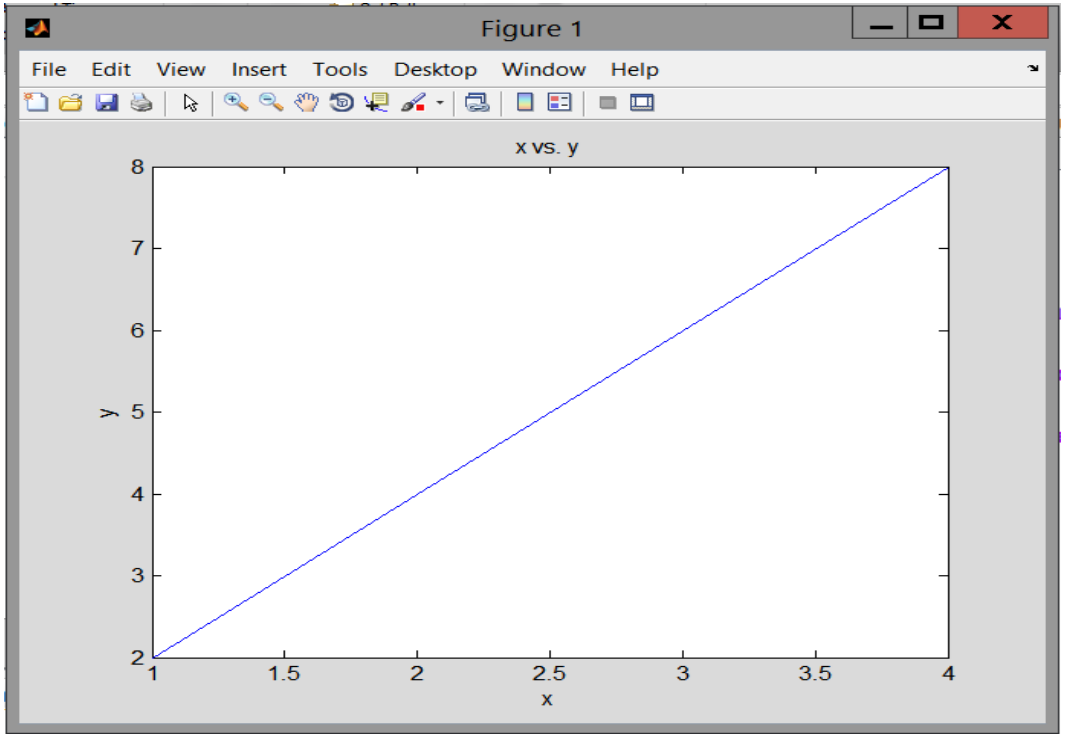
2.1 Basic Plotting

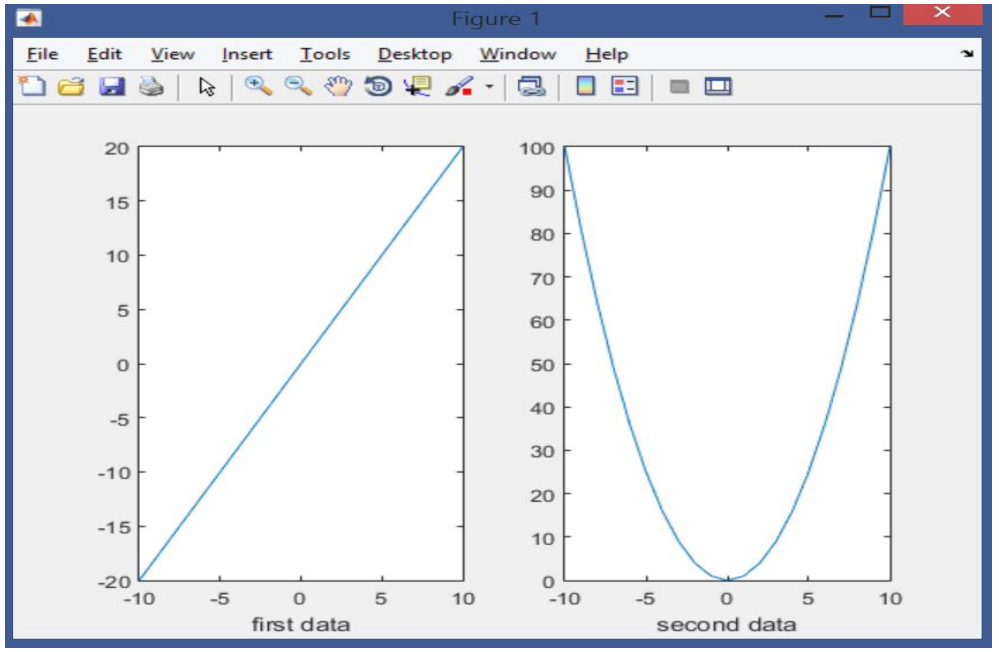
2D Line Graphs

```
>> x = 1:4;
>> y = 2*x;
>> plot(x,y)
>> xlabel('x')
>> ylabel('y')
>> title('x vs. y')
```



Subplots

```
>> subplot(1,2,1)
>> plot(x,y)
>> xlabel('first data')
>> subplot(1,2,2)
>> z = x.^2;
>> plot(x,z)
>> xlabel('second data')
```



Subplots

Marker type & colour

```
>> plot(x,y,'or','MarkerFaceColor','r', 'MarkerSize',8)
>> hold on
>> plot(x,y,'k')
```

Plot next figure on top of last one

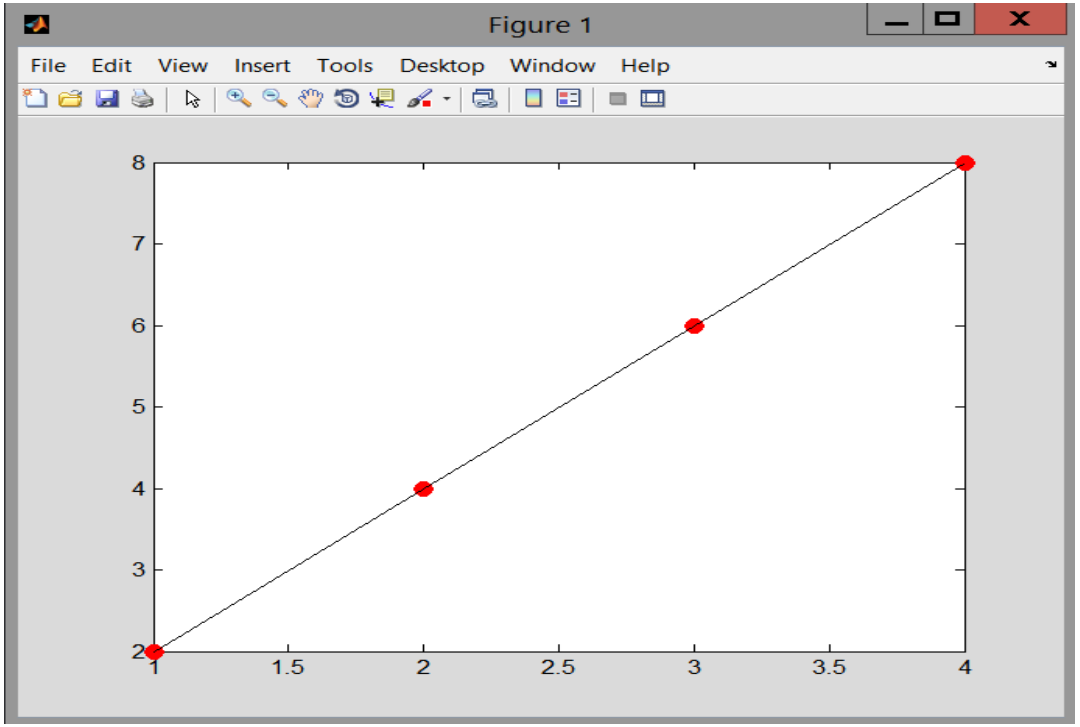
To replace figure use

```
>> hold off
```

- Properties come as **name-value** pairs.
- Read

```
>> doc plot
```

for all pairs.

