MATLAB FUNDAMENTALS AND PROGRAMMING TECHNIQUES

(BASIC)

Jeffrey Liu

support@terasoft.com.tw

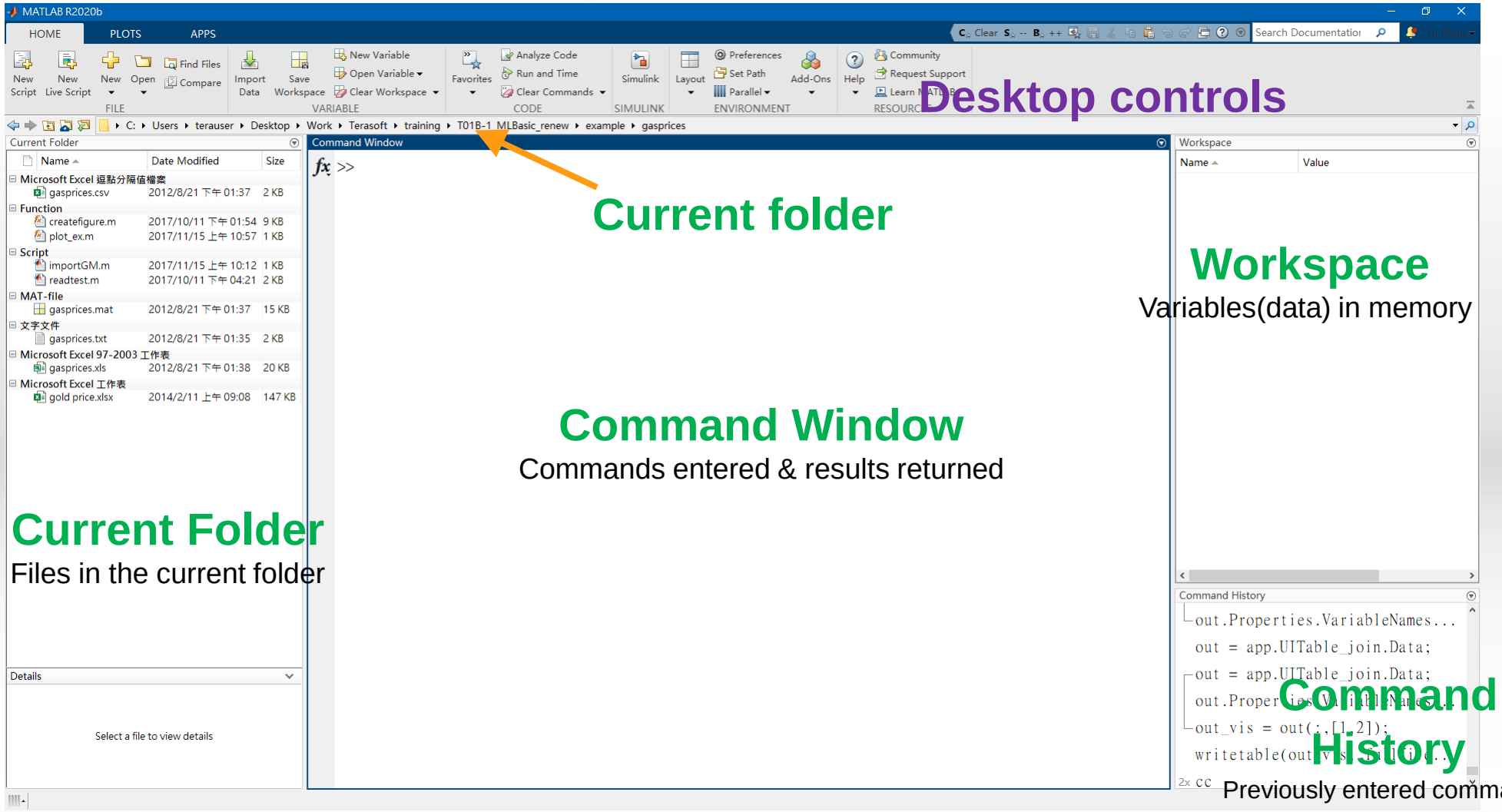
Course Outline

- Working with the MATLAB User Interface
- Variables, Commands, and Plots
- Vector, Matrix, and Arrays
- Table of Data
- Conditional Data Selection
- Analysis with Data
- Increasing Automation with Programming Constructs
- Appendix: Data Types

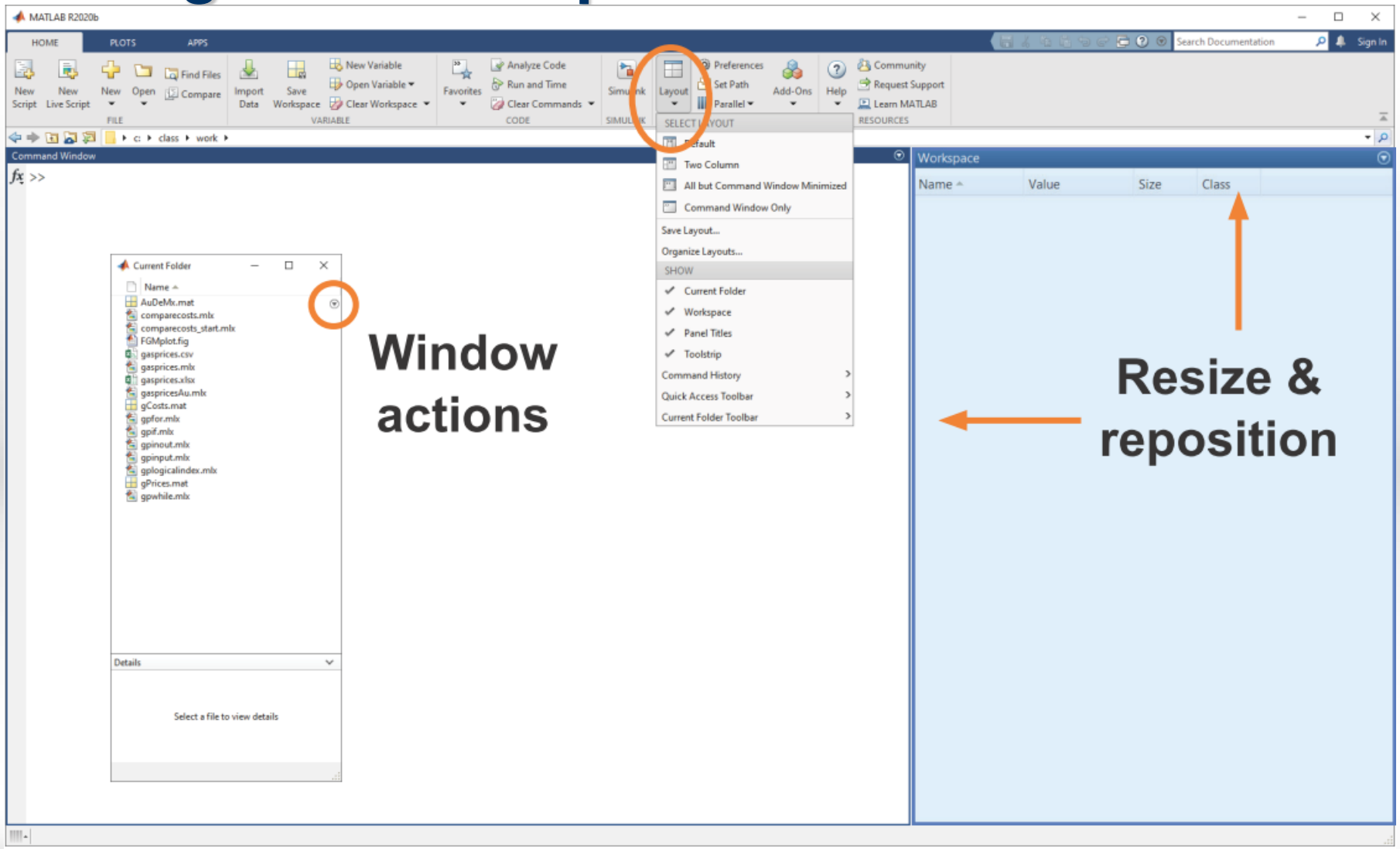
Course Outline

- **Working with the MATLAB User Interface**
- Variables, Commands, and Plots
- Vector, Matrix, and Arrays
- Table of Data
- Conditional Data Selection
- Analysis with Data
- Increasing Automation with Programming Constructs
- Appendix: Data Types

