**[< My Courses](https://matlabacademy.mathworks.com/" \t "_top)**

**MATLAB Fundamentals**

☰CONTENTClose

**Quick Reference**

Expand all

1. Getting Started
2. Plotting and Common Modifications
3. Working with Live Scripts
4. Creating and Manipulating Arrays
5. Accessing Data in Arrays
6. Mathematical and Statistical Operations with Arrays
7. Visualizing Data in 2D and 3D
8. Conditional Data Selection
9. Review Project I
10. Tables of Data
11. Organizing Data
    1. [Combining Tables](https://matlabacademy-content.mathworks.com/4.11/R2019a/content/quick-reference/mlbe_quick_reference.html#Data%20Manipulation/Grouping%20Data/Combining%20Tables)
    2. [Table Properties](https://matlabacademy-content.mathworks.com/4.11/R2019a/content/quick-reference/mlbe_quick_reference.html#Array%20Types/Tables/Table%20Properties)
    3. [Indexing into Cell Arrays](https://matlabacademy-content.mathworks.com/4.11/R2019a/content/quick-reference/mlbe_quick_reference.html#Syntax/Indexing/Indexing%20into%20Cell%20Arrays)
    4. [Working with Dates and Times](https://matlabacademy-content.mathworks.com/4.11/R2019a/content/quick-reference/mlbe_quick_reference.html#Data%20Operations/Working%20with%20Dates%20and%20Times)
    5. [Operating on Dates and Times](https://matlabacademy-content.mathworks.com/4.11/R2019a/content/quick-reference/mlbe_quick_reference.html#Data%20Operations/Operating%20on%20Dates%20and%20Times)
    6. [Representing Discrete Categories](https://matlabacademy-content.mathworks.com/4.11/R2019a/content/quick-reference/mlbe_quick_reference.html#Array%20Types/Categoricals/Representing%20Discrete%20Categories)
12. Preprocessing Data
13. Common Data Analysis Techniques
14. Programming Constructs
15. Increasing Automation with Functions
16. Troubleshooting Code
17. Review Project II
18. Conclusion

**1. Getting Started**

**Summary: Entering Commands**

**Creating Variables**

|  |
| --- |
| Defining a variable y. |
| C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\6F73DD52.tmp |
| C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\FF0FF930.tmp |
| C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\F1B226BE.tmp |
| C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\2031417C.tmp |

**Modifying Variables**

|  |
| --- |
| You can use the **Up** (↑) and **Down** (↓) arrows on the keyboard to scroll through previous commands. |
| C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\6B4630EA.tmp |

**Cleaning Up**

[clear](http://www.mathworks.com/help/matlab/ref/clear.html)

*Clears variables from the workspace.*

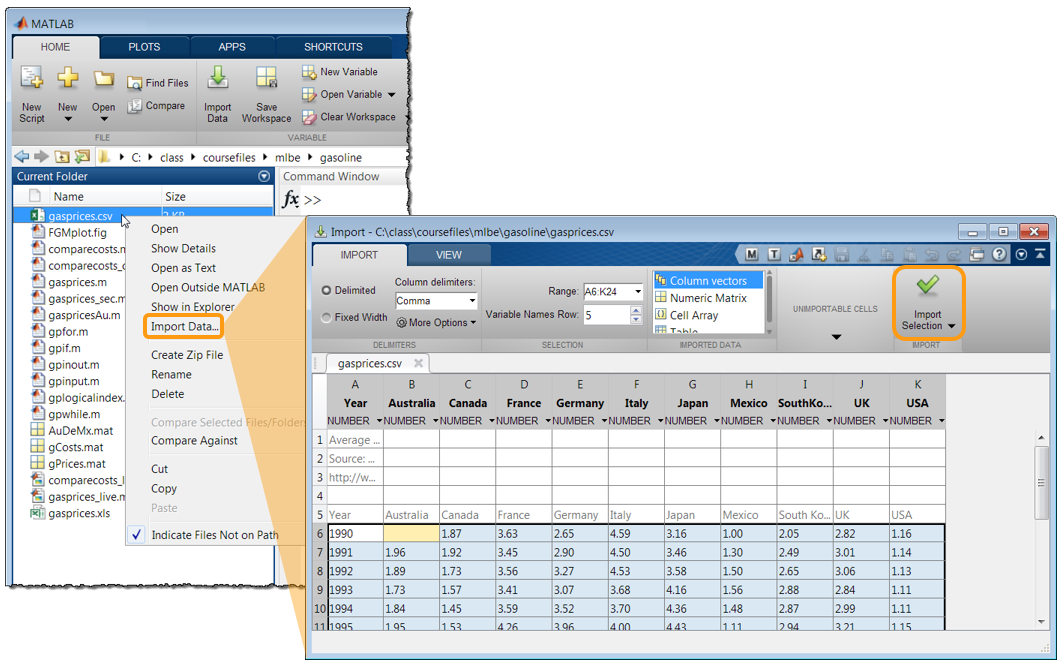
[clc](http://www.mathworks.com/help/matlab/ref/clc.html)

*Clears the Command Window and moves the cursor to the upper left corner of the window.*

**Summary: Getting Data into MATLAB**

**Import Tool**

You can use the Import Tool to interactively bring your data into MATLAB.



**Saving and Loading Data**

You can save and load MAT-files programmatically using the save and load commands.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Creates myData.mat containing current workspace variables. | |  | | --- | |  | |  | |  | | save myData |
| Instead of saving the entire MATLAB workspace, you can selectively save individual variables. Saves x and y to someFile.mat. | |  | | --- | |  | |  | |  | | save someFile x y |
| Loads variables from myData.mat into current workspace. | |  | | --- | |  | |  | |  | | load myData |

[save](http://www.mathworks.com/help/matlab/ref/save.html)

*Saves workspace variables to a MAT-file.*

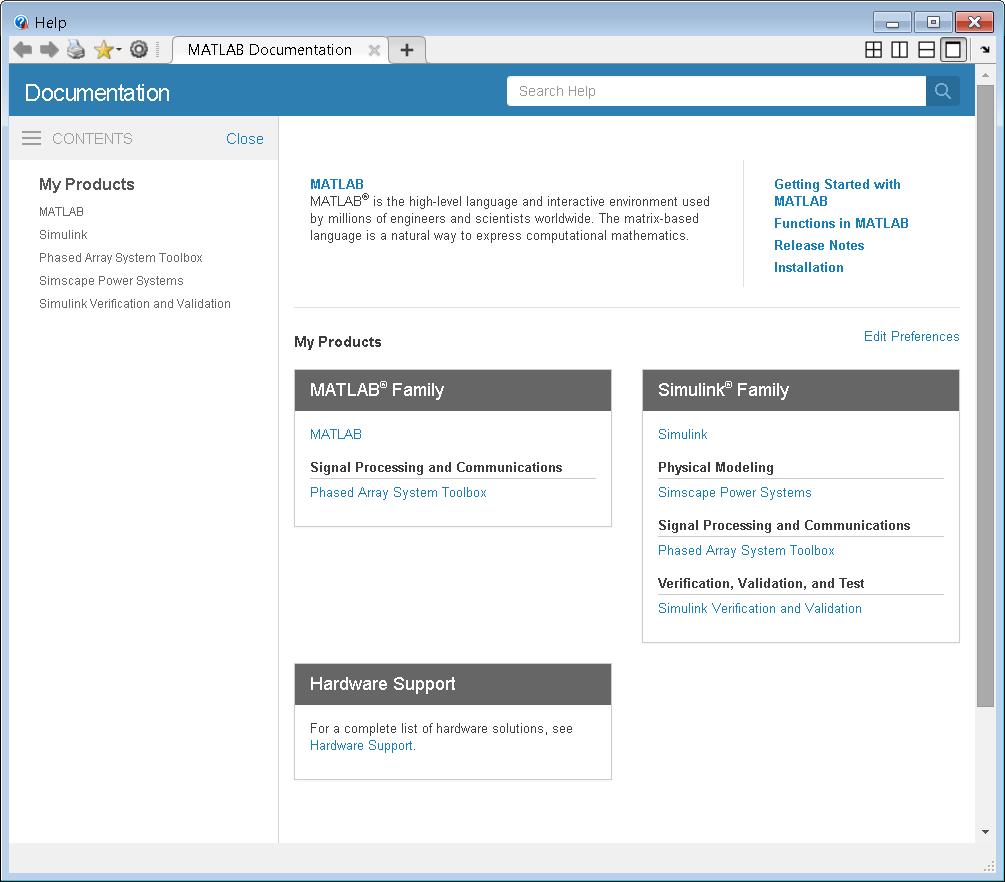
[load](http://www.mathworks.com/help/matlab/ref/load.html)

*Loads variables from a MAT-file into the workspace.*

**Summary: Obtaining Help**

The [doc](http://www.mathworks.com/help/matlab/ref/doc.html) function opens the Help Browser.

>> doc

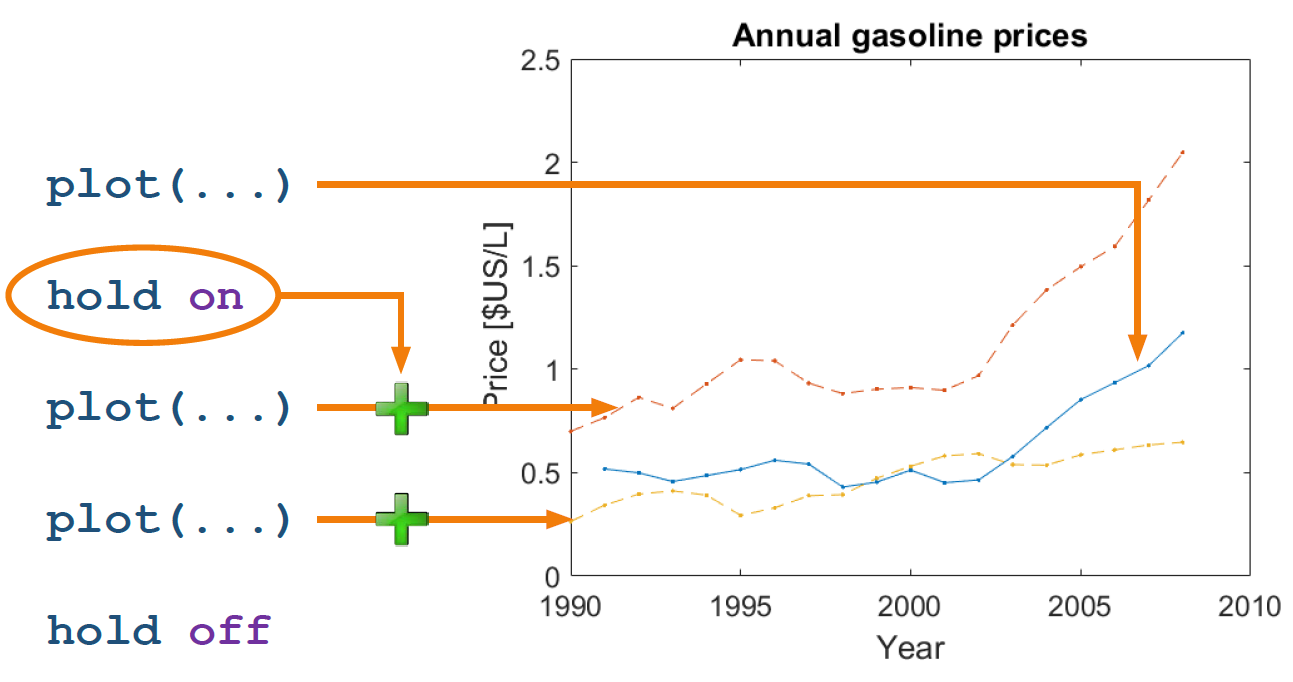


**2. Plotting and Common Modifications**

**Summary: Plotting**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Here, the [plot](http://www.mathworks.com/help/matlab/ref/plot.html) function plots Germany against Year. | |  | | --- | |  | |  | |  | | plot(Year,Germany)  C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\322A1A54.tmp |
| You can also use an optional line specification string to change properties of the plot.  To see options for line style, line color, and marker style, reference the documentation for “[Line Specification](http://www.mathworks.com/help/matlab/ref/linespec.html)”. | |  | | --- | |  | |  | |  | | plot(Australia,Germany,'mo')  C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\8DBAF782.tmp |

Issuing the hold on command allows you to plot multiple data sets together on the same axes.



