Chapter 4

Using Script Files and Managing Data

MATLAB An Introduction With Applications, 6th Edition
Dr. Amos Gilat
The Ohio State University

Slide deck by
Dr. Greg Reese
Miami University

In this chapter will study

- How to input data into a script file
- How MATLAB stores data
- Ways to display and save data
- How to exchange data between MATLAB and other programs

MATLAB *workspace* made up of variables that you define and store during a MATLAB session. It includes variables

- Defined in the Command Window
- Defined in script files

A script file can access all variables that you defined in the Command Window

`whos` command is like `who` command (Chapter 1) but with more information

```
>> 'Variables in memory'
ans =
Variables in memory
>> a = 7;
>> E = 3;
>> d = [5,  a+E,  4,  E^2]
d =
     5    10     4     9
>> g = [a, a^2,  13;  a*E,  1,  a^E]
g =
      7    49     13
     21     1    343
>> who
Your variables are:
E    a    ans  d    g
>> whos
  Name      Size            Bytes  Class     Attributes

  E         1x1                 8  double
  a         1x1                 8  double
  ans       1x19               38  char
  d         1x4                32  double
  g         2x3                48  double
```

Typing a string.

The string is assigned to ans.

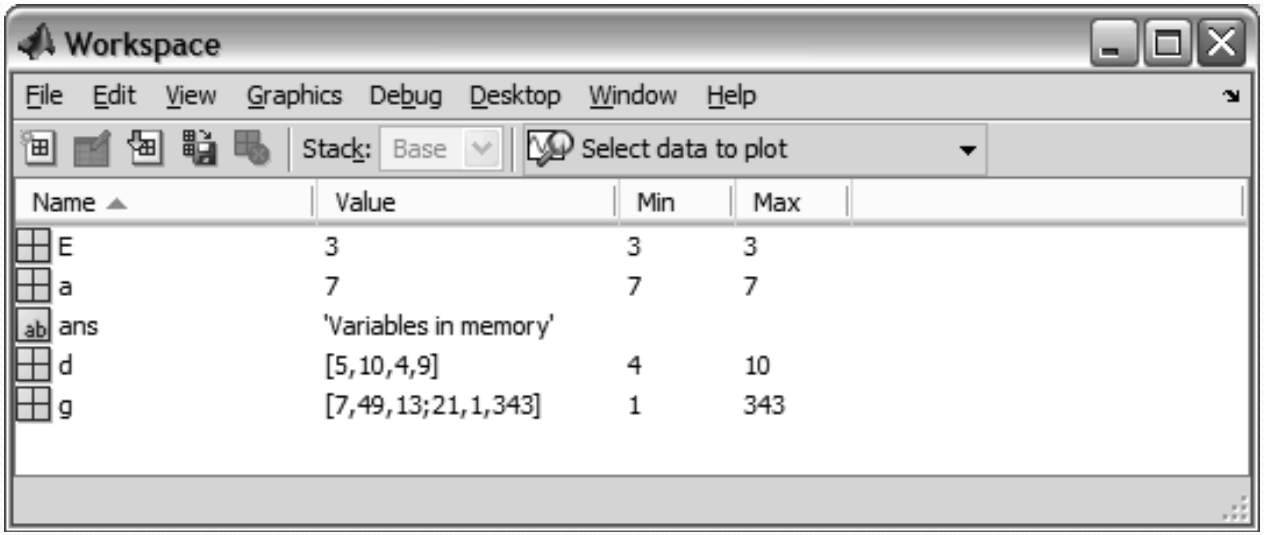
Creating the variables a, E, d, and g.

The who command displays the variables currently in the workspace.

The whos command displays the variables currently in the work-space and information about their size and other information.