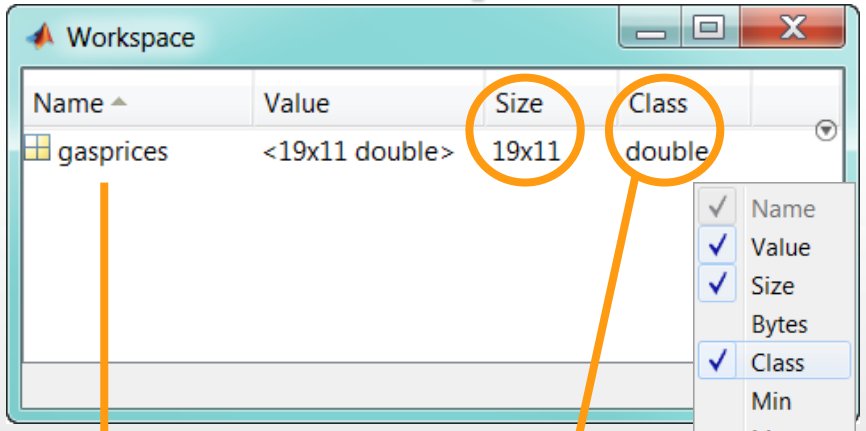
The MATLAB® Desktop



Customizing the Desktop



Variables in the Base Workspace



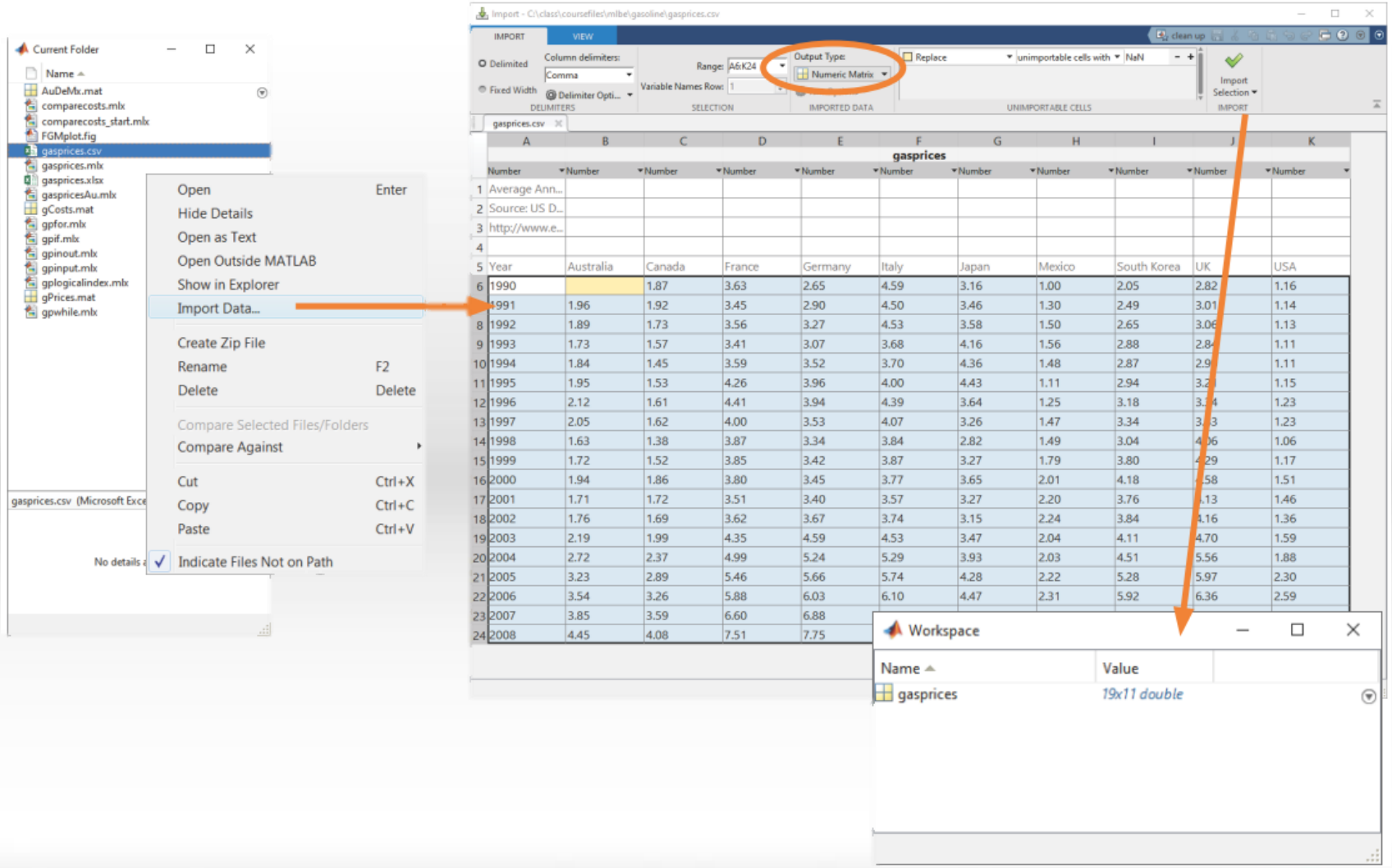
1990	NaN	1.87	3.63	2.65	4.59	3.16	1	2.05	2.82	1.16
1991	1.96	1.92	3.45	2.9	4.5	3.46	1.3	2.49	3.01	1.14
1992	1.89	1.73	3.56	3.27	4.53	3.58	1.5	2.65	3.06	1.13
1993	1.73	1.57	3.41	3.07	3.68	4.16	1.56	2.88	2.84	1.11
1994	1.84	1.45	3.59	3.52	3.7	4.36	1.48	2.87	2.99	1.11
1995	1.95	1.53	4.26	3.96	4	4.43	1.11	2.94	3.21	1.15
1996	2.12	1.61	4.41	3.94	4.39	3.64	1.25	3.18	3.34	1.23
1997	2.05	1.62	4	3.53	4.07	3.26	1.47	3.34	3.83	1.23
1998	1.63	1.38	3.87	3.34	3.84	2.82	1.49	3.04	4.06	1.06
1999	1.72	1.52	3.85	3.42	3.87	3.27	1.79	3.8	4.29	1.17
2000	1.94	1.86	3.8	3.45	3.77	3.65	2.01	4.18	4.58	1.5
2001	1.71	1.72	3.51	3.4	3.57	3.27	2.2	3.76	4.13	1.
2002	1.76	1.69	3.62	3.67	3.74	3.15	2.24	3.84	4.16	1.
2003	2.19	1.99	4.35	4.59	4.53	3.47	2.04	4.11	4.7	1.59
2004	2.72	2.37	4.99	5.24	5.29	3.93	2.03	4.51	5.56	1.88
2005	3.23	2.89	5.46	5.66	5.74	4.28	2.22	5.28	5.97	2.3
2006	3.54	3.26	5.88	6.03	6.1	4.47	2.31	5.92	6.36	2.59
2007	3.85	3.59	6.6	6.88	6.73	4.49	2.4	6.21	7.13	2.8
2008	4.45	4.08	7.51	7.75	7.63	5.74	2.45	5.83	7.42	3.27