**Summary: Annotating Plots**

Textual information can be added to plots using separate annotation functions. All of these functions take text input.

[title](http://www.mathworks.com/help/matlab/ref/title.html)

*Add title to plot*

[xlabel](http://www.mathworks.com/help/matlab/ref/xlabel.html)

*Label the x-axis*

[ylabel](http://www.mathworks.com/help/matlab/ref/ylabel.html)

*Label the y-axis*

[legend](http://www.mathworks.com/help/matlab/ref/legend.html)

*Add legend to plot*

Grid lines can be added or removed to plots.

[grid](http://www.mathworks.com/help/matlab/ref/grid.html)

*Display axes grid lines*

**Example**

title('International Gasoline Prices')

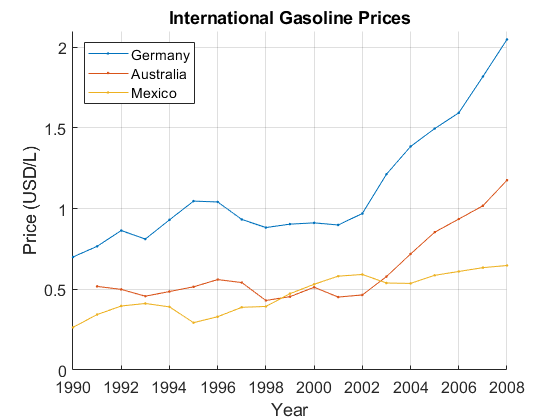
xlabel('Year')

ylabel('Price (USD/L)')

legend('Germany','Australia','Mexico',...

'Location','northwest')

grid on



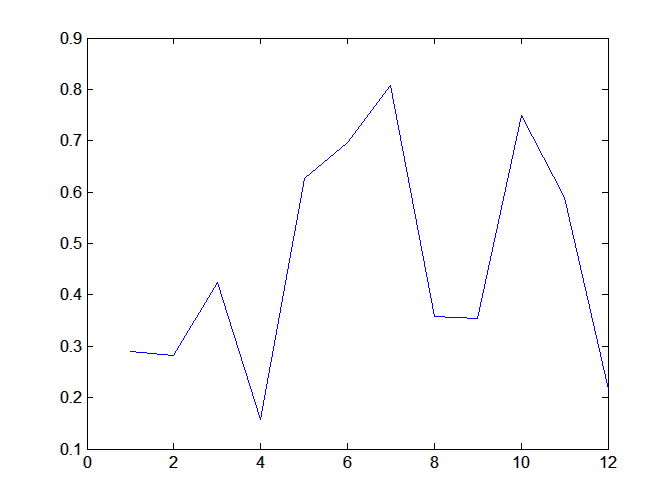
**Summary: Axis Control**

**Get Axes Limits**

v = axis

v =

0 12 0.1 0.9



**Custom Axis Limits**

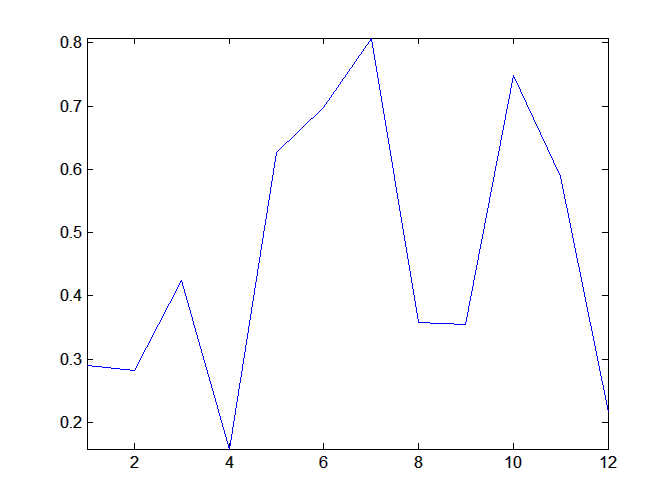
xlim([-1 13])

ylim([-1 2])



**Axis Limits = Data Range**

axis tight



**3. Working with Live Scripts**

**Scripts and the Base Workspace**

Scripts can operate on the variables that are present in the base workspace before the script is run. Similarly, the variables that are generated after the script is run are stored in the base workspace.







*Scripts can use the variables in the base workspace.*

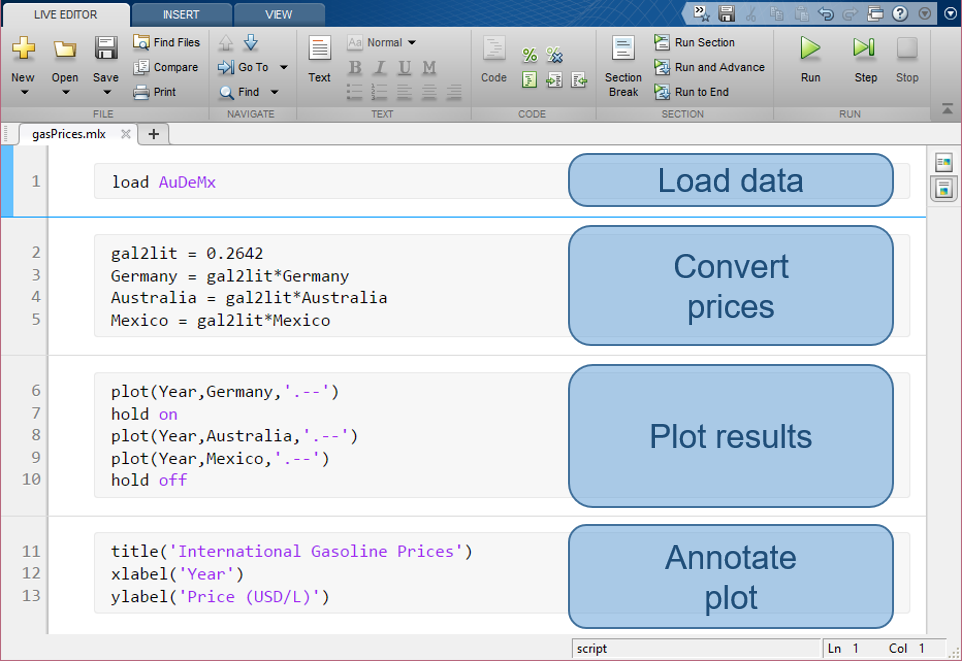
**Summary: Create and Run a Script**

Use the controls in the MATLAB Toolstrip to create and run scripts.

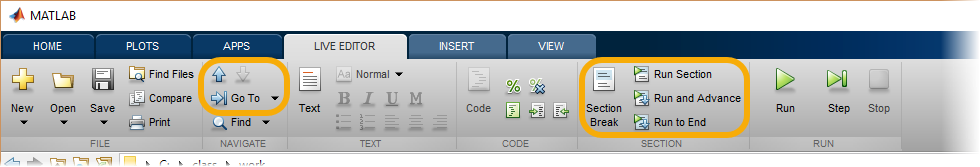
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Create** | C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\AFAC3890.tmp |  | **Run** | C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\2C9E7D1E.tmp |

**Summary: Code Sections**

Code sections allow you to organize your code and run sections of code independently. On the **Live Editor** tab, in the **Section** section, click **Section Break** to create a new code section, or press **Ctrl+Alt+Enter**.



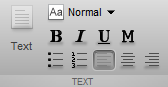
You can use the controls on the **Live Editor** tab of the toolstrip to create, navigate to, and run different sections.



**Summary: Comments and Text**

**Formatted Text**

To insert a line of text, click the C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\32B7F3E8.tmp **Text** button in the **Text** section of the **Live Editor** tab in the MATLAB Toolstrip.  
  
Format the text using the formatting options provided in the **Text** section.



**Comments**

To create a comment, add % *comment* where you want to add more information.

load AuDeMx

% Converts from US$/gal to US$/L

gal2lit = 0.2642;   % conversion factor

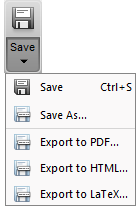
Germany = gal2lit\*Germany;

Australia = gal2lit\*Australia;

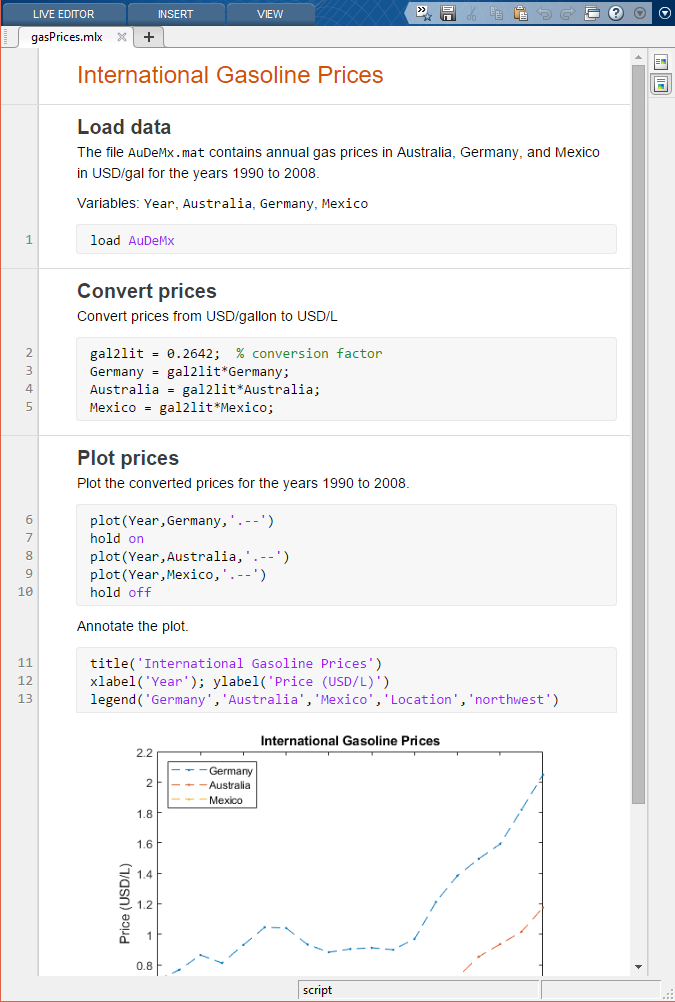
Mexico = gal2lit\*Mexico;