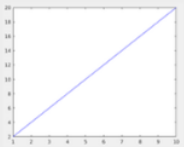


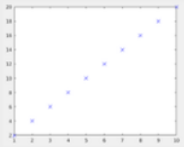
Colors		Symbols		Line Types	
Blue	b	Point	.	Solid	-
Green	g	Circle	o	Dotted	:
Red	r	X-mark	x	Dashdot	-.
Cyan	c	Plus	+	Dashed	--
Magenta	m	Star	*		
Yellow	y	Square	s		
Black	k	Diamond	d		
White	w	Triangle(down)	v		
		Triangle(up)	^		
		Triangle(left)	<		
		Triangle(right)	>		
		Pentagram	p		
		Hexagram	h		

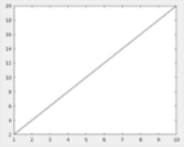
- Available markers, colours & line types.

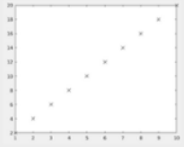
Or go to www.pollev.com/jsands601

Output of >> plot(1:10,2*[1:10],'xb','MarkerSize',8)





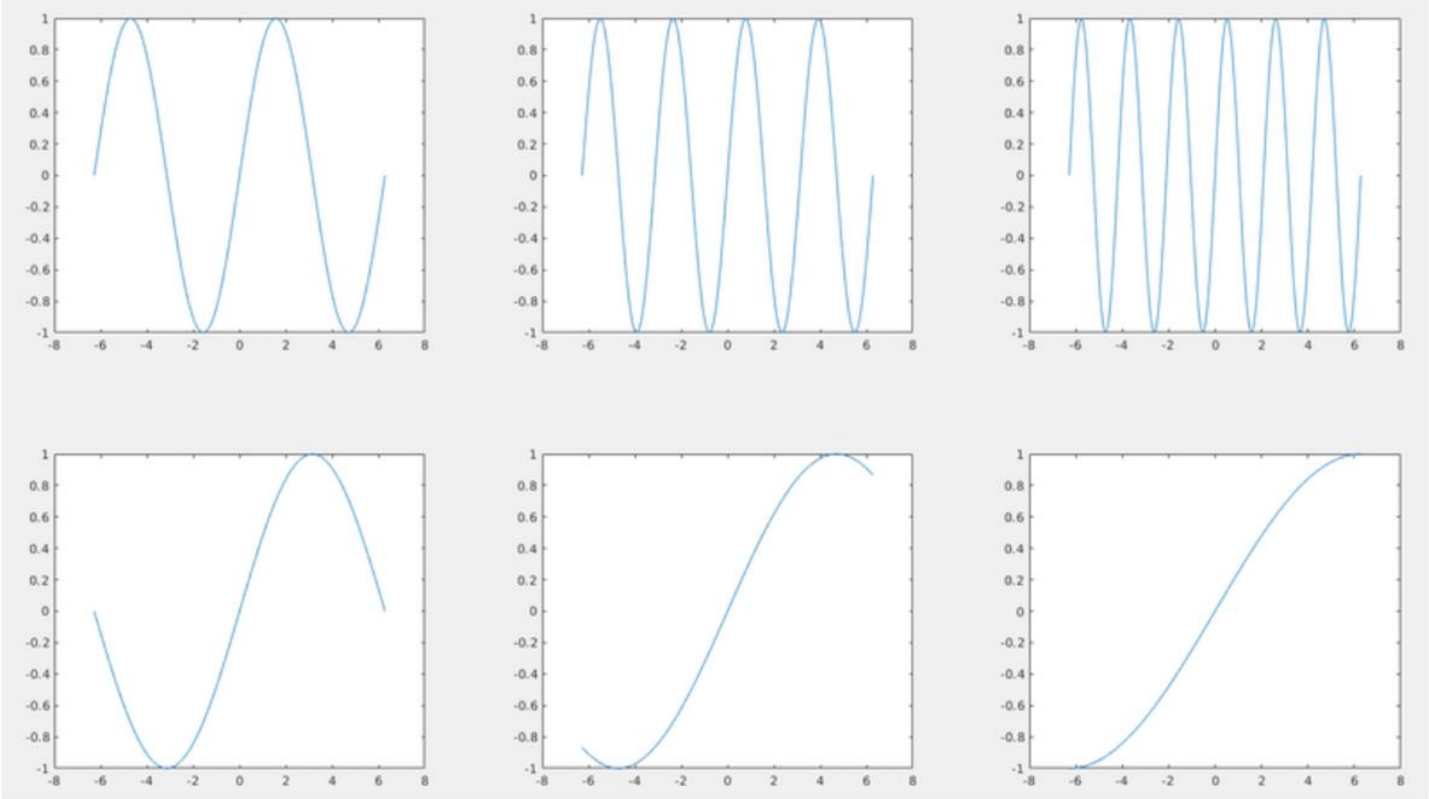




To 0

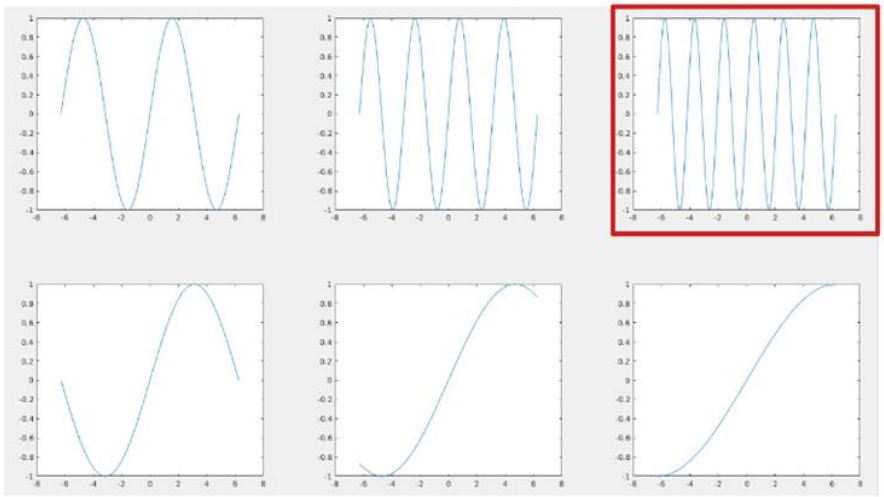
Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Which graph is referenced by subplot(2,3,5)?



Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Which command gives the plot in the red box?



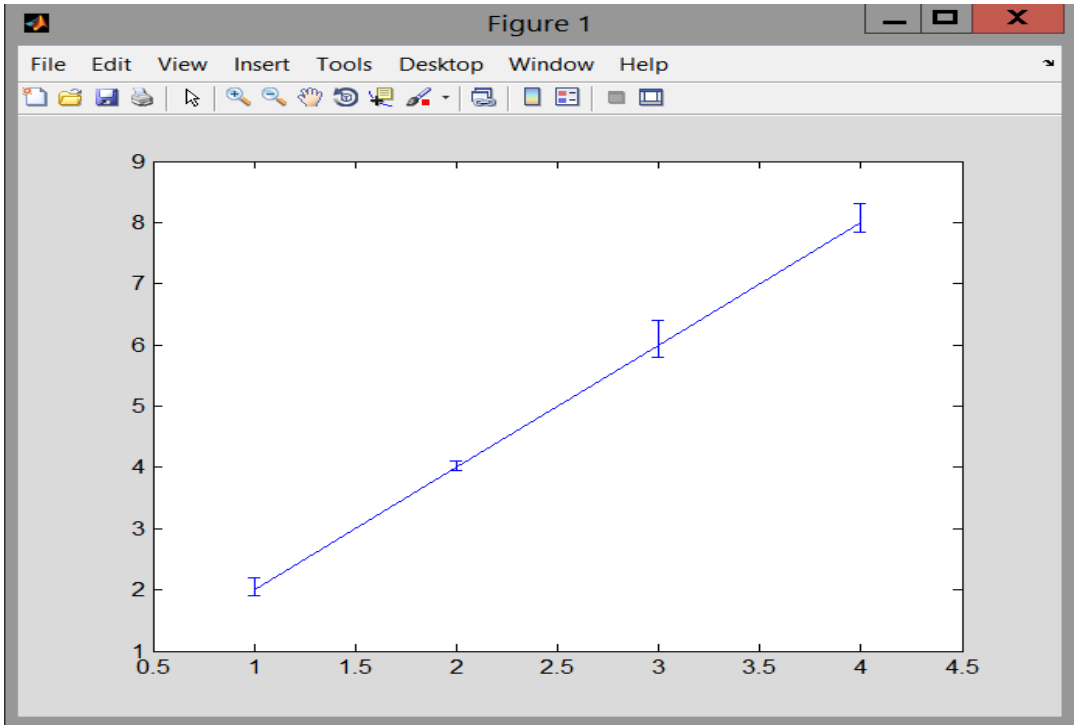
- subplot(2,3,3)
- subplot(3,2,3)
- subplot(2,3,1)
- subplot(3,2,1)