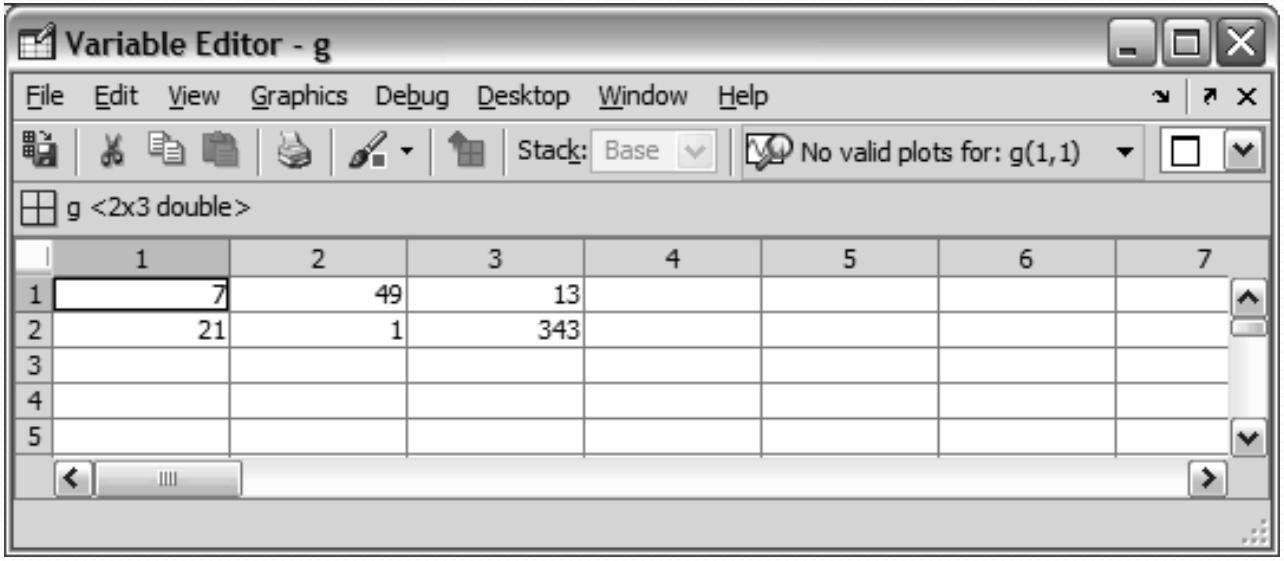
Can also view workspace variables in the Workspace Window

- To open Workspace Window, click on Layout icon, then Workspace



To edit (change) a variable in the Workspace Window

1. Double-click on variable to get the Variable Editor Window
2. In that window can modify numbers



	1	2	3	4	5	6	7
1	7	49	13				
2	21	1	343				
3							
4							
5							

In Variable Editor Window

- To change a character, place cursor to right of character and press BACKSPACE or to left and press DELETE
- To delete a number, select it by dragging or double-clicking, then press DELETE or BACKSPACE

To delete a variable from the Workspace Window

- Select variable by dragging or double-clicking, then
  - Press DELETE or BACKSPACE
    or
  - Right click and select Delete
- Can also delete a variable from Command Window with command

```
>> clear variable_name
```

e.g.,

```
>> clear g
```

When MATLAB executes (runs) a script file, any variables used in file must already have values assigned to them, i.e., the variables must already be in the workspace

Can assign a value to a variable in three ways

1. Assign value in script file

- Assignment statement is part of script
- To use different value, must edit file, save file, and run file again

Note – when variable value (a number) is part of script, value is said to be *hard-coded*

The following is an example of such a case. The script file (saved as Chapter4Example2) calculates the average points scored in three games.

```
% This script file calculates the average points scored in three games.
% The assignment of the values of the points is part of the script file.
game1=75;
game2=93;
game3=68;
ave_points=(game1+game2+game3)/3
```

The variables are assigned values within the script file.

The display in the Command Window when the script file is executed is:

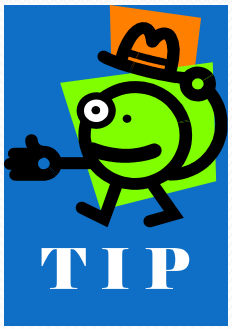
```
>> Chapter4Example2

ave_points =
   78.6667
>>
```

The script file is executed by typing the name of the file.