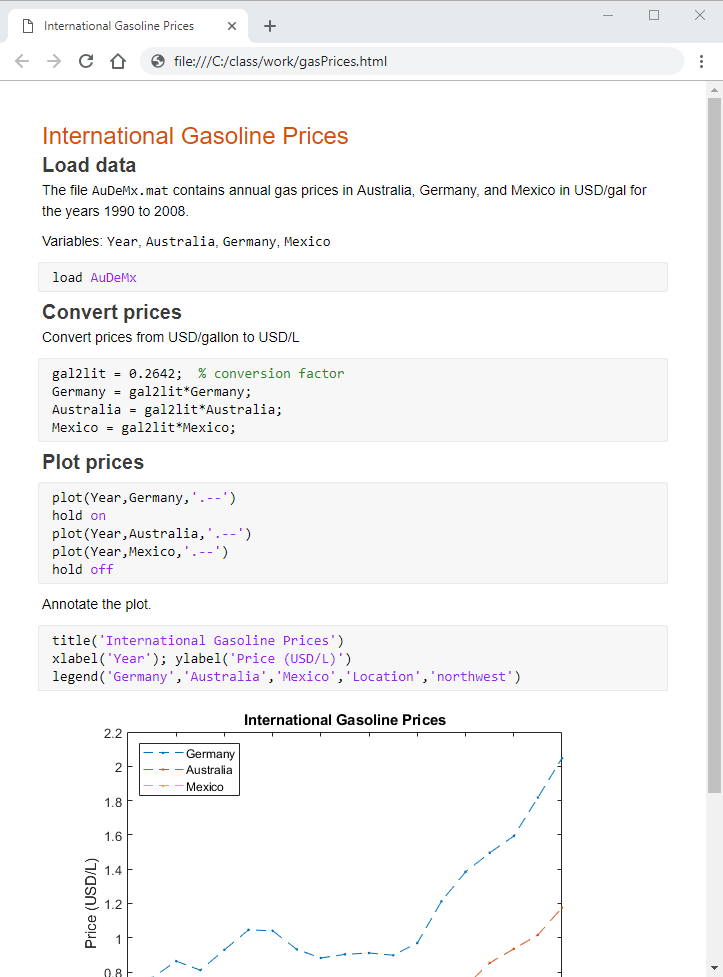
**Exporting Live Script Files**

You can export your live script and results using the **Save** button in the **Live Editor** tab.  
  
Available formats include PDF, HTML, and LaTeX.



The saved file will closely resemble the live script when viewed in the Live Editor with output inline.





**Plain code files (.m)**

To share your code with users of previous MATLAB versions, you can save a live script as a plain code file (.m). Users will be able to open and run your file, and any formatting will be converted to comments with markup.

**4. Creating and Manipulating Arrays**

**Summary: Manually Entering Arrays**

**Create a Row Vector**

Use square brackets and separate the values using a comma or a space.

**Example**

a = [10 15 20 25]

a =

10 15 20 25

**Create a Column Vector**

Use square brackets and separate the values using a semi-colon.

**Example**

b = [2;3;5;7]

b =

2

3

5

7

**Transpose a Vector**

Use the transpose operator

'

.

**Example**

c = b'

c =

2 3 5 7

**Create a Matrix**

Use square brackets and enter values row-by-row. Separate values in a row using a comma or a space, and use a semicolon to start a new row.

**Example**

A = [1 3 5;2 4 6]

A =

1 3 5

2 4 6

**Summary: Creating Evenly-Spaced Vectors**

**Given the Start Value, End Value, and Interval**

Use the colon operator to separate the starting value, interval, and the ending value.

**Example**

a = 3:2:7

a =

3 5 7

**When Interval is 1**

Use the colon operator to separate the starting and the ending value.

**Example**

b = 3:7

b =

3 4 5 6 7

**Given the Start Value, End Value, and Number of Elements**

Use the function linspace when the number of elements in the vector are known.

**Example**

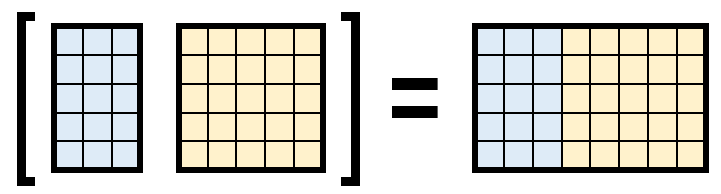
c = linspace(3.2,8.1,5)

c =

3.2 4.42 5.65 6.87 8.1

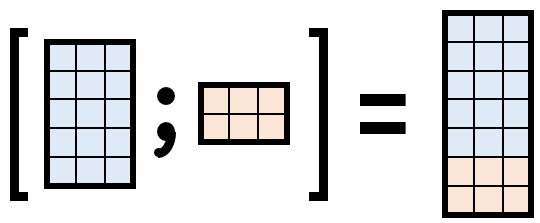
**Summary: Concatenating Arrays**

**Horizontal Concatenation**



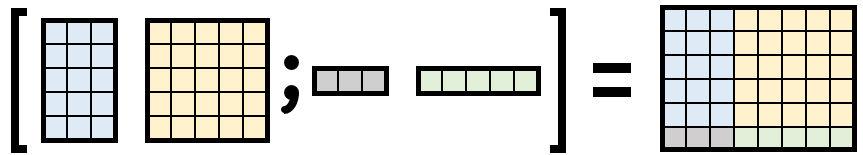
Separate elements using a **comma (,)** or **space (  )**

**Vertical Concatenation**



Separate elements using a **semicolon (;)**

**Combined Concatenation**



Create each row separating elements with a **comma (,)** or **space (  )**, then separate the rows with a **semicolon (;)**

**Summary: Array Creation Functions**

Several functions exist that allow you to create arrays.

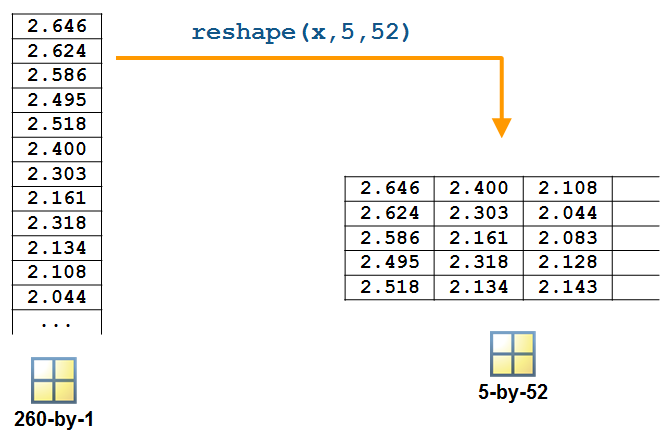
|  |  |  |
| --- | --- | --- |
| C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\286B0B98.tmp |  | C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\582EAD66.tmp |

Most of these functions support the calling syntaxes shown below.

| **Calling syntax** | **Output** |
| --- | --- |
| *fun*(*m*,*n*) | C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\158A8664.tmp  *m-by-n* |
| *fun*(*n*) | C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\62B80E12.tmp  *n-by-n* |

**Summary: Reshaping Arrays**

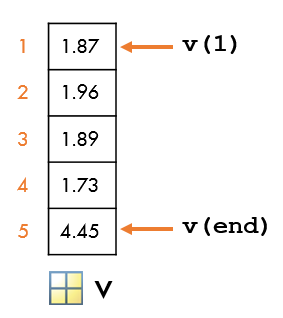
The following column of information is reshaped into a matrix.



|  |  |  |
| --- | --- | --- |
|  |  | x = rand(260,1); |
| Specify the dimensions for the new array. | |  | | --- | |  | |  | |  | | y = reshape(x,5,52); |
| For convenience, you can also leave one of the dimensions blank when calling [reshape](http://www.mathworks.com/help/matlab/ref/reshape.html) and that dimension will be calculated automatically. | |  | | --- | |  | |  | |  | | y = reshape(x,5,[]); |

**5. Accessing Data in Arrays**

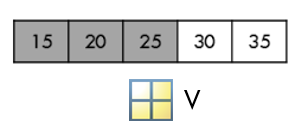
**Summary: Indexing into Vectors**



**Summary: Accessing Multiple Elements**

**Extract Multiple Elements**

To extract elements from a vector:



**Step 1 - Create an Index**

Create a vector containing the index or locations of elements to be extracted.

**Example**

I = 1:3

I =

1 2 3

**Step 2 - Indexing**

Use the index inside the parentheses.

**Example**

s = v(I)

s =

15 20 25

**Step 1 and Step 2**

You can also combine steps 1 and 2.

**Example**

s = v(1:3)

s =

15 20 25

**Assign Multiple Elements**

Use multiple indexing with assignment to modify multiple elements at once.

**Same Value**

Assign one value to multiple elements.

**Example**

v(1:3) = 10

v =

10 10 10 30 35

**Multiple Values**

Assign different values to multiple elements.

**Example**

v(1:3) = [15 10 5]

v =

15 10 5 30 35

**Summary: Matrix Indexing**

**Row, Column Indexing**

Whether you're indexing into a matrix with scalar values or vector values, the format is always the same.

Use the row number to index.  
  
If multiple rows are to be extracted, create a **vector** containing the row numbers and use that as the row index.

output = M(*row*,*column*)

Use the column number or a vector of column numbers to be extracted as the column index.

output = M(*row*,*column*)

This is row-**comma**-column indexing. Separate the row index and column index by a comma.

output = M(*row*,*column*)

**Extract a single element**

output = M(2,3)

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\5DD079AA.tmp*M*

**Extract multiple elements**

output = M(2,[3 4])

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\BDA2CF48.tmp*M*

**Extract multiple contiguous elements**

output = M(2,2:4)

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\82D21C96.tmp*M*

**Extract complete rows or columns**

output = M(2,:)

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

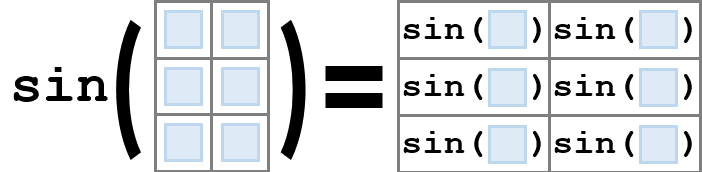
C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\99E42D14.tmp*M*

**6. Mathematical and Statistical Operations with Arrays**

**Summary: Performing Array Operations**

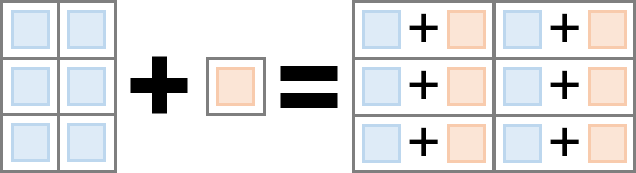
There are many operators that behave in element-wise manner, i.e., the operation is performed on each element of the array individually.