To 0

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

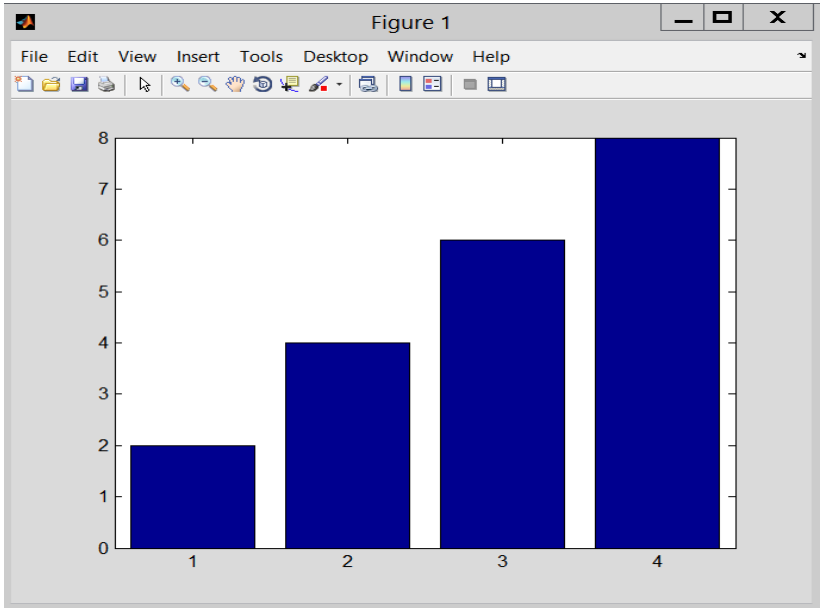
Error Bars

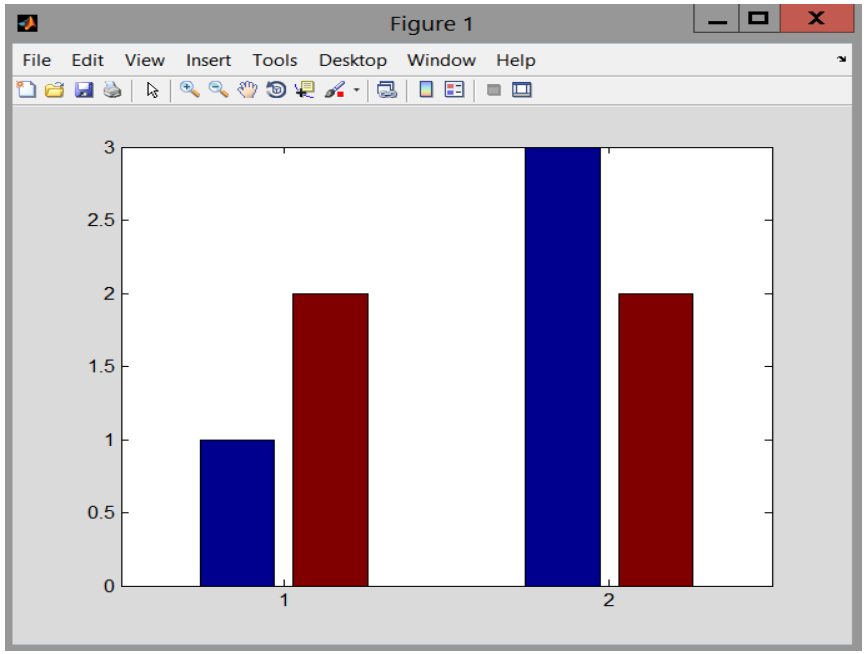
```
>> eUp = [0.2,0.1,0.4,0.3];
>> eDown = 0.5*eUp;
>> errorbar(x,y,eDown,eUp)
```



Bar Plots

```
>> bar(x,y)
```

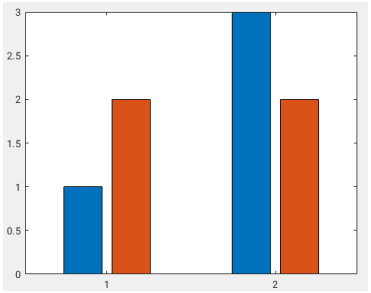





```
>> z = [1,2;3,2];
>> bar(z)
```

Getting Graphics Objects Handles

- Save the plot data as a variable to access its properties later.
- Use the get current axes command "**gca**" to edit the plot area.




```
>> z = [1,2;3,2];
>> h = bar(z);
>> ax = gca
```

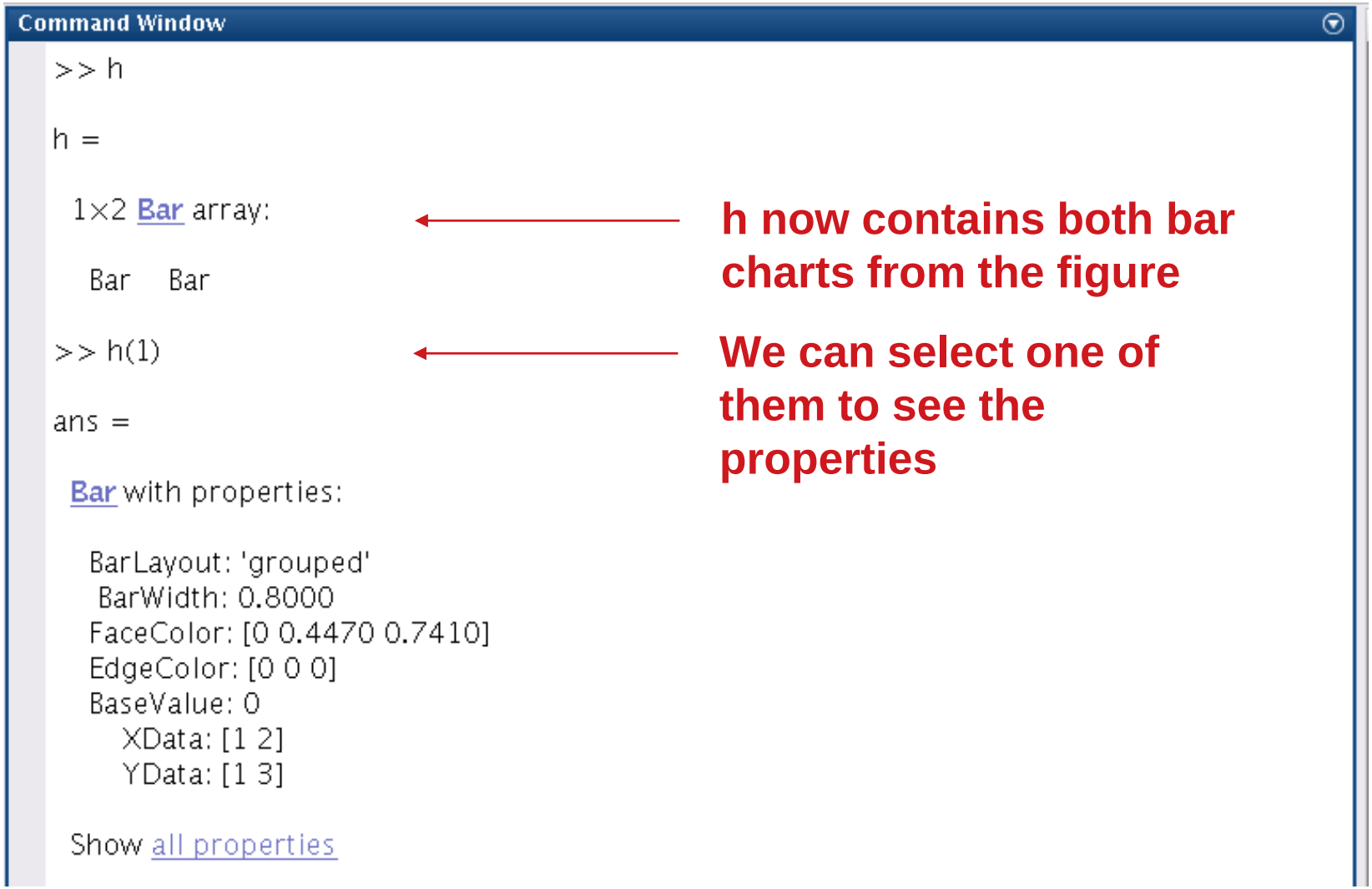
Method 2

Command Window

```
>> z = [1,2;3,2];
>> h = bar(z);
>> ax = gca

ax =

  Axes with properties:

            XLim: [0.5143 2.4857]
            YLim: [0 3]
          XScale: 'linear'
          YScale: 'linear'
   GridLineStyle: '-'
        Position: [0.1300 0.1100 0.7750 0.8150]
           Units: 'normalized'

  Show all properties
```