19

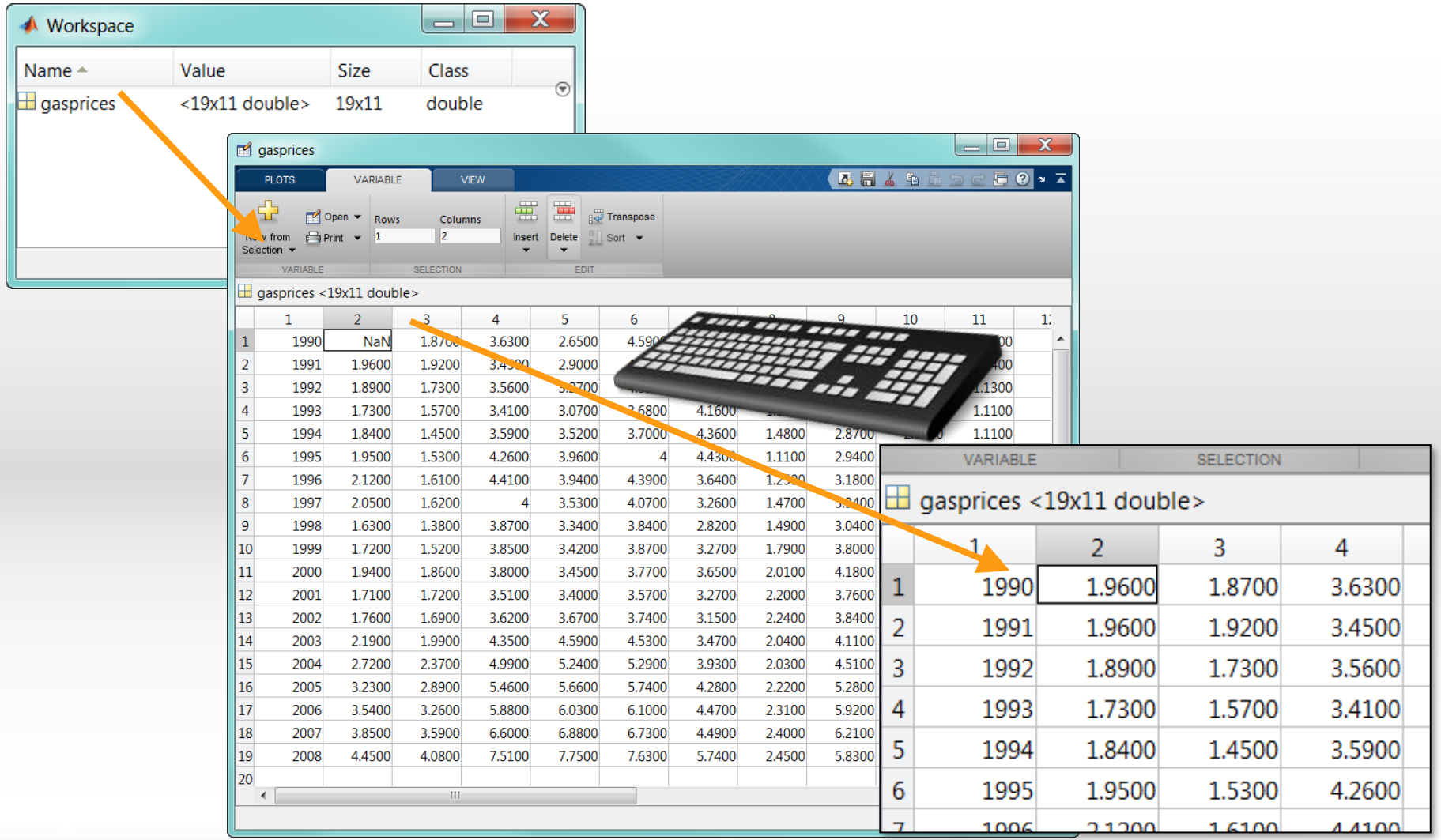
11

numeric data → “**double precision**”

Interactive Importing

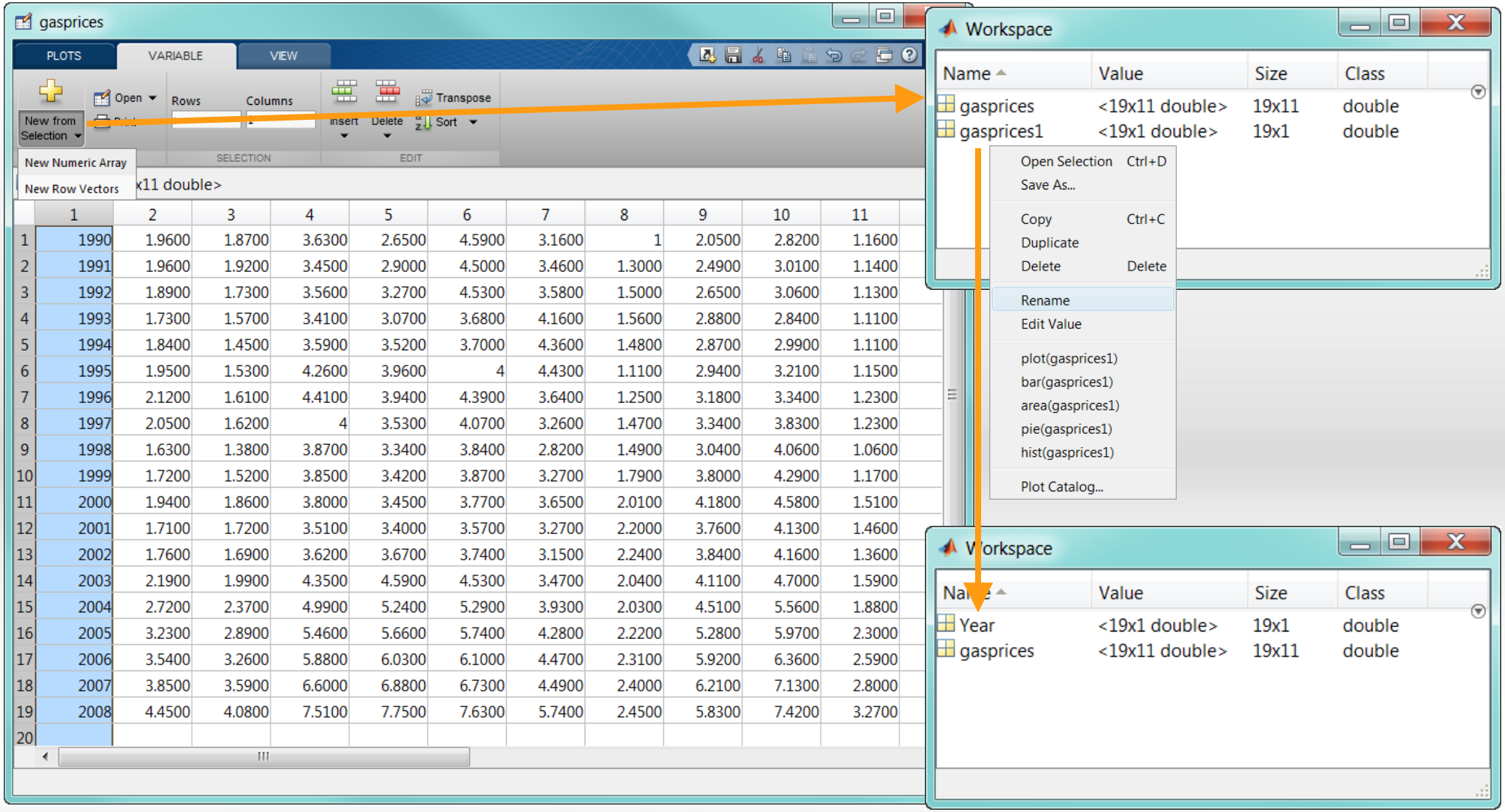


The Variable Editor



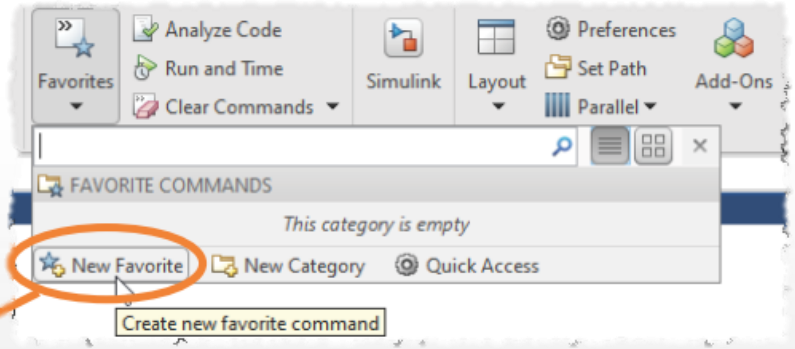
	1	2	3	4
1	1990	1.9600	1.8700	3.6300
2	1991	1.9600	1.9200	3.4500
3	1992	1.8900	1.7300	3.5600
4	1993	1.7300	1.5700	3.4100
5	1994	1.8400	1.4500	3.5900
6	1995	1.9500	1.5300	4.2600