**Mathematical Functions**



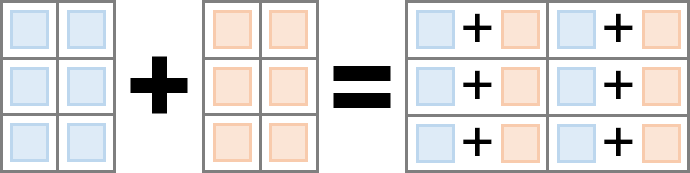
| **Other Similar Functions** | |
| --- | --- |
| sin | Sine |
| cos | Cosine |
| log | Logarithm |
| round | Rounding Operation |
| sqrt | Square Root |
| mod | Modulus |
| Many more | |

**Scalar Expansion**



| **Operators** | |
| --- | --- |
| + | Addition |
| - | Subtraction |
| \* | Multiplication |
| / | Division |
| ^ | Exponentiation |

**Arithmetic Operators**



| **Operators** | |
| --- | --- |
| + | Addition |
| - | Subtraction |
| .\* | Element-wise Multiplication |
| ./ | Element-wise Division |
| .^ | Element-wise Exponentiation |

*Note that, for performing the arithmetic operations on two matrices, they should have identical dimensions.*

**Implicit Expansion**



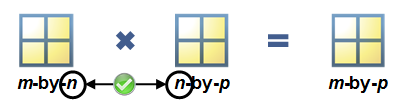
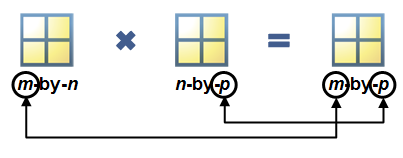
| **Operators** | |
| --- | --- |
| + | Addition |
| - | Subtraction |
| .\* | Element-wise Multiplication |
| ./ | Element-wise Division |
| .^ | Element-wise Exponentiation |

*Array operations can be performed on operands of different compatible sizes. Two arrays have compatible sizes if the size of each dimension is either the same or one.*

**Summary: Matrix Multiplication**

**Matrix Multiplication**

Matrix multiplication requires that the inner dimensions agree. The resultant matrix has the outer dimensions.

**Matrix “Division”**

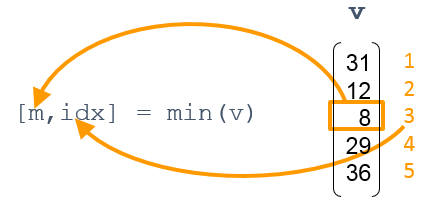
| **Expression** | **Interpretation** |
| --- | --- |
| x = B/A | Solves x\*A = B (for x) |
| x = A\B | Solves A\*x = B (for x) |

**Summary: Calculating Statistics of Vectors**

**Common Statistical Functions**

| **Function** | **Description** |
| --- | --- |
| min | Returns the minimum element |
| max | Returns the maximum element |
| mean | Returns the average of the elements |
| median | Returns the median value of the elements |

**Using min and max**



**Ignoring NaNs**

When using statistical functions, you can ignore NaN values

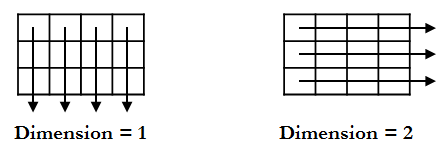
avg = mean(v,'omitnan')

**Summary: Statistical Operations on Matrices**

Some common mathematical functions which calculate a value for each column in a matrix include:

| **Function** | **Behavior** |
| --- | --- |
| max | Largest elements |
| min | Smallest elements |
| mean | Average or mean value |
| median | Median value |
| mode | Most frequent values |
| std | Standard deviation |
| var | Variance |
| sum | Sum of elements |
| prod | Product of elements |

|  |  |  |
| --- | --- | --- |
|  |  | A = [8 2 4 ; 3 2 6 ; 7 5 3 ; 7 10 8]  A =  8 2 4  3 2 6  7 5 3  7 10 8 |
|  |  | Amax = max(A)  Amax =  8 10 8 |
|  |  | Astd = std(A)  Astd =  2.2174 3.7749 2.2174 |
|  |  | Asum = sum(A)  Asum =  25 19 21 |

Many statistical functions accept an optional dimensional argument that specifies whether the operation should be applied to columns independently (the default) or to rows.  


>>

M

= mean(

A

,

dim

)

|  |  |
| --- | --- |
| M | Vector of average values along dimension dim. |
| **Outputs** | |

|  |  |
| --- | --- |
| A | Matrix |
| dim | Dimension across which the mean is taken. 1: the mean of each column 2: the mean of each row |
| **Inputs** | |

**7. Visualizing Data in 2D and 3D**

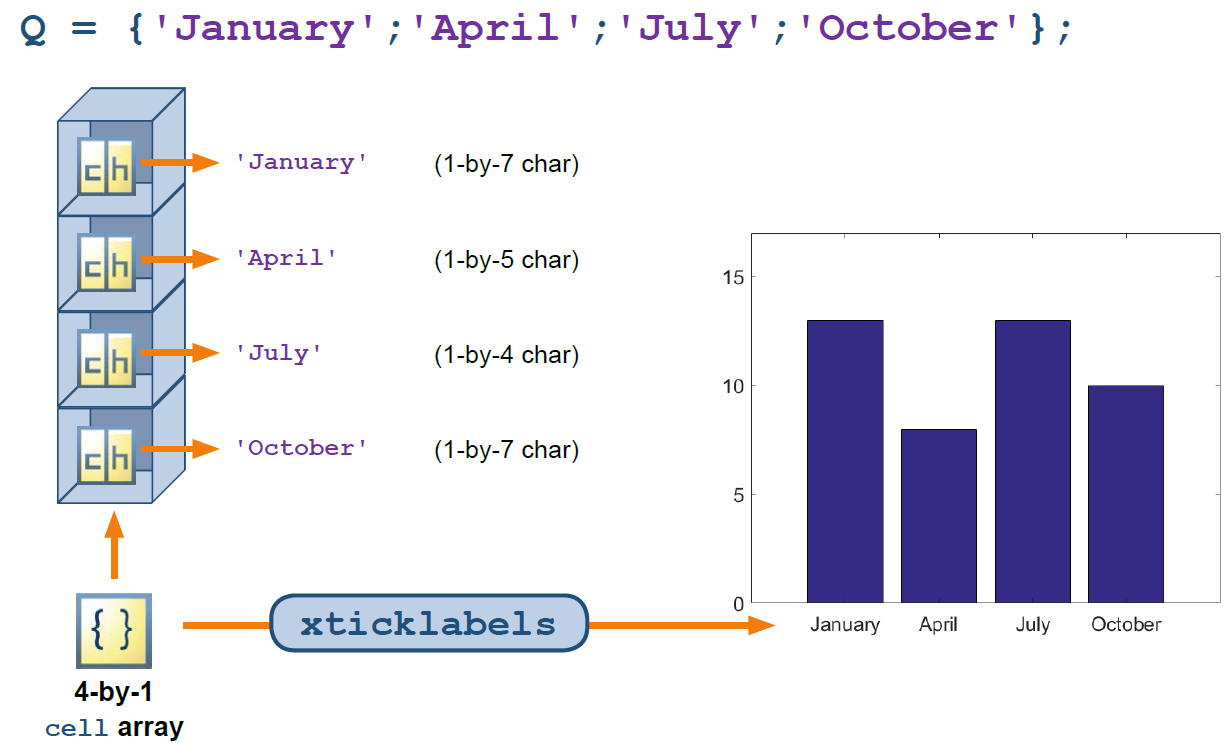
**Summary: Identifying Available Vector Plot Types**

| **Function** | **Description** |
| --- | --- |
| scatter | Scatter plot, with variable marker size and color |
| bar | Bar graph (vertical and horizontal) |
| stem | Discrete sequence (signal) plot |
| stairs | Stairstep graph |
| area | Filled area plot |
| pie | Pie chart |
| histogram | Histogram |

See the complete list of all available plots [here](http://www.mathworks.com/help/matlab/creating_plots/types-of-matlab-plots.html).

**Summary: Annotations Using Arrays of Text**

Cell arrays of text are useful for annotating visualizations. Use curly braces, {}, to create a cell array.



[xticks](http://www.mathworks.com/help/matlab/ref/xticks.html)

*Sets tick locations along the x-axis.*

[xticklabels](http://www.mathworks.com/help/matlab/ref/xticklabels.html)

*Labels the x-axis ticks.*

[xtickangle](http://www.mathworks.com/help/matlab/ref/xtickangle.html)

*Rotates the x-axis tick labels.*

**Summary: Customizing Plot Properties**

**Specifying Property Values**

plot(x,y,linespec,*Property1*,*Value1*,*Property2*,*Value2*,*Property3*,*Value3*,...)

Common line properties to modify:

* 'LineWidth' (width of the line and marker edges)
* 'MarkerSize' (size of the marker symbols)
* 'MarkerEdgeColor' (color of the edge of the marker symbols)
* 'MarkerFaceColor' (color of the interior of the marker symbols)
* 'Color' (color of the line, particularly when given as RGB values)

[MATLAB Line Properties reference](https://www.mathworks.com/help/matlab/ref/matlab.graphics.primitive.line-properties.html)

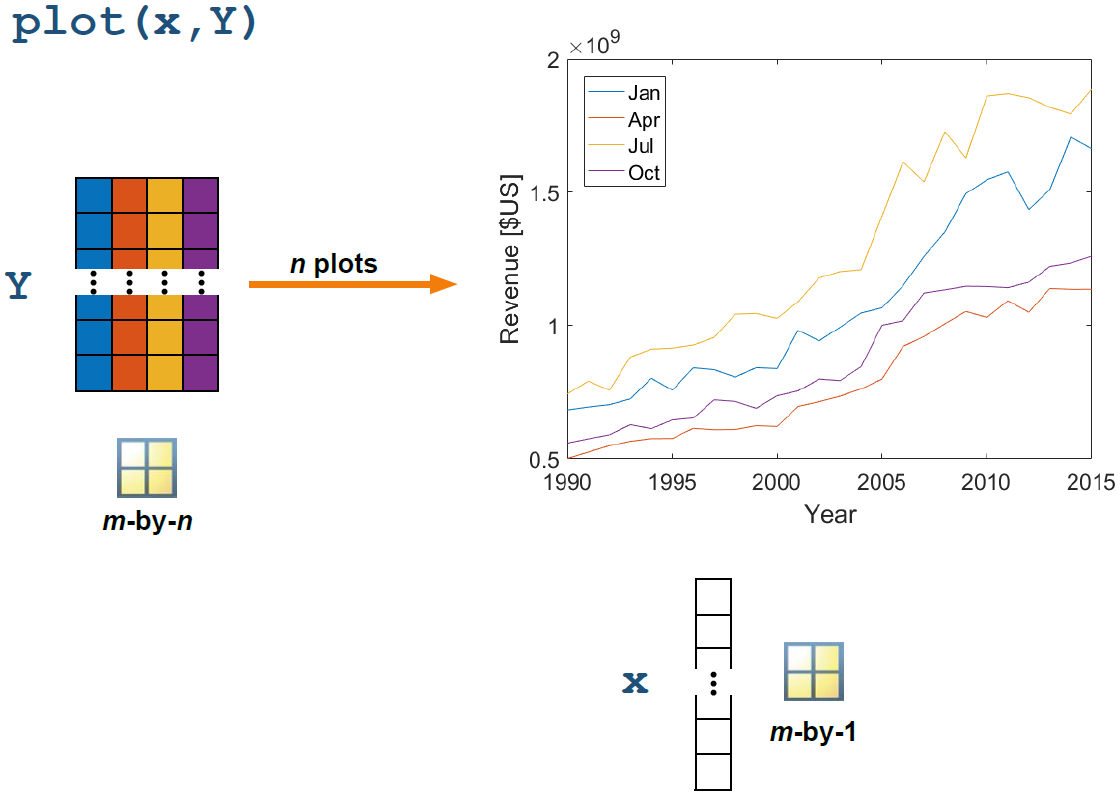
**Specifying Colors**

|  |  |  |  |
| --- | --- | --- | --- |
| red ('r') | green ('g') | blue ('b') | black ('k') |
| magenta ('m') | yellow ('y') | cyan ('c') | white ('w') |

Or as a vector [R G B] where each value is from 0 to 1.