Workspace

Name	Value
ax	1x1 Axes
h	1x2 Bar
z	[1,2;3,2]



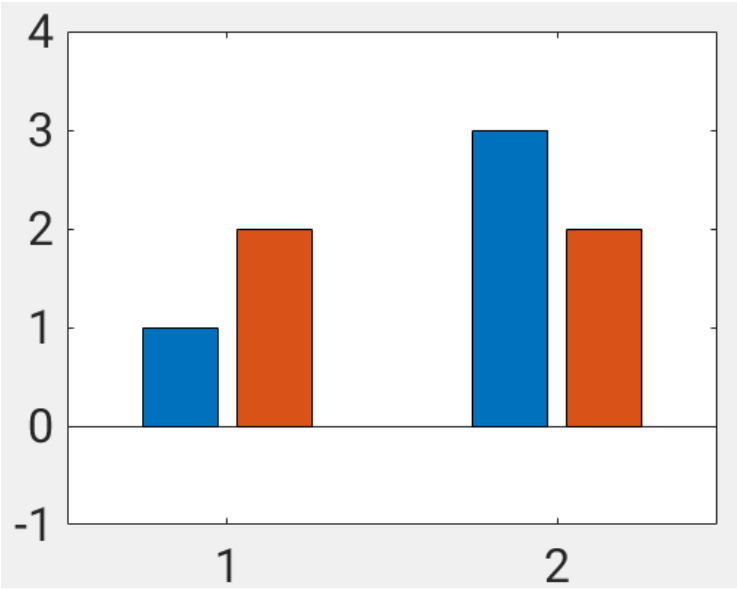
h now contains both bar charts from the figure

We can select one of them to see the properties

Setting Properties

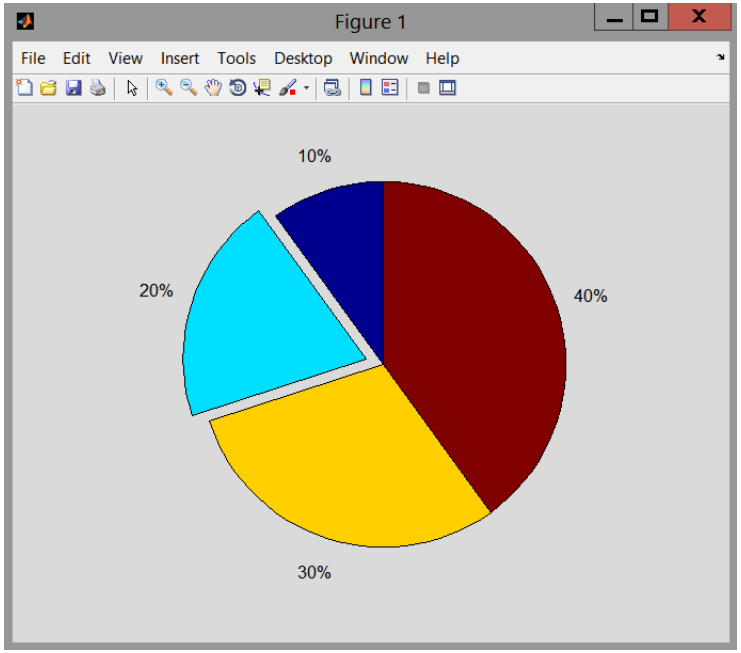
- Once we save a graphics object as a variable we can set its properties, such as those listed above, using the **set** command.
- Note we can **get** and **set** properties with any plot or graphics object.

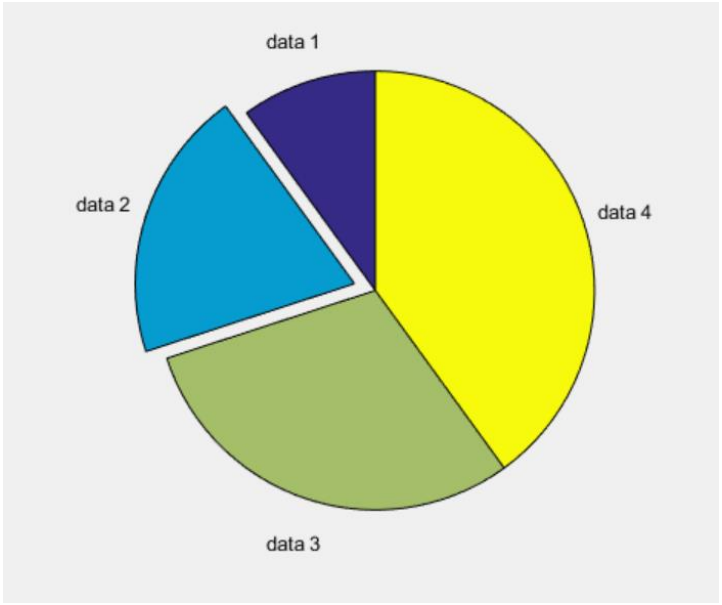
```
>> set(ax,'YLim', [-1 4])
>> set(ax,'Fontsize', 24)

>> get(ax,'YLim')

ans =

    -1     4
```



2D Pie Charts

```
>> explode = [0,1,0,0];
>> pie(x,explode);
```

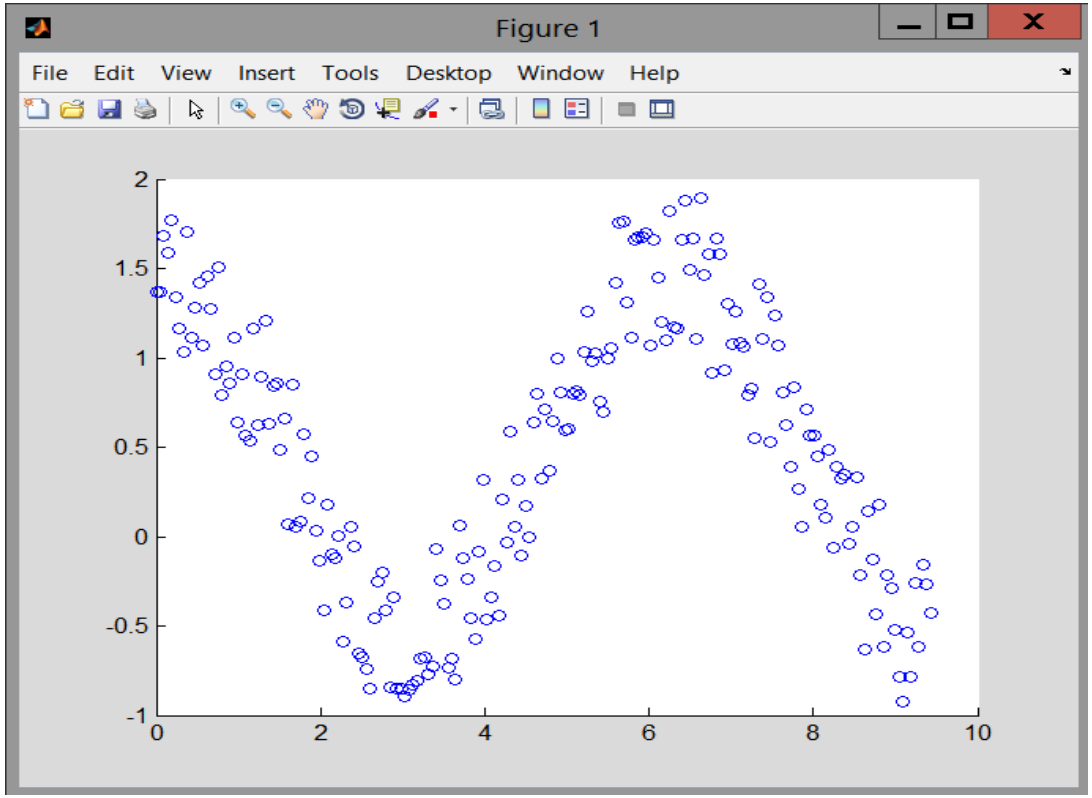





```
>> explode = [0,1,0,0];
>> labels = { 'data 1', 'data 2', 'data 3', 'data 4' };
>> pie(x,explode,labels);
```