The variable `ave_points` with its value is displayed in the Command Window.

2. Assign value in Command Window

- Define variable and assign its value in Command Window
  - From before, know that script file will recognize variable
- To use different value, assign new value in Command Window and run file again
  - Don't need to resave file



Instead of retyping entire command, use up-arrow to recall command and then edit it

For the previous example in which the script file has a program that calculates the average of points scored in three games, the script file (saved as Chapter4Example3) is:

```
% This script file calculates the average points scored in three games.
% The assignment of the values of the points to the variables
% game1, game2, and game3 is done in the Command Window.

ave_points=(game1+game2+game3)/3
```

The Command Window for running this file is:

```
>> game1 = 67;
>> game2 = 90;
>> game3 = 81;
```

The variables are assigned values in the Command Window.

```
>> pChater4Example3

ave_points =
   79.3333

>> game1 = 87;
>> game2 = 70;
>> game3 = 50;
>> Chapter4Example3

ave_points =
    69
>>
```

The script file is executed.

The output from the script file is displayed in the Command Window.

New values are assigned to the variables.

The script file is executed again.

The output from the script file is displayed in the Command Window.

3. Assign by prompt in script file

- Script file *prompts* (asks) user to enter a value, then script assigns that value to a variable

Use MATLAB `input` command to ask for and get value from user

```
variable_name=input('prompt')
```

`prompt` **is text that** `input` **command displays in Command Window**

- You must put text between single quotes

```
variable_name=input('prompt')
```

When script executes `input` command

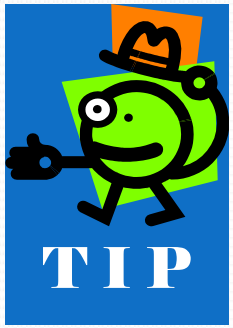
1. Displays prompt text in Command Window
2. Puts cursor immediately to right of prompt
3. User types value and presses ENTER
4. Script assigns user's value to variable and displays value unless `input` command had semicolon at end

```
% This script file calculates the average of points scored in
three games.
% The points from each game are assigned to the variables by
% using the input command.
game1=input('Enter the points scored in the first game ');
game2=input('Enter the points scored in the second game ');
game3=input('Enter the points scored in the third game ');
ave_points=(game1+game2+game3)/3
```

Script output (in Command Window)

```
>> Chapter4Example4
Enter the points scored in the first game   67
Enter the points scored in the second game   91
Enter the points scored in the third game   70

ave_points =
    76
>>
```

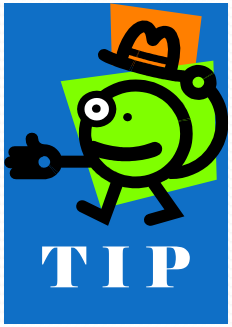
The computer displays the message. Then the value of the score is typed by the user and the **Enter** key is pressed.



It's helpful to put a space, or a colon and a space, at the end of the prompt so that the user's entry is separated from the prompt.

Example script file:

```
age = input('Age in 2012');

age = input('Age in 2012 ');

age = input('Age in 2012: ');
```



Output of script shown with value of "30" that user entered

```
Age in 201230
```
bad

```
Age in 2012 30
```
better

```
Age in 2012: 30
```
good

Can also prompt for and assign a text string to a variable.

Method 1

Use input as before but user must type in beginning and ending quote marks

```
>> name = input( 'Your name: ' )
Your name: 'Joe'
name =
     Joe
```

User must type quotes

Method 2

Pass `'s'` as second argument to input.