**Summary: Plotting Multiple Columns**

You can use the plot function on a matrix to plot each column as a separate line in your plot.



**Summary: Visualizing Matrices**

You can use visualization functions to plot your three-dimensional data.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| z is a 5-by-5 matrix | |  | | --- | |  | |  | |  | | z  z =  0 0 0 0 0  0 0 -6 0 0  0 -3 1 3 0  0 0 8 1 0  0 0 0 0 0 |
| The [surf](http://www.mathworks.com/help/matlab/ref/surf.html) function plots each point z(j,k) over the point *x*=k and *y*=j | |  | | --- | |  | |  | |  | | surf(z)  C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\5C9BB9DE.tmp |
| To specify *x* and *y* coordinates, you can pass them in as vectors. Here,   * The number of elements of x must match the number of columns of z * The number of elements of y must match the number of rows of z | |  | | --- | |  | |  | |  | | x = 11:15;  y = 21:25;  surf(x,y,z)  C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\B442E19C.tmp |

**8. Conditional Data Selection**

**Summary: Logical Operations and Variables**

**Relational Operators**

|  |  |
| --- | --- |
| == | Equal |
| > | Greater than |
| < | Less than |
| >= | Greater than or equal |
| <= | Less than or equal |
| ~= | Not equal |

**Example**

v = [6 7 8 9];

w = [2 4 8 16];

NE = v ~= w

NE =

1 1 0 1

**Logical Operators**

|  |  |
| --- | --- |
| & | AND |
| | | OR |
| ~ | NOT |

**Example**

v = [6 7 8 9];

w = [2 4 8 16];

x = 5;

A = (v > x) & (w > x)

A =

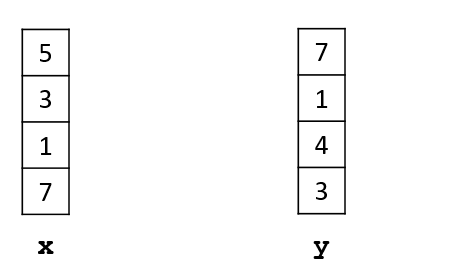
0 0 1 1

**Summary: Counting Elements**

| **Purpose** | **Function** | **Output** |
| --- | --- | --- |
| Are any of the elements true? | [any](http://www.mathworks.com/help/matlab/ref/any.html) | true/false |
| Are all the elements true? | [all](http://www.mathworks.com/help/matlab/ref/all.html) | true/false |
| How many elements are true? | [nnz](http://www.mathworks.com/help/matlab/ref/nnz.html) | double |
| What are the indices of the elements that are true? | [find](http://www.mathworks.com/help/matlab/ref/find.html) | double |

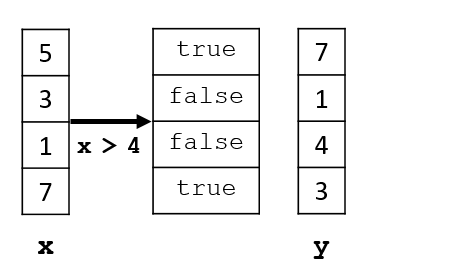
**Summary: Logical Indexing**

**Purpose:** Select the elements of an array based on certain criteria.



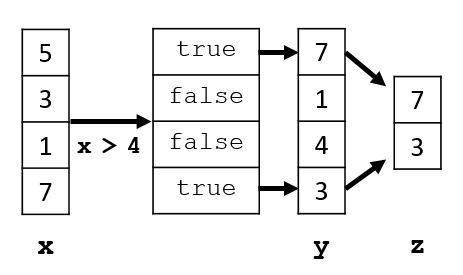
**Step 1:** Create a logical vector by evaluating the given condition.  
  
**Example:**

idx = x > 4



**Step 2:** Use the logical vector as an index into another array to extract the elements corresponding to the true values.  
  
**Example:**

|  |  |  |
| --- | --- | --- |
| idx = x > 4 z = y(idx) | or | z = y(x > 4) |



**9. Review Project I**

**10. Tables of Data**

**Summary: Storing Data in a Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| The [readtable](http://www.mathworks.com/help/matlab/ref/readtable.html) function creates a table in MATLAB from a data file. | |  | | --- | |  | |  | |  | | EPL = readtable('EPLresults.xlsx'); |
| The [table](http://www.mathworks.com/help/matlab/ref/table.html) function can create a table from workspace variables. | |  | | --- | |  | |  | |  | | teamWinsTable = table(team,wins)  teamWins =  **team** **wins**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_  'Arsenal' 20  'Chelsea' 12  'Leicester City' 23  'Manchester United' 19 |
| The [array2table](http://www.mathworks.com/help/matlab/ref/array2table.html) function can convert a numeric array to a table. The [VariableNames](http://www.mathworks.com/help/matlab/ref/array2table.html#input_argument_namevalue_variablenames) property can be specified as a cell array of names to include as variable names in the table. | |  | | --- | |  | |  | |  | | stats = array2table(wdl, ...  'VariableNames',{'Wins','Draws','Losses'})  stats =  **Wins** **Draws** **Losses**  \_\_\_\_ \_\_\_\_\_ \_\_\_\_\_\_  20 11 7  12 14 12  23 12 3  19 9 10 |

**Summary: Sorting Rows of Table Data**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| The [sortrows](http://www.mathworks.com/help/matlab/ref/sortrows.html) function sorts the data in ascending order, by default. | |  | | --- | |  | |  | |  | | EPL = sortrows(EPL,'HomeWins'); |
| Use the optional 'descend' parameter to sort the list in descending order. | |  | | --- | |  | |  | |  | | EPL = sortrows(EPL,'HomeWins','descend'); |
| You can also sort on multiple variables, in order, by specifying a cell array of variable names. | |  | | --- | |  | |  | |  | | EPL = sortrows(EPL,{'HomeWins','AwayWins'},'descend'); |
| You can also show [summary](http://www.mathworks.com/help/matlab/ref/summary.html) statistics for variables in a table. | |  | | --- | |  | |  | |  | | summary(EPL) |

**Summary: Extracting Portions of a Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Display the original table. | |  | | --- | |  | |  | |  | | EPL  EPL =  **Team** **HW** **HD** **HL** **AW** **AD** **AL**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ \_\_ \_\_ \_\_ \_\_ \_\_  'Leicester City' 12 6 1 11 6 2  'Arsenal' 12 4 3 8 7 4  'Manchester City' 12 2 5 7 7 5  'Manchester United' 12 5 2 7 4 8  'Chelsea' 5 9 5 7 5 7  'Bournemouth' 5 5 9 6 4 9  'Aston Villa' 2 5 12 1 3 15 |
| Inside parenthesis, specify the row numbers of the observations and column numbers of the table variables you would like to select. | |  | | --- | |  | |  | |  | | EPL(2:4,[1 2 5])  ans =  **Team HW AW**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ \_\_  'Arsenal' 12 8  'Manchester City' 12 7  'Manchester United' 12 7 |
| You may also use the name of the variable for indexing.  If you want to reference more than one variable, use a cell array containing the variable names. | |  | | --- | |  | |  | |  | | EPL(2:4,{'Team','HW','AW'})  ans =  **Team HW AW**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ \_\_  'Arsenal' 12 8  'Manchester City' 12 7  'Manchester United' 12 7 |

**Summary: Extracting Data from a Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Display the original table. | |  | | --- | |  | |  | |  | | EPL  EPL =  **Team** **HW** **HD** **HL** **AW** **AD** **AL**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ \_\_ \_\_ \_\_ \_\_ \_\_  'Leicester City' 12 6 1 11 6 2  'Arsenal' 12 4 3 8 7 4  'Manchester City' 12 2 5 7 7 5  'Manchester United' 12 5 2 7 4 8 |
| You can use dot notation to extract data for use in calculations or plotting. | |  | | --- | |  | |  | |  | | tw = EPL.HW + EPL.AW  tw =  23  20  19  19 |
| You can also use dot notation to create new variables in a table. | |  | | --- | |  | |  | |  | | EPL.TW = EPL.HW + EPL.AW  EPL =  **Team HW HD HL AW AD AL TW**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ \_\_ \_\_ \_\_ \_\_ \_\_ \_\_  'Leicester City' 12 6 1 11 6 2 23  'Arsenal' 12 4 3 8 7 4 20  'Manchester City' 12 2 5 7 7 5 19  'Manchester United' 12 5 2 7 4 8 19 |
| If you want to extract multiple variables, you can do this using curly braces. | |  | | --- | |  | |  | |  | | draws = EPL{:,{'HD','AD'}}  draws =  6 6  4 7  2 7  5 4 |

**Summary: Exporting Tables**

You can use the [writetable](http://www.mathworks.com/help/matlab/ref/writetable.html) function to create a file from a table.

writetable(*tableName*,'myFile.txt')

The file format is based on the file extension, such as .txt, .csv, or .xlsx.

[writetable](http://www.mathworks.com/help/matlab/ref/writetable.html)

*Write a table to a file.*

**11. Organizing Data**

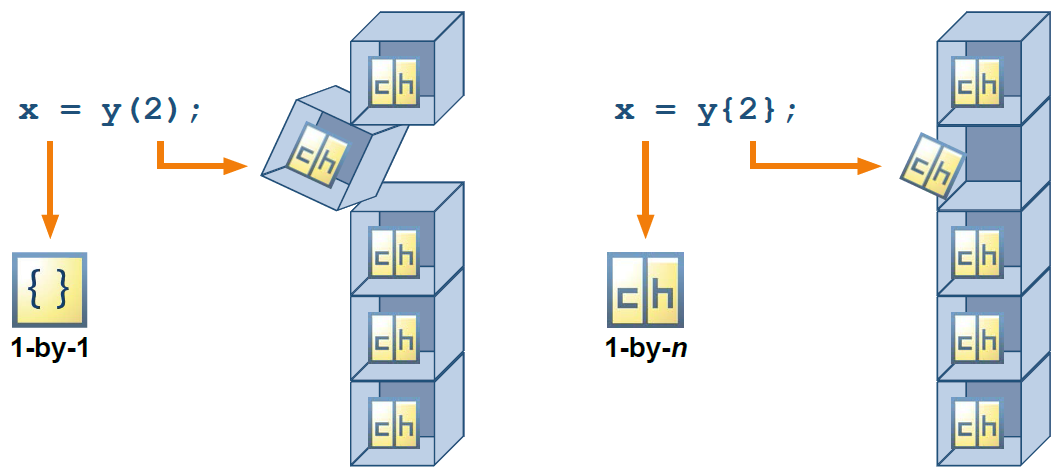
**Summary: Combining Tables**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Display the original tables. | |  | | --- | |  | |  | |  | | EPL  EPL =  **Team Points**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_**  'Leicester City' 81  'Arsenal' 71  'Aston Villa' 17  games  games =  **Wins**  **\_\_\_\_**  23  20  3  teamInfo  teamInfo =  **Team Manager**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  'Arsenal' 'Arsène Wenger'  'Aston Villa' 'Eric Black'  'Leicester City' 'Claudio Ranieri' |
| You can concatenate tables that are the same length but do not share a common variable. | |  | | --- | |  | |  | |  | | [EPL games]  ans =  **Team Points Wins**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_**  'Leicester City' 81 23  'Arsenal' 71 20  'Aston Villa' 17 3 |
| The [join](http://www.mathworks.com/help/matlab/ref/join.html) function can combine tables with a common variable. | |  | | --- | |  | |  | |  | | EPL = join(EPL,teamInfo)  EPL =  **Team Points Manager**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  'Leicester City' 81 'Claudio Ranieri'  'Arsenal' 71 'Arsène Wenger'  'Aston Villa' 17 'Eric Black' |