2D Scatter Plots

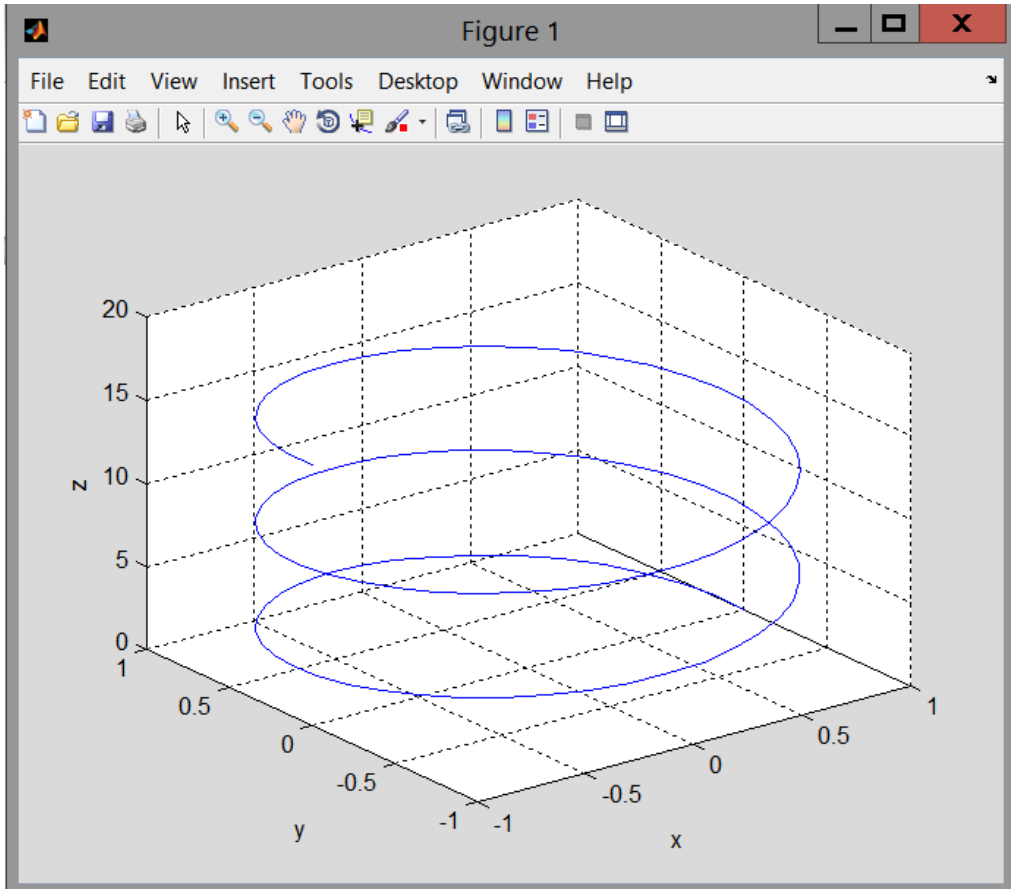
```
>> x = linspace(0,3*pi,200);
>> y = cos(x) + rand(1,200);
>> scatter(x,y)
```



3D Line Plots

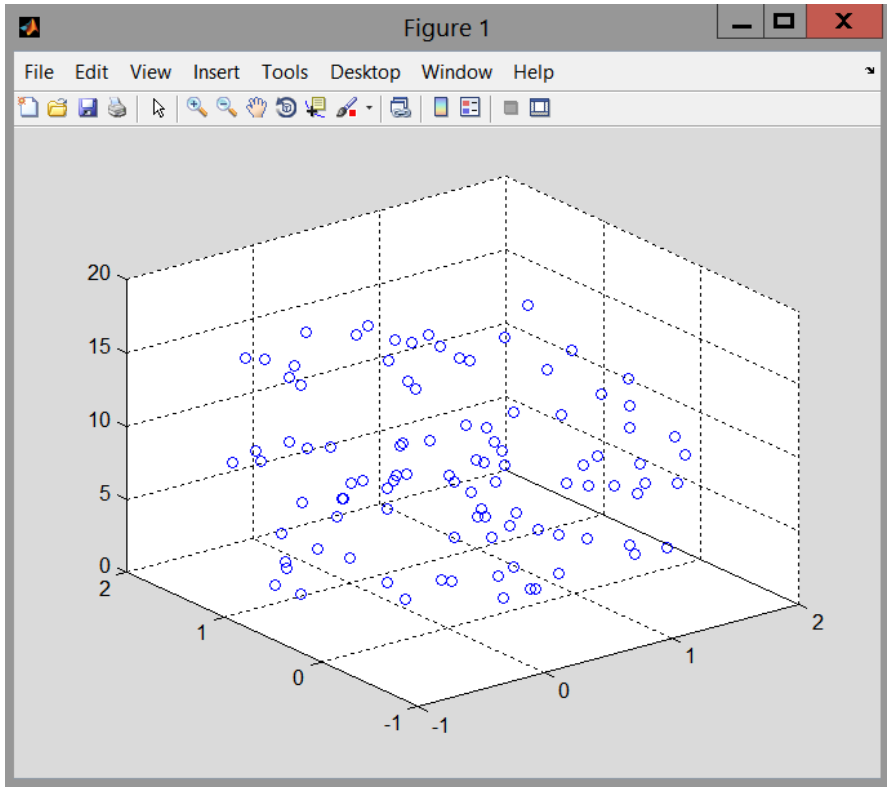
```
>> t = linspace(0,5*pi);
>> x = cos(t);
>> y = sin(t);
>> z = t;

>> plot3(x,y,z)

>> xlabel('x')
>> ylabel('y')
>> zlabel('z')

>> grid on
```