User should <u>not</u> enter quotes

```
variable_name=input('prompt', 's')
```

```
>> name=input('Your name: ', 's')
Your name: Joe
```

User enters without quotes

```
name =
     Joe
```

When omit semicolon at end of statement, MATLAB displays result on screen. You have no control over appearance of result, e.g., how many lines, what precision in numbers. Can use MATLAB command `disp` for some control of appearance and `fprintf` for full control

`disp` (display) command displays variable values or text on screen

- Displays each time on new line
- Doesn't print variable name

`disp(variable_name)` or

`disp('text string')`

```
>> abc = [5  9  1;  7  2  4]
>> disp(abc)
     5     9     1
     7     2     4

>> disp('The problem has no solution.')

The problem has no solution.
>>
```

A 2×3 array is assigned to variable abc.

The disp command is used to display the abc array.

The array is displayed without its name.

The disp command is used to display a message.

Can display tables with headers using `disp`

- Clumsy because no control of column width – must adjust headers by inserting blanks
- Better to use `fprintf`

fprintf

- Means file print formatted
  - *formatted text* is text that can be read by people
  - *unformatted text* looks random to people but computers can read it
- Can write to screen or to a file
- Can mix numbers and text in output
- Have full control of output display
- Complicated to use

Using the `fprintf` command to display text:

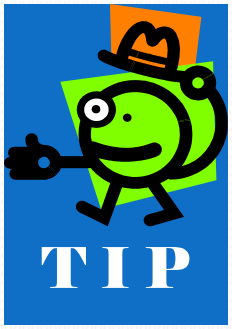
Display text with

```
fprintf('Text to display')
```

Example

```
>> fprintf( 'Howdy neighbor' )
Howdy neighbor>>
```

Problem – Command Window displays prompt (>>) at end of text, not at start of next line!



To make the next thing that MATLAB writes (after a use of `fprintf`) appear on the start of a new line, put the two characters "\n" at the end of the `fprintf` text

```
>> fprintf( 'Howdy neighbor\n' )
Howdy neighbor
>>
```

Can also use \n in middle of text to make MATLAB display remainder of text on next line

```
>> fprintf('A man\nA plan\nPanama\n')
A man
A plan
A canal
Panama
>>
```

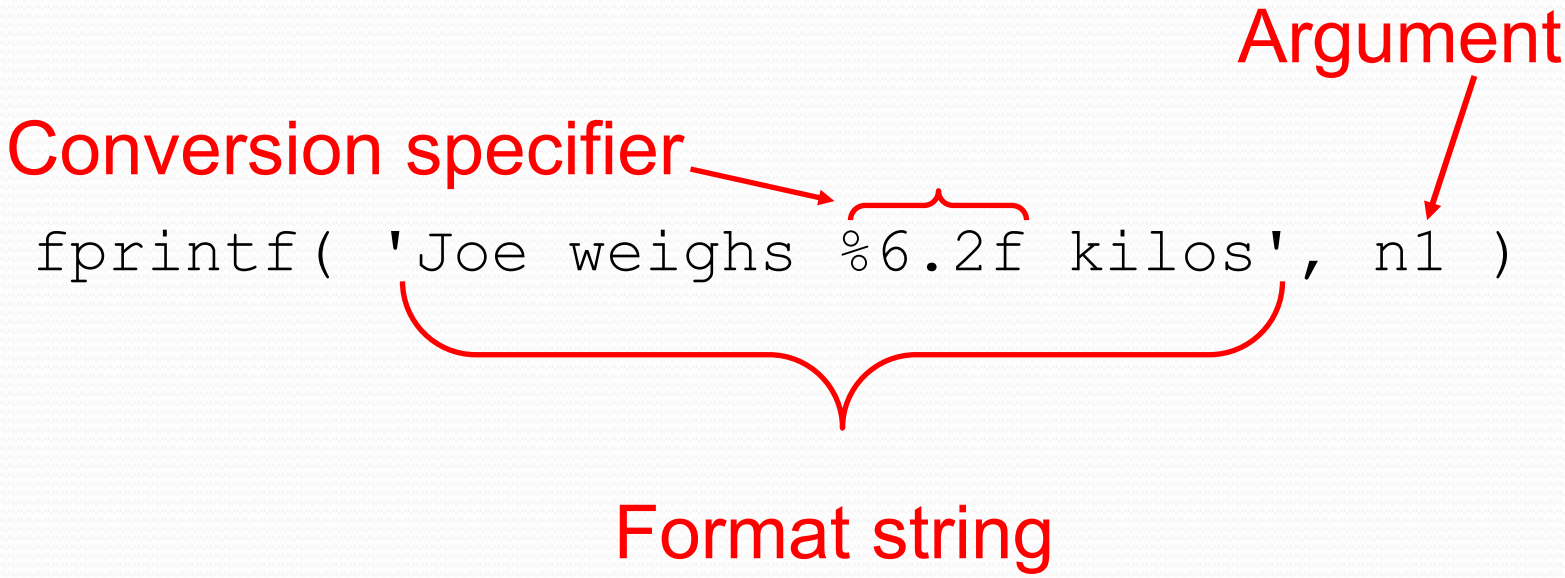
`\n` is an *escape character*, a special combination of two characters that makes `fprintf` do something instead of print the two characters