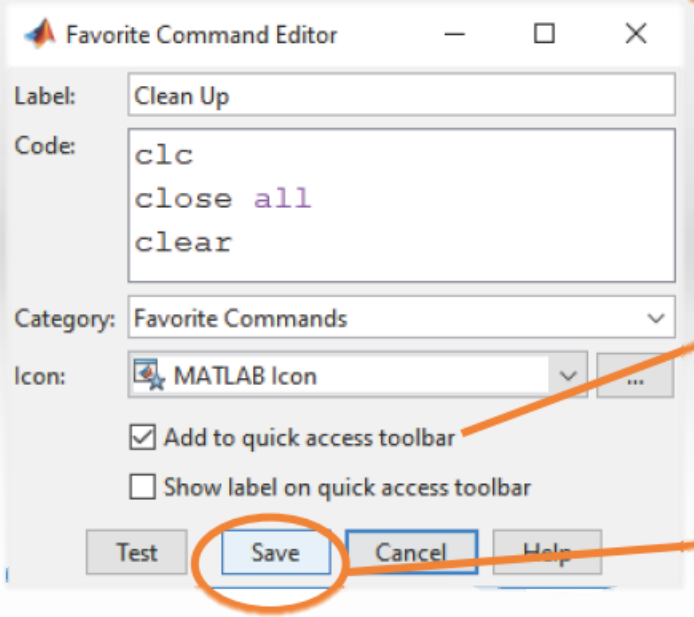
New Variables



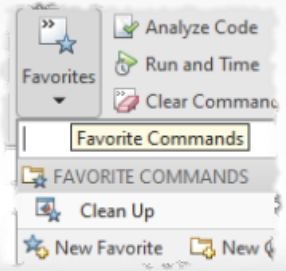
	1	2	3	4	5	6	7	8	9	10	11
1	1990	1.9600	1.8700	3.6300	2.6500	4.5900	3.1600	1	2.0500	2.8200	1.1600
2	1991	1.9600	1.9200	3.4500	2.9000	4.5000	3.4600	1.3000	2.4900	3.0100	1.1400
3	1992	1.8900	1.7300	3.5600	3.2700	4.5300	3.5800	1.5000	2.6500	3.0600	1.1300
4	1993	1.7300	1.5700	3.4100	3.0700	3.6800	4.1600	1.5600	2.8800	2.8400	1.1100
5	1994	1.8400	1.4500	3.5900	3.5200	3.7000	4.3600	1.4800	2.8700	2.9900	1.1100
6	1995	1.9500	1.5300	4.2600	3.9600	4	4.4300	1.1100	2.9400	3.2100	1.1500
7	1996	2.1200	1.6100	4.4100	3.9400	4.3900	3.6400	1.2500	3.1800	3.3400	1.2300
8	1997	2.0500	1.6200	4	3.5300	4.0700	3.2600	1.4700	3.3400	3.8300	1.2300
9	1998	1.6300	1.3800	3.8700	3.3400	3.8400	2.8200	1.4900	3.0400	4.0600	1.0600
10	1999	1.7200	1.5200	3.8500	3.4200	3.8700	3.2700	1.7900	3.8000	4.2900	1.1700
11	2000	1.9400	1.8600	3.8000	3.4500	3.7700	3.6500	2.0100	4.1800	4.5800	1.5100
12	2001	1.7100	1.7200	3.5100	3.4000	3.5700	3.2700	2.2000	3.7600	4.1300	1.4600
13	2002	1.7600	1.6900	3.6200	3.6700	3.7400	3.1500	2.2400	3.8400	4.1600	1.3600
14	2003	2.1900	1.9900	4.3500	4.5900	4.5300	3.4700	2.0400	4.1100	4.7000	1.5900
15	2004	2.7200	2.3700	4.9900	5.2400	5.2900	3.9300	2.0300	4.5100	5.5600	1.8800
16	2005	3.2300	2.8900	5.4600	5.6600	5.7400	4.2800	2.2200	5.2800	5.9700	2.3000
17	2006	3.5400	3.2600	5.8800	6.0300	6.1000	4.4700	2.3100	5.9200	6.3600	2.5900
18	2007	3.8500	3.5900	6.6000	6.8800	6.7300	4.4900	2.4000	6.2100	7.1300	2.8000
19	2008	4.4500	4.0800	7.5100	7.7500	7.6300	5.7400	2.4500	5.8300	7.4200	3.2700
20											

Favorites


Favorites
Analyze Code
Run and Time
Clear Commands
Simulink
Layout
Preferences
Set Path
Parallel
Add-Ons
FAVORITE COMMANDS
This category is empty
New Favorite
New Category
Quick Access
Create new favorite command


Favorite Command Editor
Label:
Clean Up
Code:
clc
close all
clear
Category:
Favorite Commands
Icon:
MATLAB Icon
Add to quick access toolbar
Show label on quick access toolbar
Test
Save
Cancel
Help


Search Documentation


Favorites
Analyze Code
Run and Time
Clear Commanc
Favorite Commands
FAVORITE COMMANDS
Clean Up
New Favorite
New