Makes a grid visible



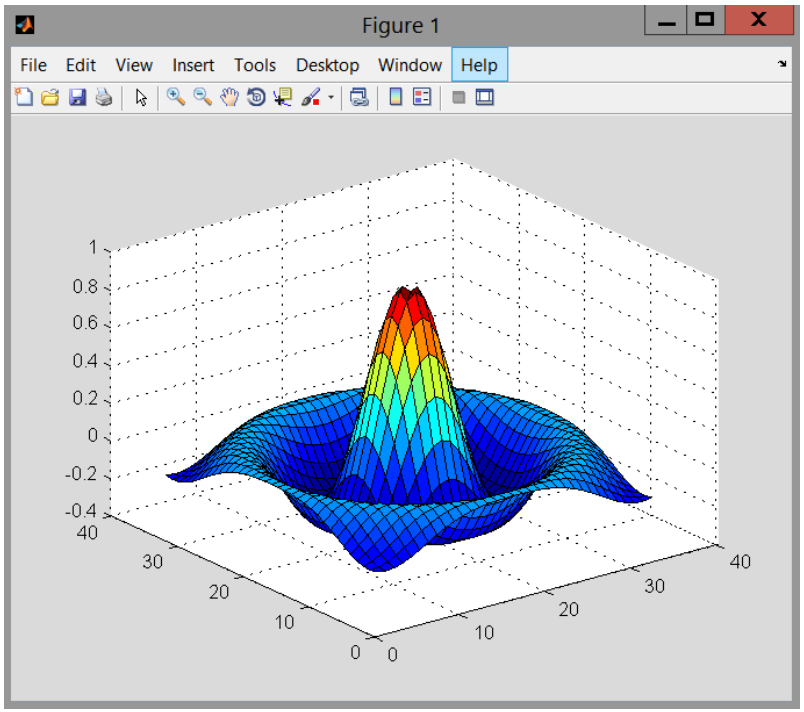
3D Scatter Plots

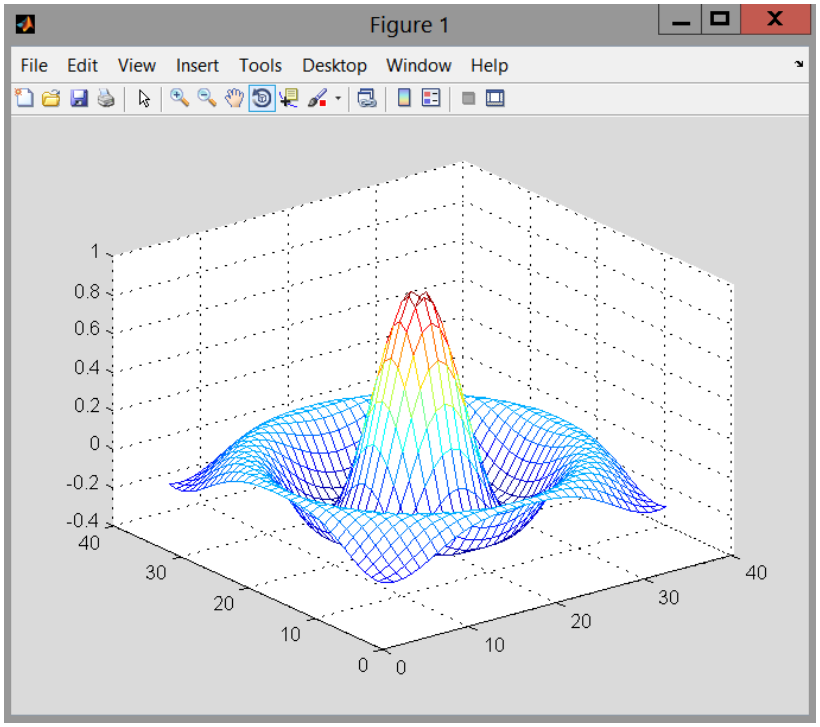
```
>> z = linspace(0,5*pi);
>> x = cos(z)+rand(1,length(z));
>> y = sin(z)+rand(1,length(z));

>> scatter3(x,y,z)
```



3D Surface Plots

```
>> [X,Y] = meshgrid(-8:0.5:8);
>> R = sqrt(X.^2 + Y.^2);
>> Z = sin(R)./R;
>> mesh(Z)
>> surf(Z)
```

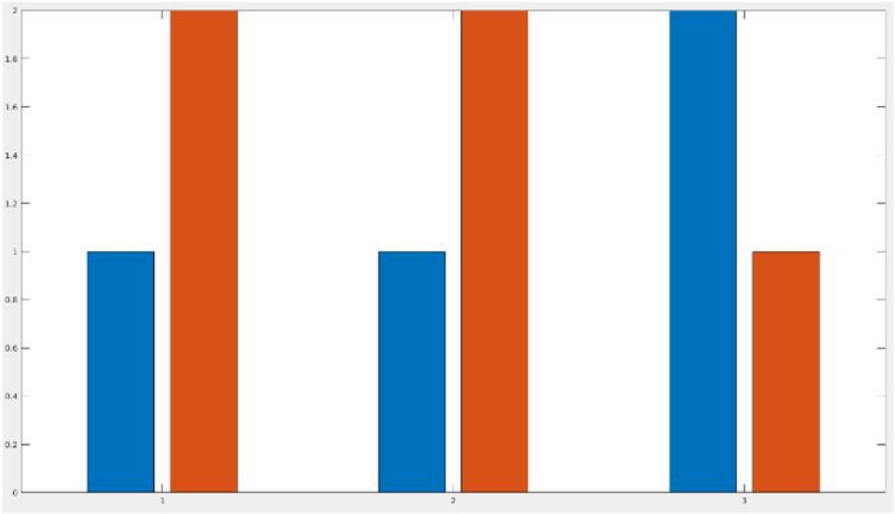




- To plot against the original input coordinates we can do:

```
>> surf(X,Y,Z)
```

Which command produced this bar graph?



bar([2,1;1,2;1,2])

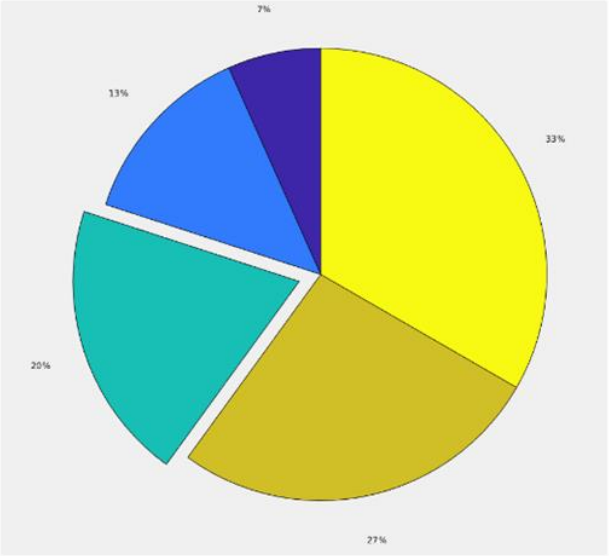
bar([1,2];[1,2];[2,1])

bar([1,2;1,2;2,1])

bar([1,2,3;1,2])

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Which command produced this pie chart?



pie([7,13,20,27,33],[0,1,0,0,0])

pie(1:5,[0,0,1,0,0])

pie([7,13,20,27,33],[0,0,0,1,0])

pie(1:5,[0,1,0,0,0])

To 0

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

2.2 MATLAB Files

- Most common MATLAB files have .m, .mat or .fig extension.
- .m-files are either scripts or functions which store your code.
- .fig-files are graphics objects for various plots.
- .mat-files are matrix data where variables are stored.
- Pre-installed MATLAB files are accessible from any directory, but user files can only be accessed by making the working directory the location of the file or by specifying the full path.

2.3 Conventions

- Make filename and variables/parameters descriptive and relevant.
- Separate words by capitalisation or underscore (e.g. firstFilename, second_filename).
- Put a block comment at the top of each file which names and describes what the file does: %{ …Comments... %}
- Put comments throughout the code to make it “human-readable”. This will help you and others when you work in a team (and me mark your code): % Comment

Example File (Q5_sol.m)

```
%% Question 5
function A = Q5_sol(f,a,b,n)
%{
Implements the composite trapezium rule:
A = (h/2) * (f(x0) + 2f(x1) + 2f(x2) + … + f(xn))
%}

% Step size:
h = (b-a)/n;

% x-values at which to evaluate function:
x = a:h:b;

% Vectorised formula for integral:
A = (h/2)*( f(x(1)) + 2*sum( f(x(2:end-1)) ) + f(x(end)) );
```

Block comment at the top to describe the file

Helpful comments throughout

2.4 Scripts

- MATLAB files which require no input.
- Series of commands which MATLAB executes one after the other.
- Useful for experimentation and drafting functions.
- Can be used as a basis to manage a family of functions.
- Create new m-file in MATLAB editor and give it a name.
- Run the file using **F5**, or typing the **filename** at the command prompt (>>)

2.5 The `disp` Command

- The **`disp`** command displays an output in the command window.