**Summary: Table Properties**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Display the table properties. | |  | | --- | |  | |  | |  | | EPL.Properties  ans =  **Table Properties** with properties:  Description: ''  UserData: []  DimensionNames: {'Row' 'Variable'}  VariableNames: {1×11 **cell**}  VariableDescriptions: {1×11 **cell**}  VariableUnits: {}  VariableContinuity: []  RowNames: {}  CustomProperties: No custom properties are set. |
| You can access an individual property of Properties using dot notation. | |  | | --- | |  | |  | |  | | EPL.Properties.VariableNames  ans =  1×11 **cell** array  Columns 1 through 4  {'Team'} {'HomeWins'} {'HomeDraws'} {'HomeLosses'}  Columns 5 through 8  {'HomeGF'} {'HomeGA'} {'AwayWins'} {'AwayDraws'}  Columns 9 through 11  {'AwayLosses'} {'AwayGF'} {'AwayGA'} |

**Summary: Indexing into Cell Arrays**



|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| The variable varNames is a cell array that contains character arrays of different lengths in each cell. | |  | | --- | |  | |  | |  | | varNames = teamInfo.Properties.VariableNames   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 'Team' |  | 'Payroll\_M\_\_' |  | 'Manager' |  | 'ManagerHireDate' | |
| Using parentheses to index produces a cell array, not the character array inside the cell. | |  | | --- | |  | |  | |  | | varName(2)   |  | | --- | | 'Payroll\_M\_\_' | |
| In order to extract the contents inside the cell, you should index using curly braces, { }. | |  | | --- | |  | |  | |  | | varName{2}   |  | | --- | | 'Payroll\_M\_\_' | |
| Using curly braces allows you to rename the variable. | |  | | --- | |  | |  | |  | | varName{2} = 'Payroll'   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 'Team' |  | 'Payroll' |  | 'Manager' |  | 'ManagerHireDate' | |

**Summary: Working with Dates and Times**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Dates are often automatically detected and brought in as datetime arrays. | |  | | --- | |  | |  | |  | | teamInfo  ans =  **Manager ManagerHireDate**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  'Rafael Benítez' 3/11/2016  'Claudio Ranieri' 7/13/2015  'Ronald Koeman' 6/16/2014  'David Unsworth' 5/12/2016  'Slaven Bilić' 6/9/2015 |
| Many functions operate on datetime arrays directly, such as [sortrows](https://www.mathworks.com/help/matlab/ref/sortrows.html). | |  | | --- | |  | |  | |  | | sortrows(teamInfo,'ManagerHireDate')  ans =  **Manager ManagerHireDate**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  'Ronald Koeman' 6/16/2014  'Slaven Bilić' 6/9/2015  'Claudio Ranieri' 7/13/2015  'Rafael Benítez' 3/11/2016  'David Unsworth' 5/12/2016 |
| You can create a [datetime](https://www.mathworks.com/help/matlab/ref/datetime.html) array using numeric inputs. The first input is year, then month, then day. | |  | | --- | |  | |  | |  | | t = datetime(1977,12,13)  t =  13-Dec-1977 |
| To create a vector, you can specify an array as input to the datetime function. | |  | | --- | |  | |  | |  | | ts = datetime([1903;1969],[12;7],[17;20])  ts =  17-Dec-1903  20-Jul-1969 |

**Summary: Operating on Dates and Times**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Create datetime variables to work with. | |  | | --- | |  | |  | |  | | seasonStart = datetime(2015,8,8)  seasonStart =  08-Aug-2015    seasonEnd = datetime(2016,5,17)  seasonEnd =  17-May-2016 |
| Use subtraction to produce a duration variable. | |  | | --- | |  | |  | |  | | seasonLength = seasonEnd - seasonStart  seasonLength =  6792:00:00 |
| Functions such as [years](http://www.mathworks.com/help/matlab/ref/years.html) and [days](http://www.mathworks.com/help/matlab/ref/days.html) can help make better sense of the output. | |  | | --- | |  | |  | |  | | seasonLength = days(seasonLength)  seasonLength =  283 |
| They can also create durations from a numeric value. | |  | | --- | |  | |  | |  | | seconds(5)  ans =  5 seconds |
| Use the [between](http://www.mathworks.com/help/matlab/ref/between.html) function to produce a context-dependent calendarDuration variable. | |  | | --- | |  | |  | |  | | seasonLength = between(seasonStart,seasonEnd)  seasonLength =  9mo 9d |
| Create a calendar duration from a numeric input with functions such as [calmonths](http://www.mathworks.com/help/matlab/ref/calmonths.html) and [calyears](https://www.mathworks.com/help/matlab/ref/calendarduration.calyears.html). | |  | | --- | |  | |  | |  | | calmonths(2)  ans =  2mo |

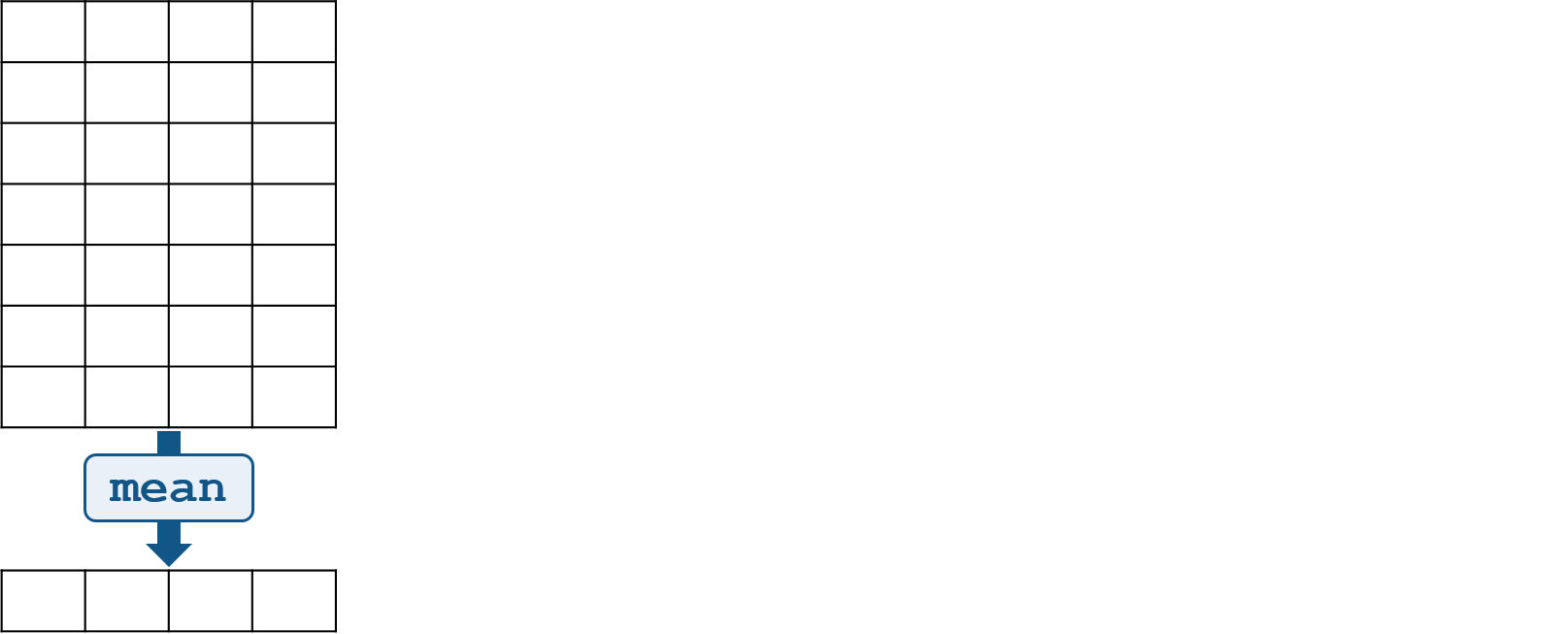
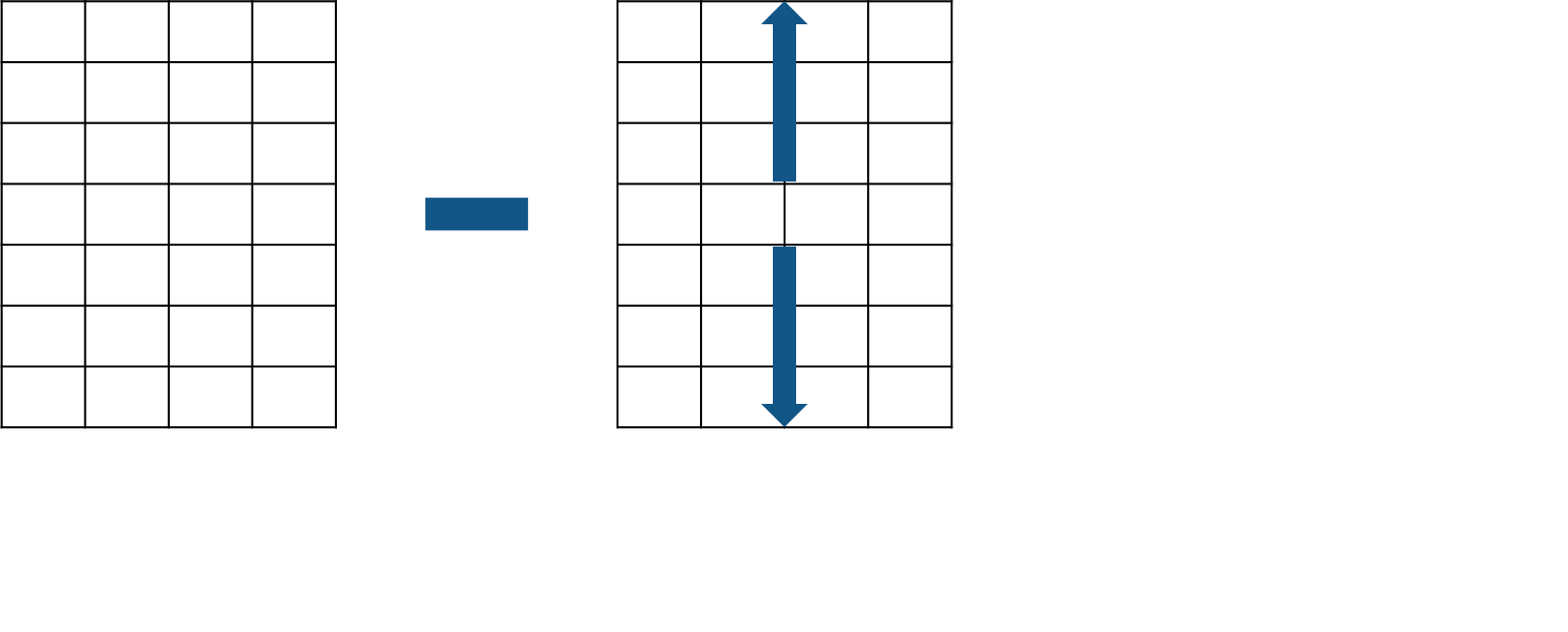
You can learn more about datetime and duration functions in the documentation.  
[Create Date and Time Arrays](https://www.mathworks.com/help/matlab/matlab_prog/represent-date-and-times-in-MATLAB.html)