`\n` – makes following text come out at start of next line

`\t` – horizontal tab

There are a few more

fprintf(format, n1, n2, n3)



```
>> fprintf( 'Joe weighs %6.2f kilos', n1 )
```

Format string

- May contain text and/or conversion specifiers
- Must be enclosed in SINGLE quotes, not double quotes, aka quotation marks (" ")

```
>> fprintf( 'Joe is %d weighs %f kilos', age, weight )
```

Arguments

- Number of arguments and conversion specifiers must be the same
- Leftmost conversion specifier formats leftmost argument, 2nd to left specifier formats 2nd to left argument, etc.

Conversion specifier

```
>> fprintf( 'Joe weighs %f kilos', n1 )
```

Common conversion specifiers

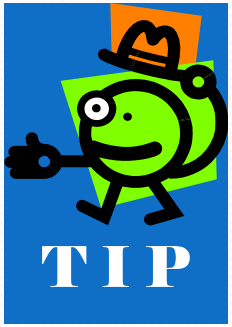
- %f fixed point (decimal always between 1's and 0.1's place, e.g., 3.14, 56.8
- %e scientific notation, e.g, 2.99e+008
- %d integers (no decimal point shown)
- %s string of characters

Conversion specifier

```
>> fprintf( 'Joe weighs %6.2f kilos', n1 )
```

To control display in fixed or scientific, use `%w.pf` or `%w.pe`

- w = width: the minimum number of characters to be displayed
- p = "precision": the number of digits to the right of the decimal point



If you omit "w", MATLAB will display correct precision and just the right length

```
>> e = exp( 1 );
>> fprintf( 'e is about %4.1f\n', e )
e is about  2.7
>> fprintf( 'e is about %10.8f\n', e )
e is about 2.71828183
>> fprintf( 'e is about %10.8e', e )
e is about 2.71828183e+000
>> fprintf( 'e is about %10.2e', e )
e is about  2.72e+000
>> fprintf( 'e is about %f\n', e )
e is about 2.718282
```

Use escape characters to display characters used in conversion specifiers

- To display a percent sign, use %% in the text
- To display a single quote, use '' in the text (two sequential single quotes)
- To display a backslash, use \\ in the text (two sequential backslashes)

Make the following strings

- Mom's apple 3.14
- Mom's apple 3.1415926
- Mom's apple 3.1e+000

```
>> fprintf( 'Mom''s apple %.2f\n', pi )
Mom's apple 3.14
>> fprintf( 'Mom''s apple %.7f\n', pi )
Mom's apple 3.1415927
>> fprintf( 'Mom''s apple %.1e\n', pi )
Mom's apple 3.1e+000
```

Format strings are often long. Can break a string by

1. Put an open square bracket ([ ) in front of first single quote
2. Put a second single quote where you want to stop the line
3. Follow that quote with an ellipsis (three periods)
4. Press ENTER, which moves cursor to next line
5. Type in remaining text in single quotes
6. Put a close square bracket ( ])
7. Put in the rest of the `fprintf` command