Chapter 1 Test Your Knowledge

1. Where does MATLAB display a listing of stored variables and associated attributes?

   A. Command Window

   B. Workspace browser

   C. Current Directory browser

   D. Command History

2. T/F: The MATLAB desktop is customizable.

3. The default MATLAB variable type is:

   A. Single

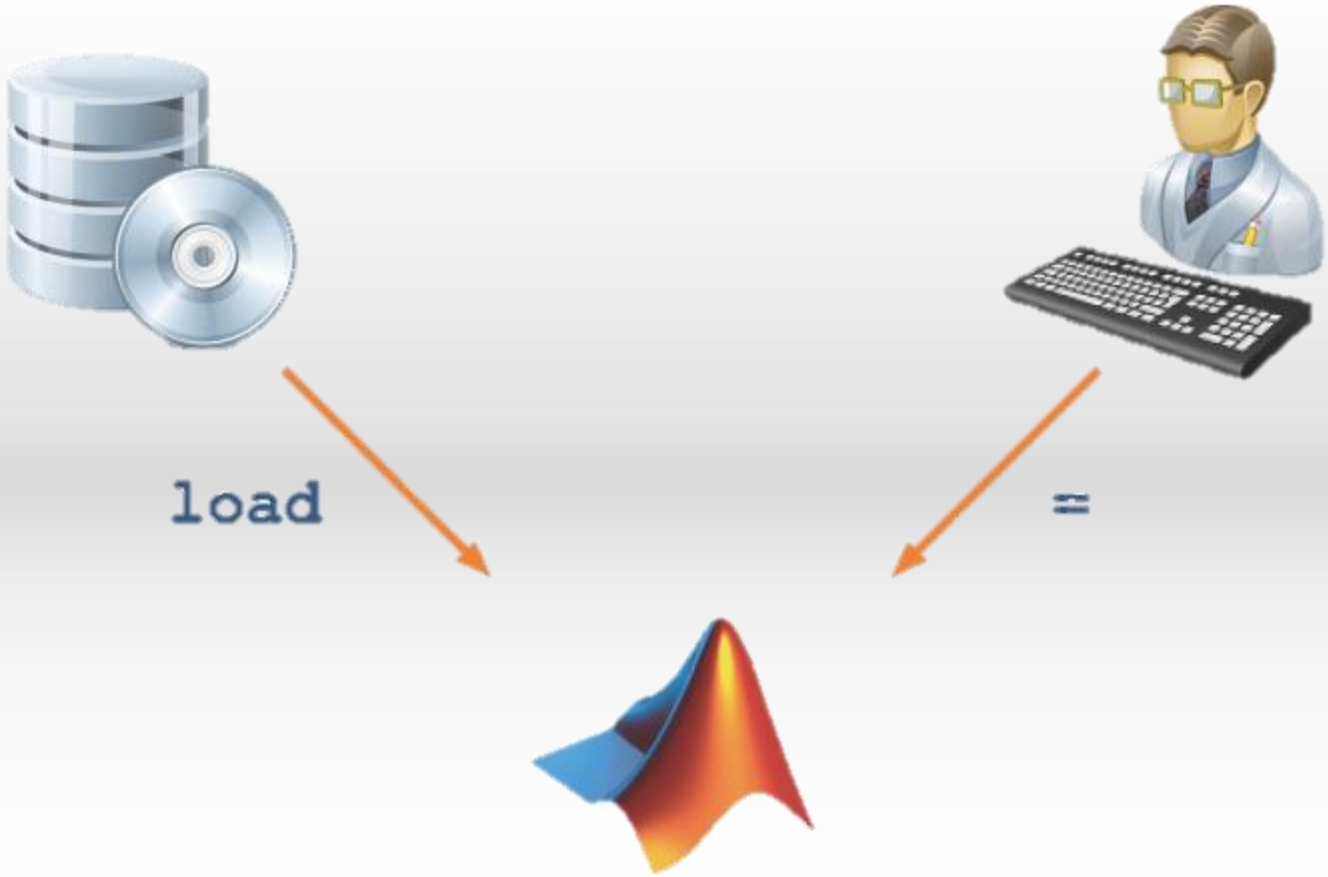
   B. Double

   C. Cell

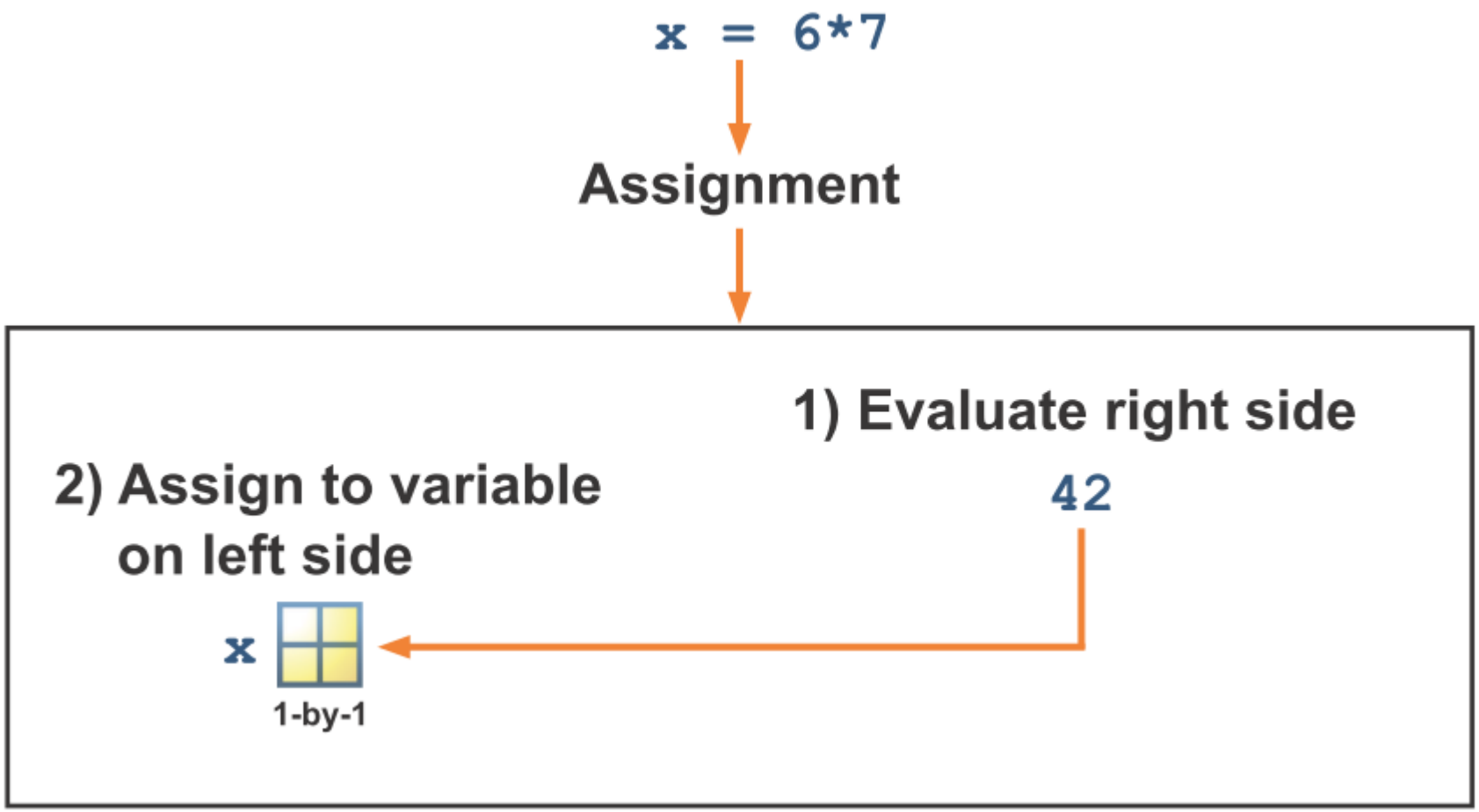
Course Outline

- Working with the MATLAB User Interface
- ➢ **Variables, Commands, and Plots**
- Vector, Matrix, and Arrays
- Table of Data
- Conditional Data Selection
- Analysis with Data
- Increasing Automation with Programming Constructs
- Appendix: Data Types

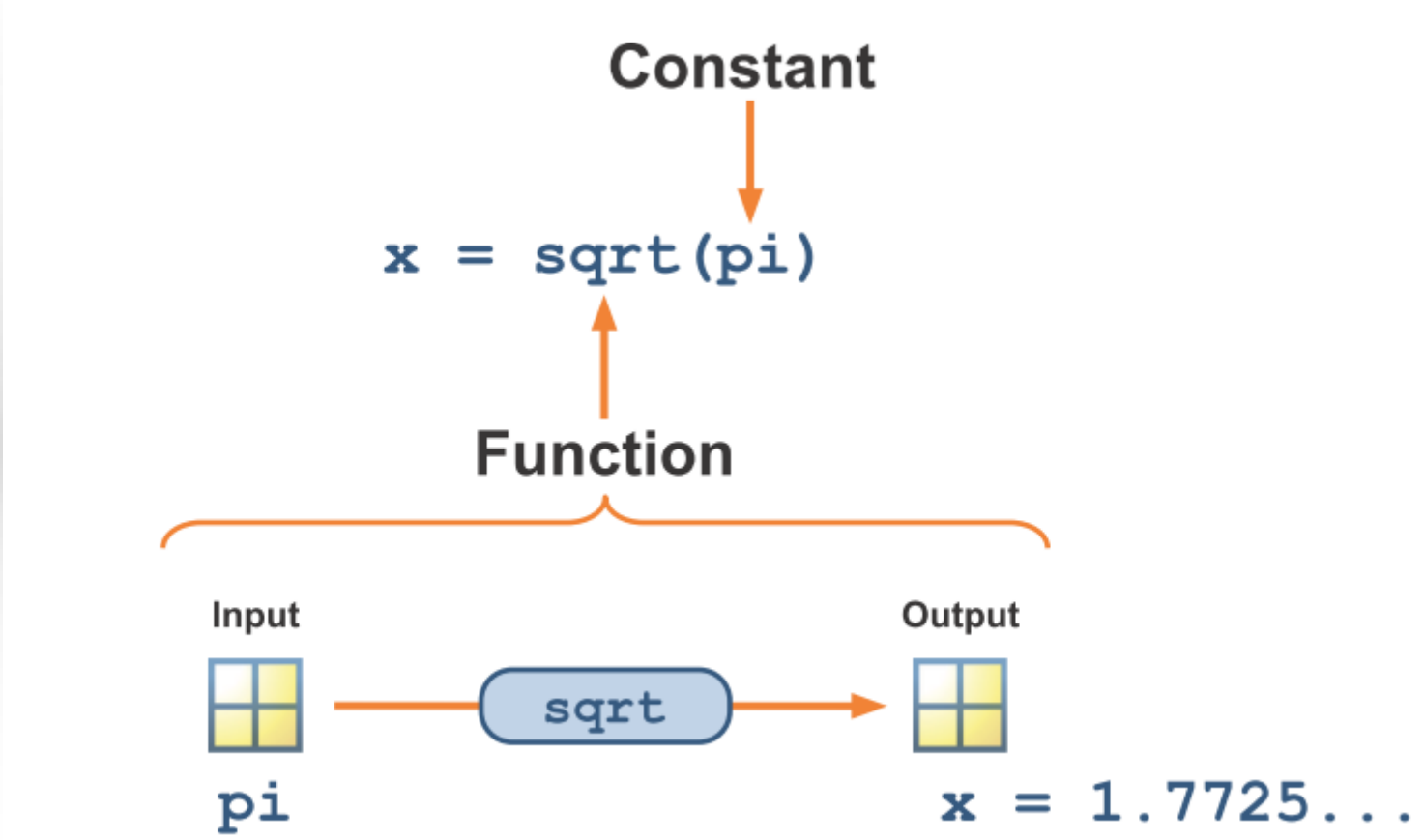
Getting data into MATLAB



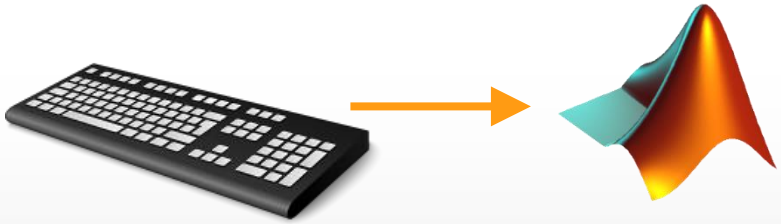
Assigning Values to Variables

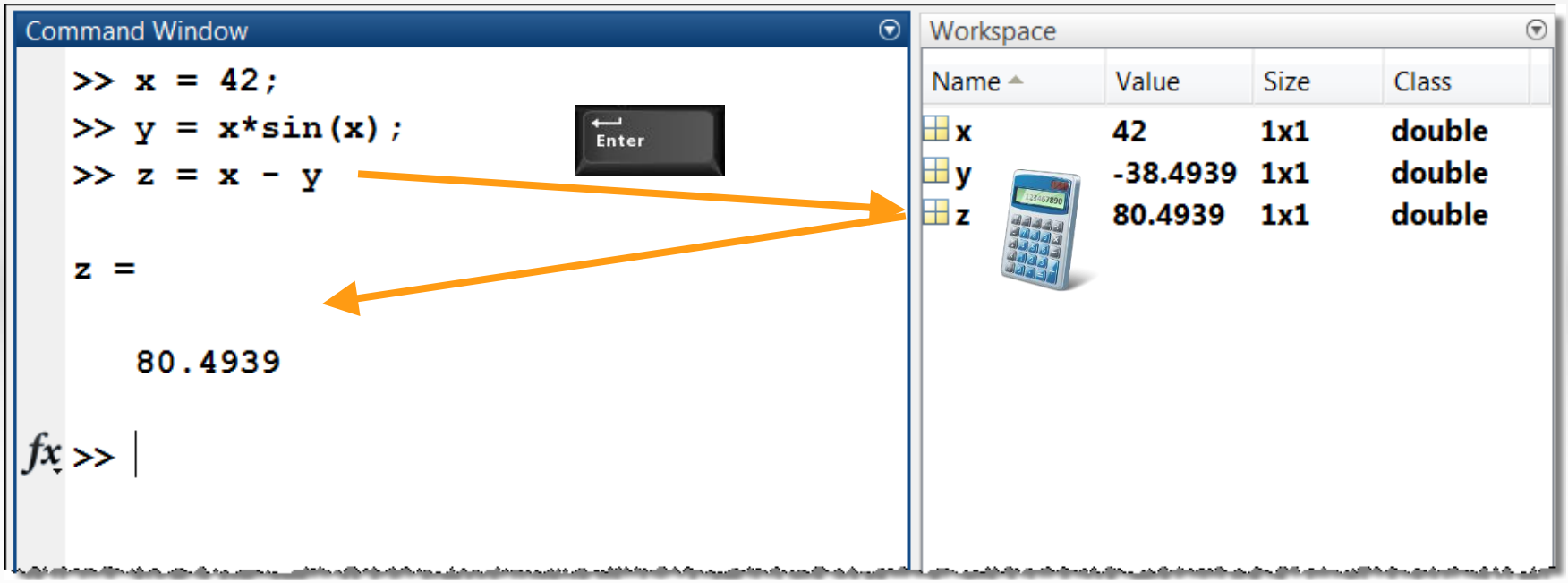


Using Functions and Built-in Constants



MATLAB® Commands





Assignment

Data Type

```
>> no_of_penguins = x*sin(pi*t);
```

1. evaluate right-hand side

2. assign resulting value to variable on left-hand side

create new variable or overwrite old one, as appropriate

Creating Characters and Strings

```
name = US
```
→ variable called US

```
name = 'US'
```
→ characters 'U' 'S'

```
name = "US"
```
→ word "US"

```
country = 'Australia'
```