- It is useful to show the user a result when running a script or function.

```
>> disp('hello')
hello

>> x = 2;
>> disp(['x is equal to ',num2str(x)])
x is equal to 2
```

Converts a number to a string

Concatenate the strings inside [ ]

EXAMPLE 1

Make sure to save your file with a descriptive name

Editor - untitled*

untitled*

```
1    % Script for calculating area of triangle.
2    b = 3;
3    h = 4;
4    A = 0.5*(b*h);
5    disp([ 'The area is ', num2str(A), '(no units)' ])
```

Editor - /home/jonathan/Documents/KMITL/Numerical-Methods/Matlab-Sketches/areaTri.m

areaTri.m

```
1    % Script for calculating area of triangle.
2 -  b = 3;
3 -  h = 4;
4 -  A = 0.5*(b*h);
5 -  disp([ 'The area is ', num2str(A), ' (no units)' ])
```

Workspace

Name	Value
A	6
b	3
h	4

Command Window

```
>> areaTri
The area is 6 (no units)
```

When poll is active, respond at **PollEv.com/jsands601**

What does this script do?

```
% Script for something...
x = 7;
y = 4;
sol = pi*y*x^2;
disp(['The answer is ',num2str(sol)])
```

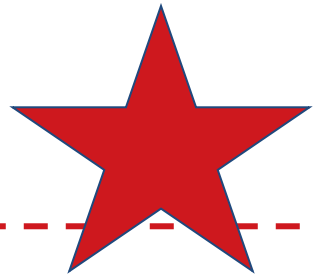
- Surface area of a cylinder with height 4 and radius 7
- Surface area of a cylinder with height 7 and radius 4
- Volume of a cylinder with height 4 and radius 7
- Volume of a cylinder with height 7 and radius 4

To 0

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

2.6 Functions

- MATLAB files which can take inputs and produce outputs.
- More versatile than scripts. Can call the same code repeatedly with different inputs.
- Useful for modularising projects.
- Has the form:



```
function [outputs] = function_name(inputs)
```

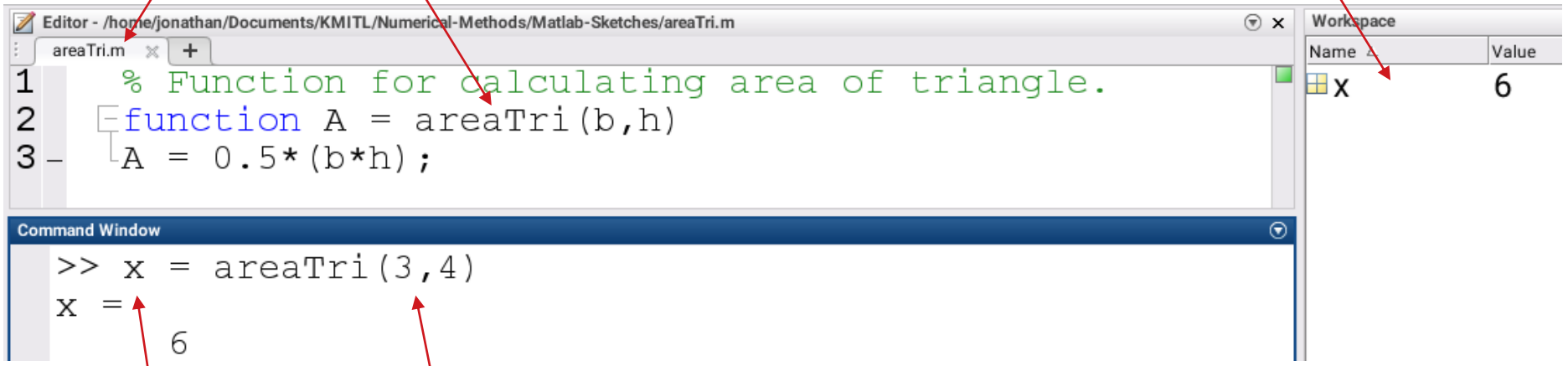
Keyword

MUST match the filename!

EXAMPLE 2 Convert the previous script into a function file.

Filename and function name match

Only *x* was created in workspace. No *A*, *b* or *h*.



Input arguments

Output argument

- Can have more than 1 output in square brackets.

```
function [y1, y2] = add_minus(a,b)
y1 = a+b;
y2 = a-b;
```

This is saved as a file called "add_minus.m"

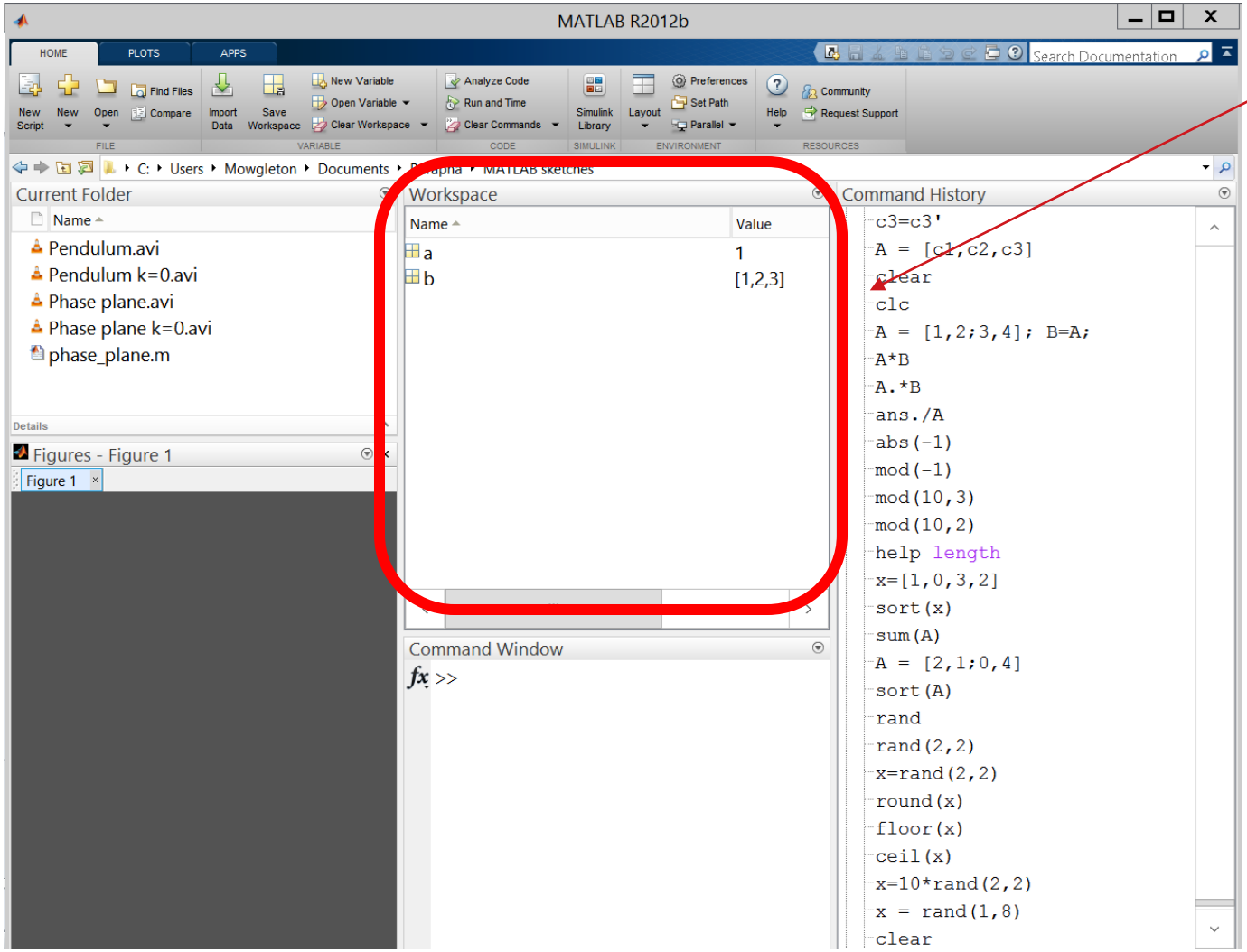
```
>> z = add_minus(3,5)
z =
     8
```

Only 1 output argument asked for so first one is returned (a+b)

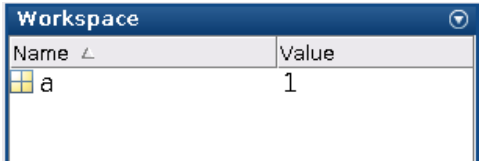
```
>> [z1,z2] =
add_minus(3,5)
z1 =
     8
z2 =
    -2
```

2 output arguments specified so both are returned

2.7 Local Vs. Global Variables



Global workspace



GLOBAL WORKSPACE

```
>> a = 1;
```

```
function a=function1(b)

a = 2*b;
```

LOCAL

```
function a=function2(c)

a = 3*c;
```

LOCAL

```
>> function1(2)
ans =
      4
```

```
>> function2(2)
ans =
      6
```

```
>> a
a =
      1
```

How many inputs/outputs?

```
function [x,y,z] = myfunc(A,B)
```

2 inputs
3 outputs

3 inputs
2 outputs

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Output of >> P = myfunc(5,3)?

```
function [p,r] = myfunc(A,B)
x = A-B;
y = A+B;
p = 2*x;
r = x+y;
P = 3*x;
```

p = 6

p = 4

P = 4

P = 6

p = 4, r = 8

P = 4, r = 8

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

2.8 Decisions

- **Keywords** are used to control the execution of commands.

```
break
case
catch
continue
else
elseif
end
for
function
global
if
otherwise
persistent
return
switch
try
while
```

- Never use keywords as variable names or filenames.