Example

```
>> weight = 178.3;
>> age = 17;
>> fprintf( ['Tim weighs %.1f lbs'...
' and is %d years old'], weight, age )

 Tim weighs 178.3 lbs and is 17 years old
```

fprintf is *vectorized*, i.e., when vector or matrix in arguments, command repeats until all elements displayed

- Uses matrix data column by column

```
x=1:5;
y=sqrt(x);
T=[x; y]
fprintf('If the number is: %i, its square root is: %f\n',T)
```

Create a vector x.

Create a vector y.

Create 2 × 5 matrix T, first row is x, second row is y.

The fprintf command displays two numbers from T in every line.

When this script file is executed, the display in the Command Window is:

```
T =
    1.0000    2.0000    3.0000    4.0000    5.0000
    1.0000    1.4142    1.7321    2.0000    2.2361
If the number is: 1, its square root is: 1.000000
If the number is: 2, its square root is: 1.414214
If the number is: 3, its square root is: 1.732051
If the number is: 4, its square root is: 2.000000
If the number is: 5, its square root is: 2.236068
```

The 2×5 matrix T.

The `fprintf` command repeats five times, using the numbers from the matrix T column after column.

Using the `fprintf` command to save output to a file:

Takes three steps to write to a file

Step a: – open file

```
fid=fopen('file_name','permission')
```

`fid` – *file identifier*, lets `fprintf` know what file to write its output in

`permission` – tells how file will be used, e.g., for reading, writing, both, etc.

Some common permissions

- `r` - open file for reading
- `w` - open file for writing. If file exists, content deleted. If file doesn't exist, new file created
- `a` - same as `w` except if file exists the written data is appended to the end of the file
- If no permission code specified, `fopen` uses `r`

See Help on `fopen` for all permission codes

Step b:

Write to file with `fprintf`. Use it exactly as before but insert `fid` before the format string, i.e.,

```
fprintf(fid,'format string',variables)
```

The passed `fid` is how `fprintf` knows to write to the file instead of display on the screen

Step c:

When you're done writing to the file, close it with the command

```
fclose(fid)
```

- Once you close it, you can't use that `fid` anymore until you get a new one by calling `fopen`

Make sure to close every file you open. Too many open files creates problems for MATLAB

Miscellaneous

- If the file name you give to `fopen` has no path, MATLAB writes it to the current directory, also called the working directory
- You can have multiple files open simultaneously and use `fprintf` to write to all of them just by passing it different `fid`s
- You can read the files you make with `fprintf` in any text editor, e.g., MATLAB's Editor window or Notepad

Use `save` command to save workspace or data

Use `load` command to retrieve stored workspace or data

Can use both to exchange data with non-MATLAB programs

Use `save` command to save some or all workspace variables to hard drive

Two forms

```
save file_name
save('file_name')
```

Either one saves all workspace variables, including their name, type, size, value

To only save specific variables, list variables after file name. For example, to save two variables named `var1` and `var2`

```
save file_name var1 var2
save('file_name','var1','var2')
```

All forms store variables in file called "file_name.mat"

- Called "mat" file
- Unformatted (binary) file
  - Only MATLAB can read mat file, not other programs
  - Can't read file in text editor, or MATLAB Editor Window

To save as formatted text (also called *ASCII text*)

```
save file_name -ascii
```

IMPORTANT – only saves values of variables, no other info, even their names!