1	2	3	4	5	6	7	8	9
'A'	'u'	's'	't'	'r'	'a'	'l'	'i'	'a'

ch 1-by-9

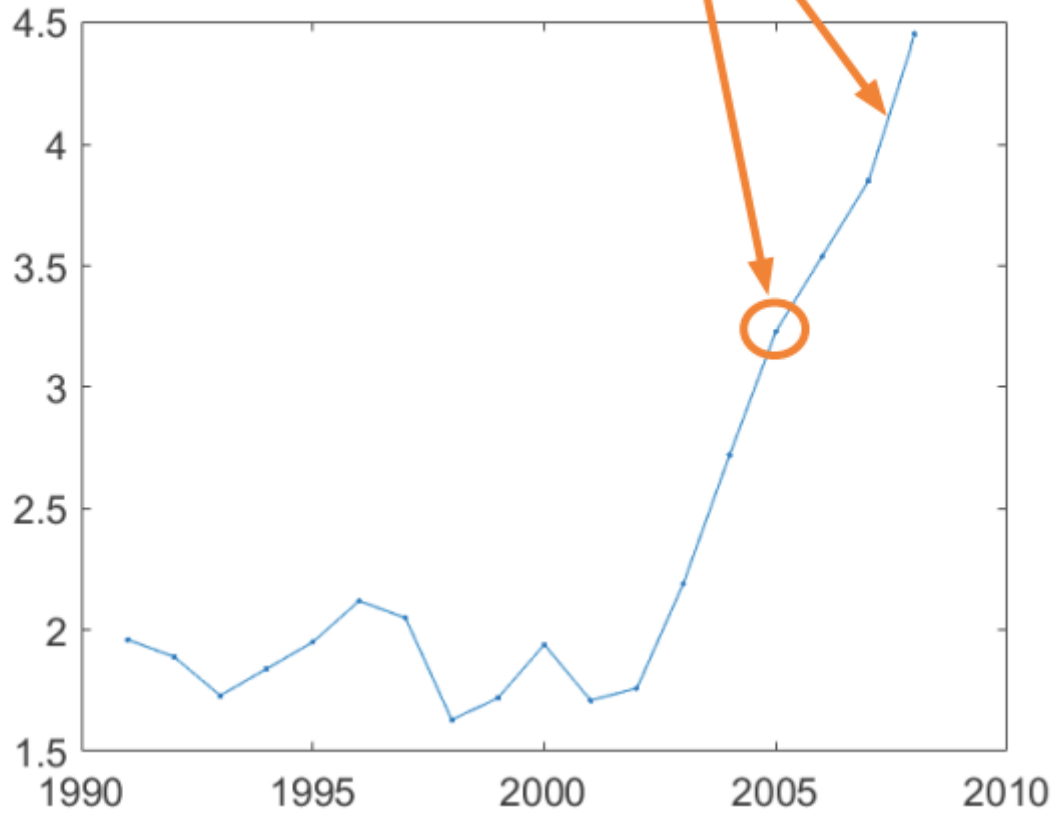
```
country = "Australia"
```

1
"Australia"

str 1-by-1

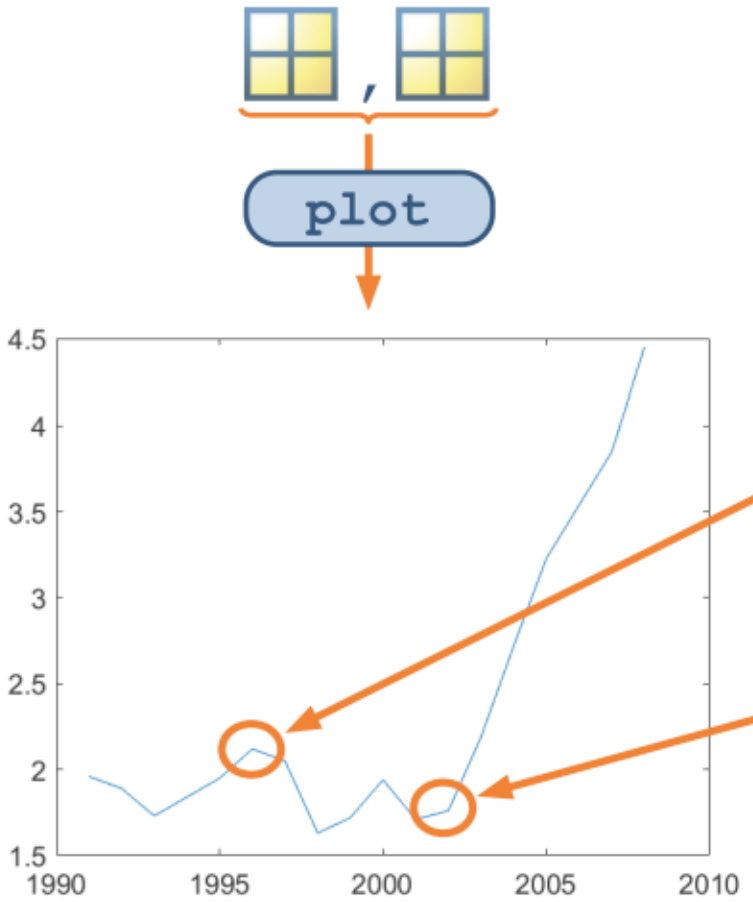
Plot Options

```
plot(Year,Australia,".-")
```