**Summary: Representing Discrete Categories**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x is a cell array. | |  | | --- | |  | |  | |  | | x = {'C','B','C','A','B','A','C'};  C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\BDCFD0A2.tmp |
| You can convert x into a categorical array, y, using the [categorical](http://www.mathworks.com/help/matlab/ref/categorical.html) function. | |  | | --- | |  | |  | |  | | y = categorical(x);  C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\2EB45900.tmp |
| You can use == to create a logical array, and count elements using nnz. | |  | | --- | |  | |  | |  | | nnz(x == 'C')  ans =  3 |
| You can view catgory statistics using the [summary](http://www.mathworks.com/help/matlab/ref/summary.html) function. | |  | | --- | |  | |  | |  | | summary(y)  **A B C**  2 2 3 |
| You can view combine catgories using the [mergecats](http://www.mathworks.com/help/matlab/ref/mergecats.html) function. | |  | | --- | |  | |  | |  | | y = mergecats(y,{'B','C'},'D')  y =  D D D A D A D |

**12. Preprocessing Data**

**Summary: Normalizing Matrix Data**

1. Apply statistical function to rows or columns of a matrix. The result is a vector.  
     
   
2. Apply array operation to the matrix and the vector. The vector is "expanded" as necessary to make the operation feasible.  
     
   

[normalize](http://www.mathworks.com/help/matlab/ref/normalize.html)

*Normalize data using a specified normalization method.*

**Summary: Working with Missing Data**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data contains missing values, in the form of both -999 and NaN. | |  | | --- | |  | |  | |  | | x = [2 NaN 5 3 -999 4 NaN]; |
| The [ismissing](http://www.mathworks.com/help/matlab/ref/ismissing.html) function identifies only the NaN elements by default. | |  | | --- | |  | |  | |  | | ismissing(x)  ans =  1×7 **logical** array  0 1 0 0 0 0 1 |
| Specifying the set of missing values ensures that ismissing identifies all the missing elements. | |  | | --- | |  | |  | |  | | ismissing(x,[-999,NaN])  ans =  1×7 **logical** array  0 1 0 0 1 0 1 |
| Use the [standardizeMissing](http://www.mathworks.com/help/matlab/ref/standardizemissing.html) function to convert all missing values to NaN. | |  | | --- | |  | |  | |  | | xNaN = standardizeMissing(x,-999)  xNaN =  2 NaN 5 3 NaN 4 NaN |

|  |  |
| --- | --- |
| **Ignores NaNs by default (default flag is 'omitnan')** | **Includes NaNs by default (default flag is 'includenan')** |
| max min | cov mean median std var |

| **Data Type** | **Meaning of "Missing"** |
| --- | --- |
| **double** **single** | NaN |
| **cell** array of **char** | Empty string ('') |
| datetime | NaT |
| duration calendarDuration | NaN |
| categorical | <undefined> |

**Summary: Interpolating Missing Data**

[fillmissing](http://www.mathworks.com/help/matlab/ref/fillmissing.html)

*Fills missing values of an array or table.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Interpolation assuming equal spacing of observations. | |  | | --- | |  | |  | |  | | z = fillmissing(y,'method') |
| Interpolation with given observation locations. | |  | | --- | |  | |  | |  | | z = fillmissing(y,'method','SamplePoints',x) |

| **Method** | **Meaning** |
| --- | --- |
| 'next' | The missing value is the same as the next nonmissing value in the data. |
| 'previous' | The missing value is the same as the previous nonmissing value in the data. |
| 'nearest' | The missing value is the same as the nearest (next or previous) nonmissing value in the data. |
| 'linear' | The missing value is the linear interpolation (average) of the previous and next nonmissing values. |
| 'spline' | Cubic spline interpolation matches the derivatives of the individual interpolants at the data points. This results in an interpolant that is smooth across the whole data set. However, this can also introduce spurious oscillations in the interpolant between data points. |
| 'pchip' | The cubic Hermite interpolating polynomial method forces the interpolant to maintain the same monotonicity as the data. This prevents oscillation between data points. |

**13. Common Data Analysis Techniques**

**Summary: Moving Window Operations**

|  |  |
| --- | --- |
| movmin | Moving minimum |
| movmax | Moving maximum |
| movsum | Moving sum |
| movmean | Moving mean |
| movmedian | Moving median |
| movstd | Moving standard deviation |
| movvar | Moving variance |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Mean calculated with a centered moving *k*-point window. | |  | | --- | |  | |  | |  | | z = movmean(y,k) |
| Mean calculated with a moving window with *kb* points backward and *kf* points forward from the current point. | |  | | --- | |  | |  | |  | | z = movmean(y,[kb kf]) |

**Summary: Linear Correlation**

You can investigate relationships between variables visually and computationally:

* Plot multiple series together. Use [yyaxis](http://www.mathworks.com/help/matlab/ref/yyaxis.html) to add another vertical axis to allow for different scales.
* Plot variables against each other. Use [plotmatrix](http://www.mathworks.com/help/matlab/ref/plotmatrix.html) to create an array of scatter plots.
* Calculate linear correlation coefficients. Use [corrcoef](http://www.mathworks.com/help/matlab/ref/corrcoef.html) to calculate pairwise correlations.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Plot multiple series together. | |  | | --- | |  | |  | |  | | yyaxis left  plot(...)  yyaxis right  plot(...)  C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\DAF08E3A.tmp |
| Plot variables against each other. | |  | | --- | |  | |  | |  | | plotmatrix(data)  C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\4EB42258.tmp |
| Calculate linear correlation coefficients. | |  | | --- | |  | |  | |  | | corrcoef(data)  ans =  1.0000 0.8243 0.1300 0.9519  0.8243 1.0000 0.1590 0.9268  0.1300 0.1590 1.0000 0.2938  0.9519 0.9268 0.2938 1.0000 |

**Summary: Polynomial Fitting**

[polyfit](http://www.mathworks.com/help/matlab/ref/polyfit.html)

*Fits a polynomial to data.*

[polyval](http://www.mathworks.com/help/matlab/ref/polyval.html)

*Evaluates a polynomial at specified locations.*

**Simple fitting**

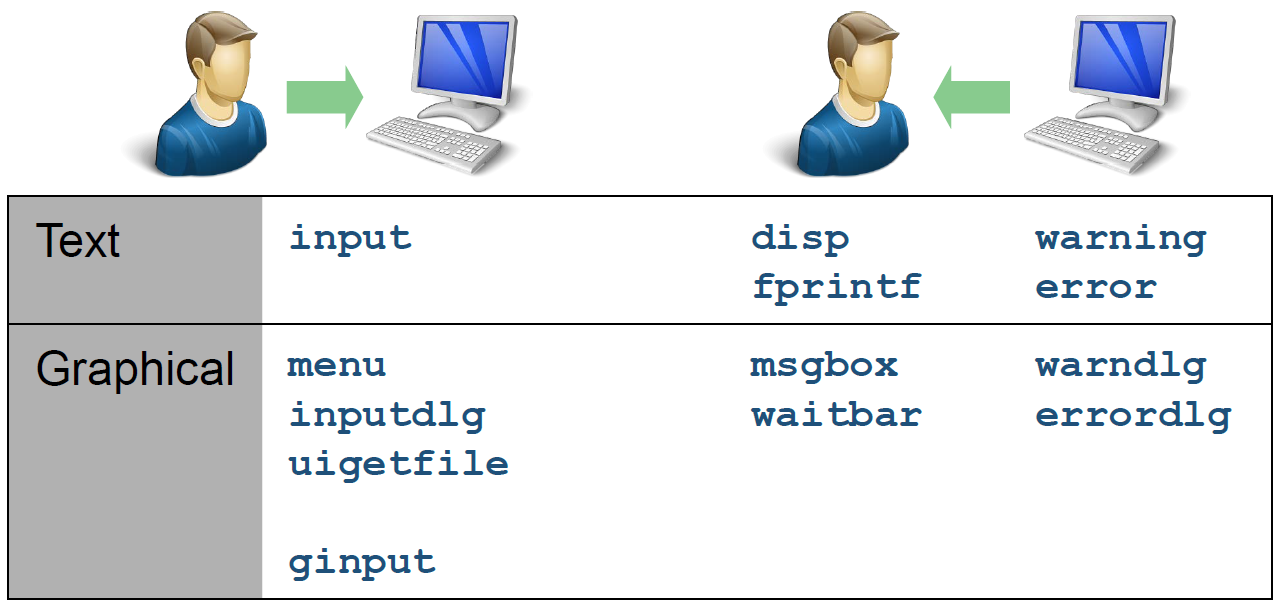
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Fit polynomial to data. | |  | | --- | |  | |  | |  | | c = polyfit(x,y,n); |
| Evaluate fitted polynomial. | |  | | --- | |  | |  | |  | | yfit = polyval(c,xfit); |

**Fitting with centering and scaling**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Fit polynomial to data. | |  | | --- | |  | |  | |  | |  | [c,~,scl] = polyfit(x,y,n); |
| Evaluate fitted polynomial. | |  | | --- | |  | |  | |  | |  | yfit = polyval(c,xfit,[],scl); |