- Can also just save certain variables, as before
- Usually just use to save value of one variable

To load data in a mat file into workspace

```
load file_name
load( 'file_name')
```

To load only specific variables from mat file, e.g., var1 and var2

```
load file_name var1 var2
load('file_name','var1','var2')
```

- If variable already exists in workspace, it is *overwritten* (its value is replaced by value in file)

To load data in a text file into workspace

```
load file_name
variable = load( 'file_name')
```

- In first form, creates variable called `file_name` and stores all file data in it
- If all rows in file don't have same number of columns, MATLAB displays an error
- Even if data created from multiple variables all with same number of columns, `load` still reads all data into one variable
  - Not very useful in this case

- MATLAB often used to analyze data collected by other programs
- Sometimes need to transfer MATLAB data to other programs
- In this section will only discuss numerical data
  - MATLAB has commands to load and save data from a number of other programs
  - Can also tell MATLAB what format data is in

Will illustrate transferring data with a specific program by discussing Microsoft Excel

- Commonly used to store data
- Works with many programs that gathers data
- Used often by people with technical data but for which MATLAB is overkill

Importing and exporting data into and from Excel:

Import (read) data from Excel with

```
variable_name=xlsread('filename')
```

- Stores all data in one variable
- If Excel file has multiple sheets, reads first one
  - To read from other sheets, pass command the sheet name
- Can read rectangular section of sheet by specifying range in command

Export (write) data to Excel file with

```
xlswrite('filename',variable_name)
```

- Can specify in command name of sheet and range to write to

MATLAB's import wizard is semi-automatic way to read data from any file

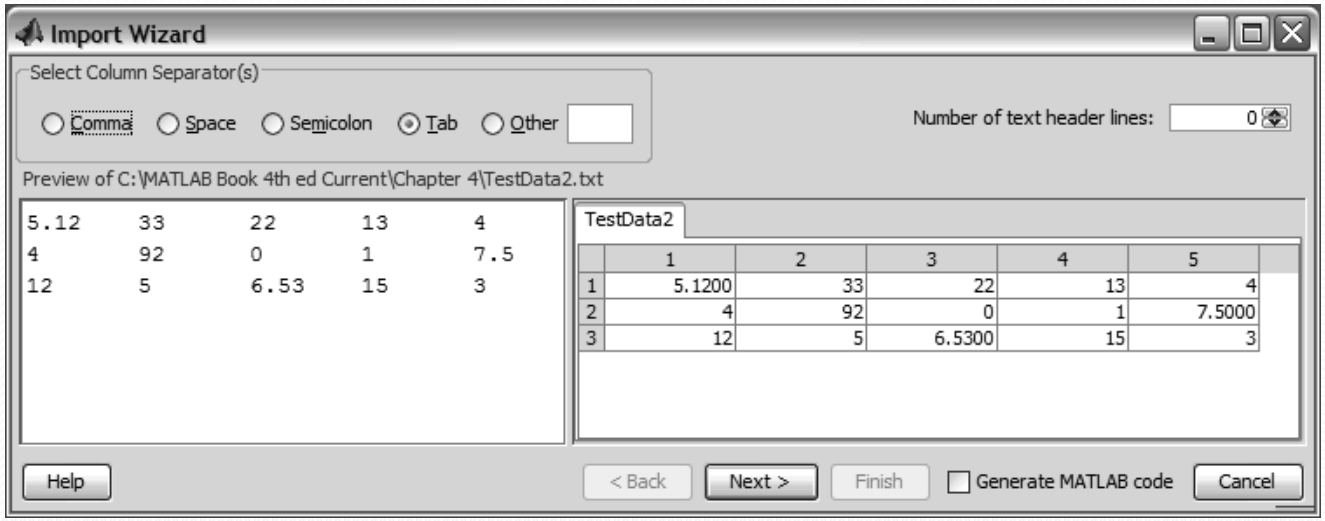
- Wizard shows what it thinks format is
- User can then adjust format

Two ways to start Import Wizard

1. In MATLAB desktop,
   click Import Data icon
2. With command `uiimport`

First Wizard display

- Wizard displays file-selection dialog box
- User picks file
- Wizard shows some of data as it is in file and as how Wizard interprets it
  - User can change column separator or number of text header lines (that Wizard will not try to read)



Second Wizard display

- Shows name and size of variable it will create
- When user selects Finish, Wizard creates that variable in workspace
  - Variable name is file name