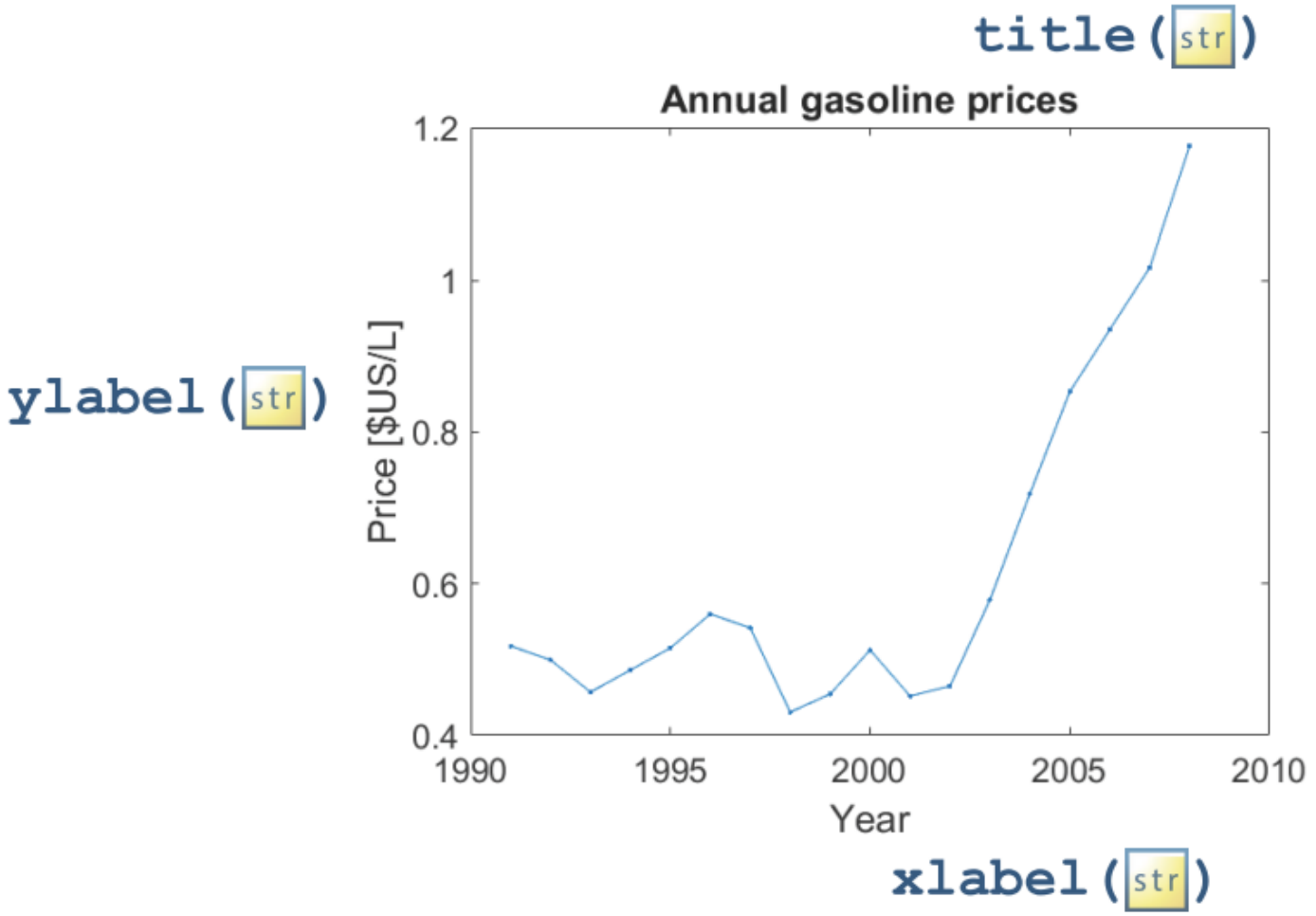
Plotting

```
plot(Year,Australia)
```



1990	NaN
1991	1.9600
1992	1.8900
1993	1.7300
1994	1.8400
1995	1.9500
1996	2.1200
1997	2.0500
1998	1.6300
1999	1.7200
2000	1.9400
2001	1.7100
2002	1.7600
2003	2.1900
2004	2.7200
2005	3.2300
2006	3.5400
2007	3.8500
2008	4.4500