**14. Programming Constructs**

**Summary: User Interaction**



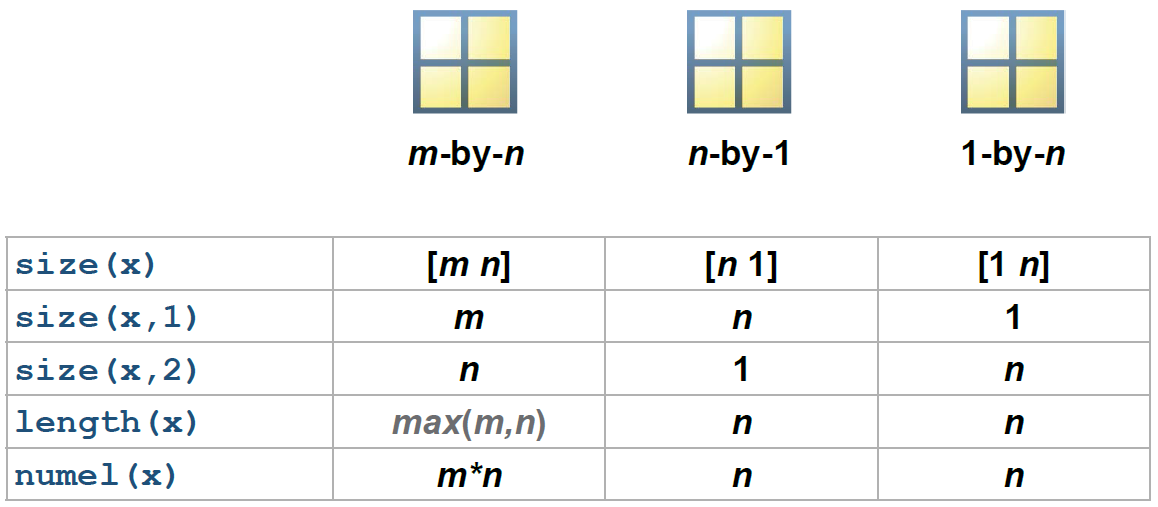
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| You can use [inputdlg](http://www.mathworks.com/help/matlab/ref/inputdlg.html) to gather input from the user. | |  | | --- | |  | |  | |  | | ctry = inputdlg('Enter a country:'); |
| You can use [disp](http://www.mathworks.com/help/matlab/ref/disp.html) to show output on the command window. | |  | | --- | |  | |  | |  | | disp('Message')  Message |
| You can use [warning](http://www.mathworks.com/help/matlab/ref/warning.html) and [error](http://www.mathworks.com/help/matlab/ref/error.html) as well. | |  | | --- | |  | |  | |  | | warning('Missing data')  Warning: Missing data  error('Missing data')  Missing data |
| The [msgbox](http://www.mathworks.com/help/matlab/ref/msgbox.html), [errordlg](http://www.mathworks.com/help/matlab/ref/errordlg.html), and [warndlg](http://www.mathworks.com/help/matlab/ref/warndlg.html) functions can display messages to the user. | |  | | --- | |  | |  | |  | | msgbox('Analysis complete')  C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\3937A924.tmp |

**Summary: Decision Branching**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| The *condition\_1* is evaluated as true or false. | |  | | --- | |  | |  | |  | | if *condition\_1* |
| If *condition\_1* is true, then the *code\_1* code block is executed. | |  | | --- | |  | |  | |  | | *code\_1* |
| Otherwise, the next case is tested. There can be any number of cases. | |  | | --- | |  | |  | |  | | elseif *condition\_2*  *code\_2*  elseif *condition\_3*  *code\_3* |
| If none of the cases are a match, then the code, *code\_e*, in else is executed. | |  | | --- | |  | |  | |  | | else  *code\_e* |
| Always end the expression with the keyword end | |  | | --- | |  | |  | |  | | end |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Evaluate expression to return a value. | |  | | --- | |  | |  | |  | | switch *expression* |
| If expression equals *value\_1*, then *code\_1* is executed. Otherwise, the next case is tested. There can be any number of cases. | |  | | --- | |  | |  | |  | | case *value 1*  *code\_1*  case *value 2*  *code\_2* |
| If none of the cases are a match, then the code, *code\_3*, in otherwise is executed. The otherwise block is optional. | |  | | --- | |  | |  | |  | | otherwise  *code\_3* |
| Always end the expression with the keyword end | |  | | --- | |  | |  | |  | | end |

**Summary: Determining Size**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Use [size](http://www.mathworks.com/help/matlab/ref/size.html) to find the dimensions of a matrix. | |  | | --- | |  | |  | |  | | s = size(prices)  s =  19 10  [m,n] = size(prices)  m =  19  n =  10  m = size(prices,1)  m =  19  n = size(prices,2)  n =  10 |
| Use [length](http://www.mathworks.com/help/matlab/ref/length.html) when working with vectors where one of the dimensions returned by size is 1. | |  | | --- | |  | |  | |  | | m = length(Year)  m =  19 |
| Use [numel](http://www.mathworks.com/help/matlab/ref/numel.html) to find the total number of elements in an array of any dimension. | |  | | --- | |  | |  | |  | | N = numel(prices)  N =  190 |

**Summary: For Loops**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| The index is defined as a vector. Note the use of the colon syntax to define the values that the index will take. | |  | | --- | |  | |  | |  | | for *index* = *first:increment:last*  *code*  end |

**Summary: While Loops**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| The condition is a variable or expression that evaluates to true or false. While *condition* is true, code executes. Once *condition* becomes false, the loop ceases execution. | |  | | --- | |  | |  | |  | | while *condition*  *code*  end |

**15. Increasing Automation with Functions**

**Summary: Creating and Calling Functions**

| **Define a function** | **Call a function** |
| --- | --- |
| C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\9EC516B0.tmp | C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\6ADCA03E.tmp |

**Summary: Function Files**

| **Function Type** | **Function Visibility** |
| --- | --- |
| **Local functions:** Functions that are defined within a script. | Visible only within the file where they are defined. |
| **Functions:** Functions that are defined in separate files. | Visible to other script and function files. |

**Summary: Workspaces**

A function maintains its own workspace to store variables created in the function body.

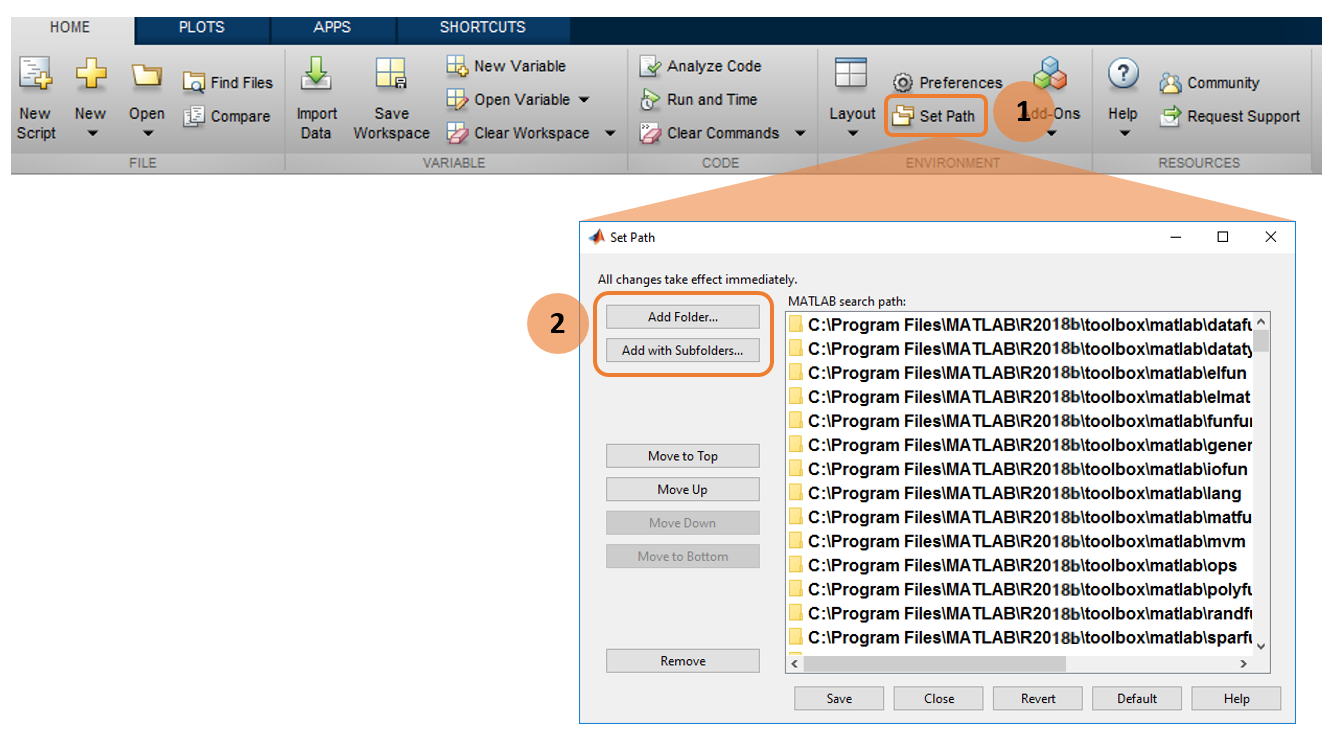
|  |  |  |
| --- | --- | --- |
| a = 42;    b = foo(a); | C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\74E576FC.tmp | foo.mlx   1. function y = foo(x) 2. a = sin(x); 3. x = x + 1; 4. b = sin(x); 5. y = a\*b; 6. end |
| | **Base Workspace** | | | --- | --- | | C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\4D77826A.tmp  a | 42 | | C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\240C9608.tmp  b | 0.7623 | |  | | **Function Workspace** | | | --- | --- | | C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\DC135156.tmp  a | -0.9165 | | C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\BEE4FFD4.tmp  b | -0.8318 | | C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\53967902.tmp  x | 43 | | C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\E85C0060.tmp  y | 0.7623 | |

**Adding Folders to MATLAB Path**

The search path, or *path* is a subset of all the folders in the file system. MATLAB can access all files in the folders on the search path.

To add folders to the search path:

1. On the **Home** tab, in the **Environment** section, click **Set Path**.
2. Add a single folder or a set of folders using the buttons highlighted below.



MATLAB can access the data and code files in the folders on the search path irrespective of the current folder.

**Calling Precedence**

Using functions reduces the possibility of variable name conflicts. However, you can still have conflicts if a user-created function has the same name as a built-in MATLAB function or if a variable has the same name as a function.

Suppose you use a variable named date to store the date of the first observation from the electricity data.

elecData = readtable('elec\_res.csv');

date = elecData.Dates(1)

date =

01-Jan-1990

| **Base Workspace** | |
| --- | --- |
| C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\F46DA3AC.tmp  elecData | *315x2 table* |
| C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\4EC429A.tmp  date | *1x1 datetime* |

However, MATLAB has an in-built function also named date that returns a character vector containing today's date.



*date.m*

Now, if you try to use date, will MATLAB refer to the variable in the Workspace or the function named date?

nextDate = date + 1

?

|  |  |  |  |
| --- | --- | --- | --- |
| C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\A54F7C86.tmp    **date** | or | C:\Users\jiku0755\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\F9A2C284.tmp    *date.m* | ? |

In MATLAB, there are rules for interpreting any named item. These rules are referred to as the function precedence order. Most of the common reference conflicts can be resolved using the following order:

1. Variables
2. Functions defined in the current script
3. Files in the current folder
4. Files on MATLAB search path

A more comprehensive list can be found [here](http://www.mathworks.com/help/matlab/matlab_prog/function-precedence-order.html).

So, in the example shown above, the *variable* date is used instead of the *function* date.

|  |  |  |
| --- | --- | --- |
|  |  | elecData = readtable('elec\_res.csv');  date = elecData.Dates(1);  date =  01-Jan-1990  nextDate = date + 1  nextDate =  02-Jan-1990 |

**16. Troubleshooting Code**

**Summary: Code Analyzer**

Use the MATLAB Code Analyzer messages shown in the Editor to identify and fix syntax errors.

| **Icon** | **Meaning** |
| --- | --- |
|  | There is a potential for unexpected results or poor code performance. |
|  | There are syntax errors that must be addressed. |

**Summary: Debugging Run-Time Errors**

When debugging MATLAB code, a common workflow is as follows.

Note that after you've identified and fixed any bugs, you should stop your debugging session, save your changes, and clear all breakpoints before running your code again.

**17. Review Project II**

**18. Conclusion**