True vs. False

- Use **if/else/switch** to check for **TRUE** or **FALSE** statements.
- In MATLAB "**1**" means **TRUE**, and "**0**" means **FALSE**.
- If I type,

```
>> 1 > 3
```

```
0
```

(MATLAB returns 0, which is FALSE)

- If I type,

```
>> 1 < 3
```

```
1
```

(MATLAB returns 1, which is TRUE)

- If I type,

```
>> 1 < 3 && 2 > 5
```

"1 less than 3 ***AND*** 2 greater than 4"

```
    0
```

(MATLAB returns 0, which is FALSE)

- If I type,

```
>> 1 < 3 || 2 > 5
```

"1 less than 3 ***OR*** 2 greater than 4"

```
    1
```

(MATLAB returns 1, which is TRUE)

Output of >> 2*3 < 2*4 || 2*4 < 2*3

0

1

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

If-Else

Syntax:

```
if condition 1
      some code
elseif condition 2
      some other code
else
      code to execute for all other cases
(default)
end
```

EXAMPLE 3

```
function ifElse1(x)

if x < 0
     disp('x is negative.')
elseif x == 0
     disp('x is equal to
zero.')
else
     disp('x is positive.')
end
```

Comparative equals (not assignment)

```
>> ifElse1(2)
x is positive.
```

EXAMPLE 4

```
function ifElse2(x,y)

if x > 0 && y > 0
     disp('x and y are positive.')
elseif x > 0 || y > 0
     disp('Either x or y is
positive.')
else
     disp('Another result.')
end
if x ~= y
     disp('x and y are not equal.')
end
```

And

Or

Not equal

```
>> ifElse2(2,-2)
Either x or y is positive.
x and y are not equal.
```

Switch

Syntax:

```
switch variable
 case 1
      some code
 case 2
      some other code
 otherwise
      code to execute for all other cases
(default)
end
```

EXAMPLE 5

```
function myswitch(x)

switch x
      case 'a'
            disp('First letter of alphabet.')
      case 'b'
            disp('Second letter of alphabet.')
      otherwise
            disp('Something else.')
end
```

```
>> myswitch('a')
First letter of alphabet.
```

Output of >> afunc(-2,5,-3)

```
function afunc(a,b,c)
if a > b || a > c && a < 0
    disp("Hello")
elseif a > b && a > c && a < 0
    disp("Hi")
else
    disp("Ahoy")
end
```

Hello

Hi

Ahoy

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

2.9 Loops

- There are 2 types of loop – `while` and `for`.
- They are used to repeat the same command or code block over and over again.
- We must be extremely careful not to cause **infinite loops**.
- If you cause an infinite loop you must cancel the computation or your **computer will crash**.

While

- Syntax:

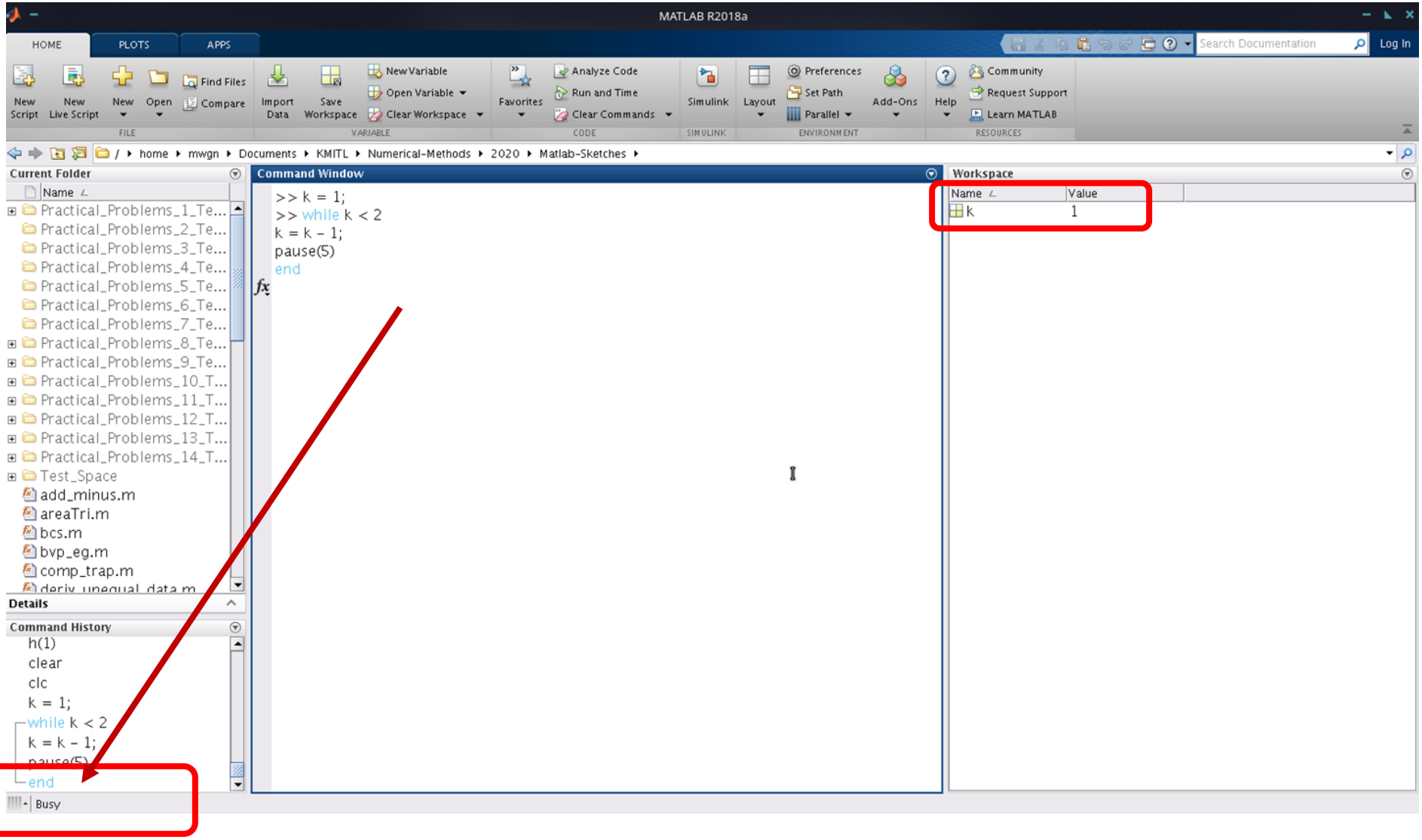
```
while condition is true
   execute this code
end
```

EXAMPLE 6

```
k = 0;
while k <= 10
      disp(k)
      k = k + 1;
end
```

Very important to avoid infinite loop. If infinite loop occurs, interrupt MATLAB execution with **CTRL+C** at the command prompt.

An Infinite Loop ⟶ Press **CTRL+C**



Stopping a While Loop Early

- We can **break** a while loop before the last iteration if a condition is reached.

```
x = 10;
while 1
      x = x - 5;
      if x < 0
            break
      end
end
```

For

Syntax:

```
for values in this range
     execute this code
end
```

```
x = [1,2,3,4,5];
for k = 1:length(x)
     x(k) = 2*x(k);
end
>> x
x =
     2  4  6  8  10
```

Note right hand side is executed before assignment to the left

Which Loop to Use?

- Use `while` **when you don't know** how many iterations you will need – Loop until criteria is met.

- Use `for` **when you know** the number of iterations you want.

2.10 Modulo

- Useful mathematical function in programming. It gives the **remainder after division**.

7 % 2 = 1 ⟶ **7 divided by 2 has remainder 1**

17 % 3 = 2 ⟶ **17 divided by 3 has remainder 2**

- In MATLAB:

```
>> mod(15,3)
ans =
0

>> mod(17,3)
ans =
2
```

Which of these cause an infinite loop?

```
k = 2;
while 1
   k = 2*k;
   if k > 10
      break
   end
end
```

```
k = -2;
while 1
   k = 2*k;
   if k > 10
      break
   end
end
```

```
k = 2;
while k < 1000
   k = -abs(2*k);
end
```

```
k = 2;
while k < 1000
   k = -2*k;
end
```

To 0

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**