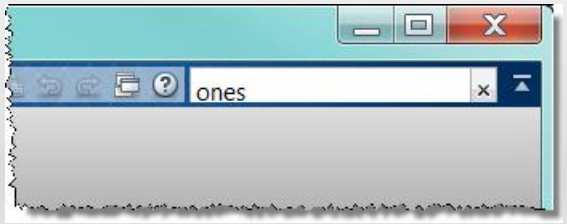
Annotating Plots



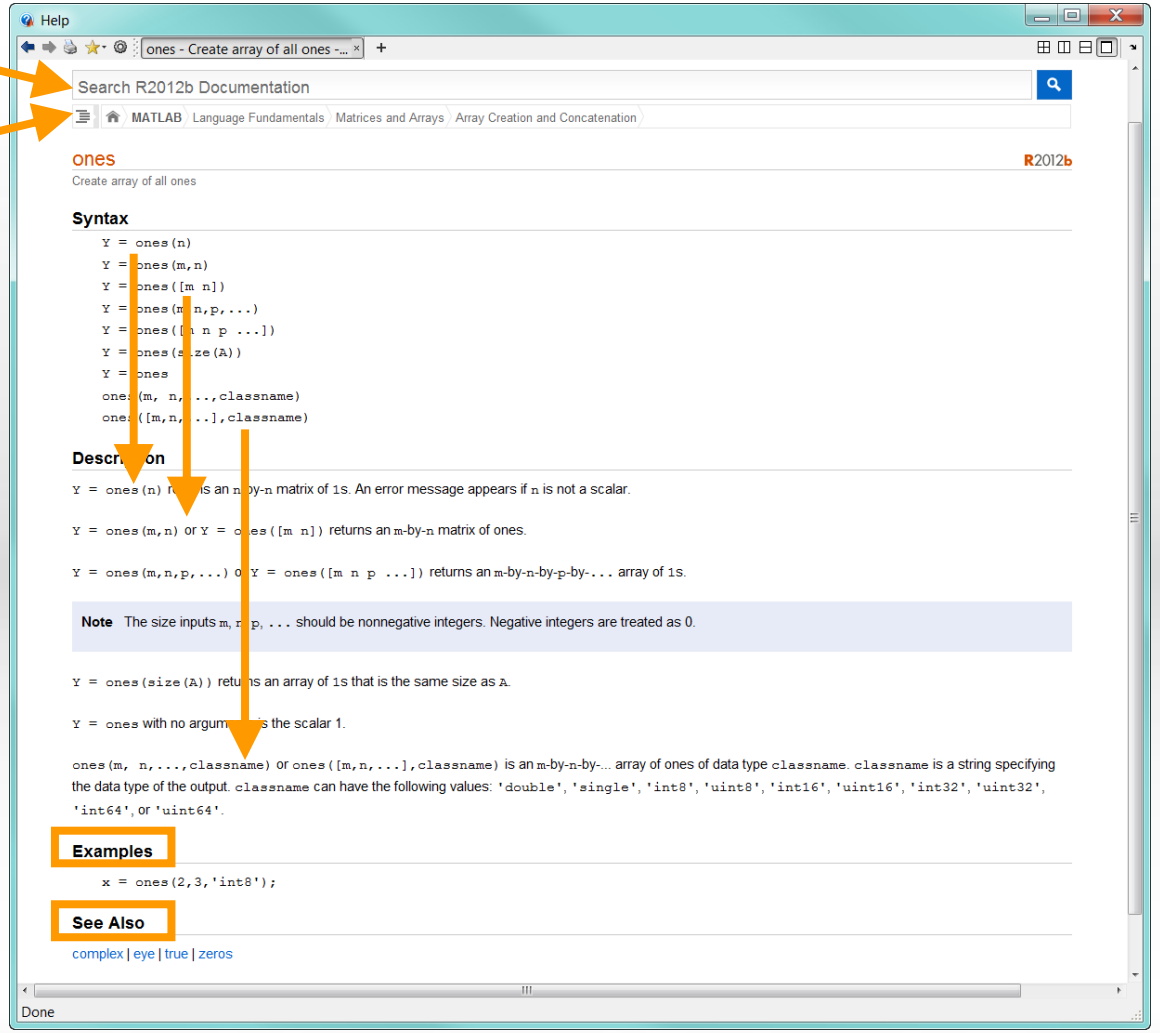
Help and Documentation

search

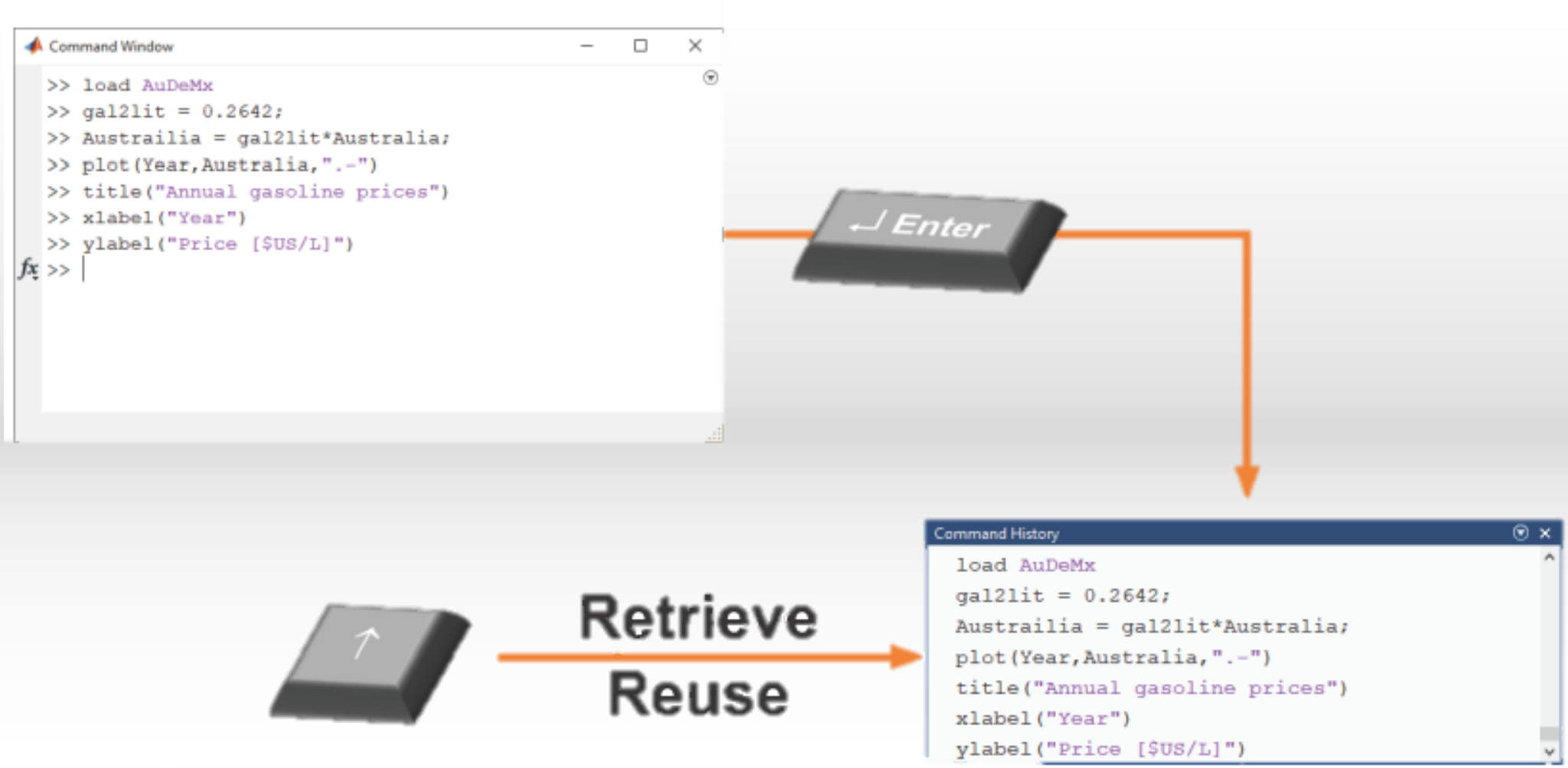
browse



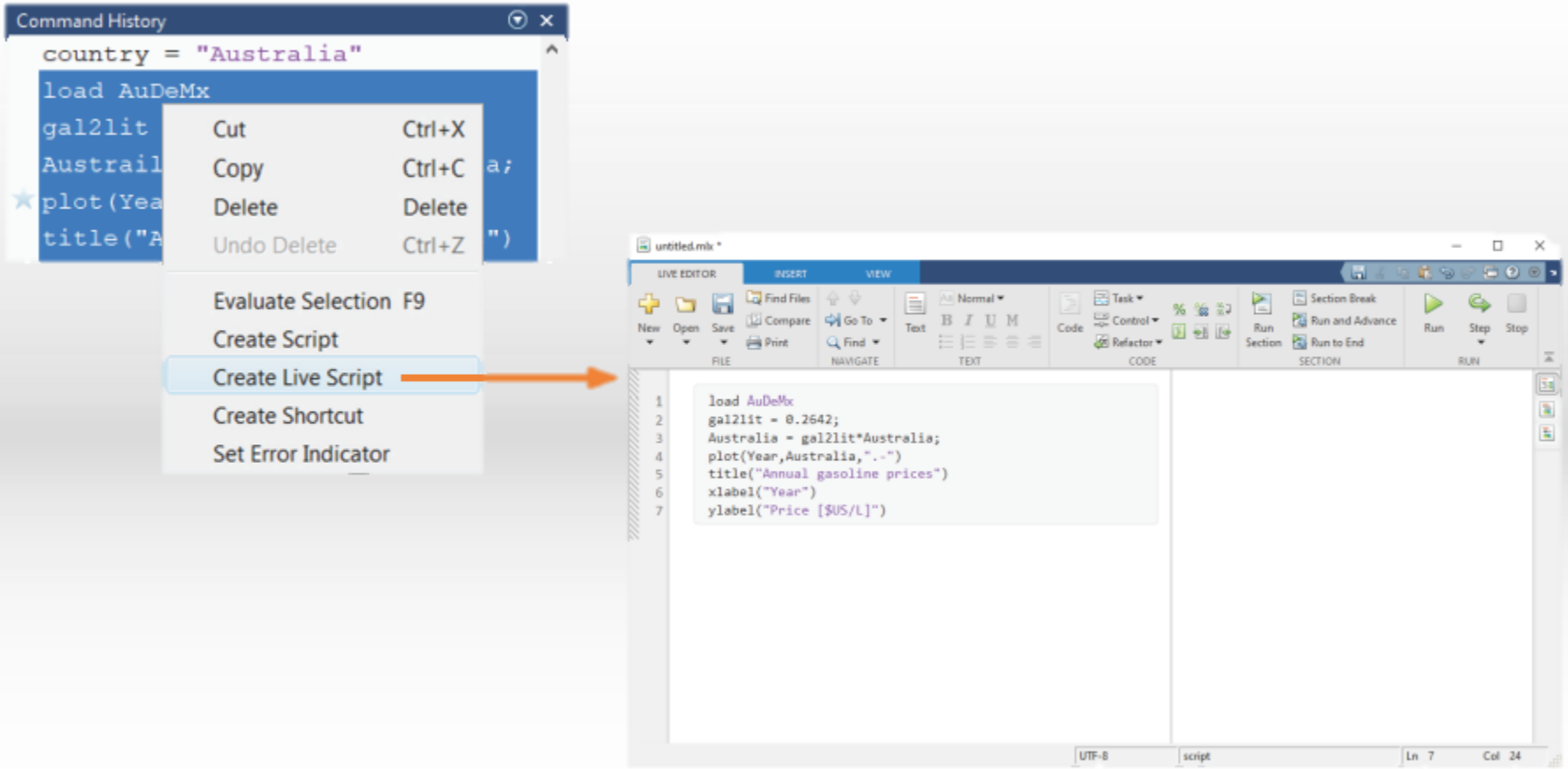
help
doc
docsearch



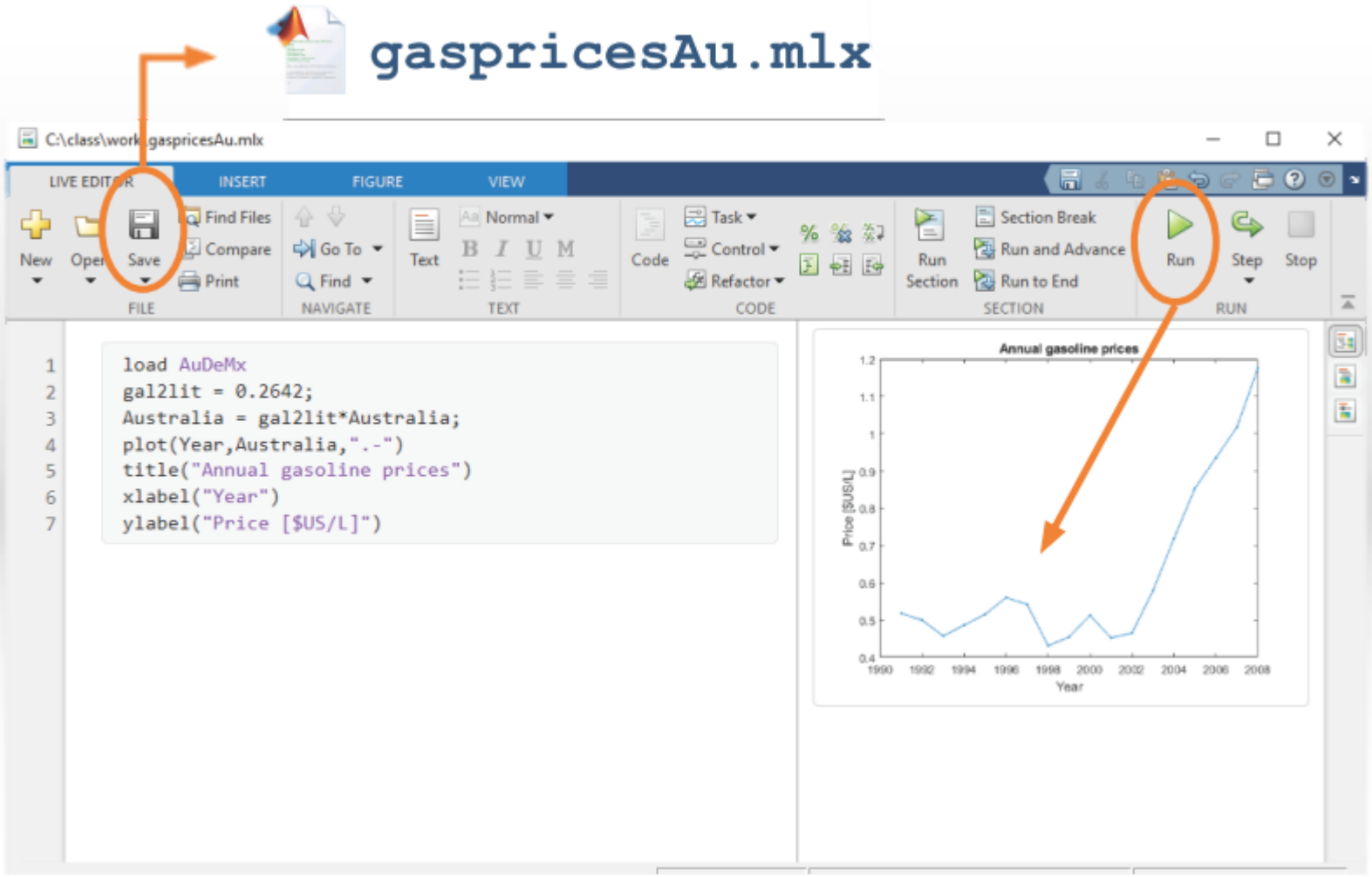
The Command History



The MATLAB Live Editor

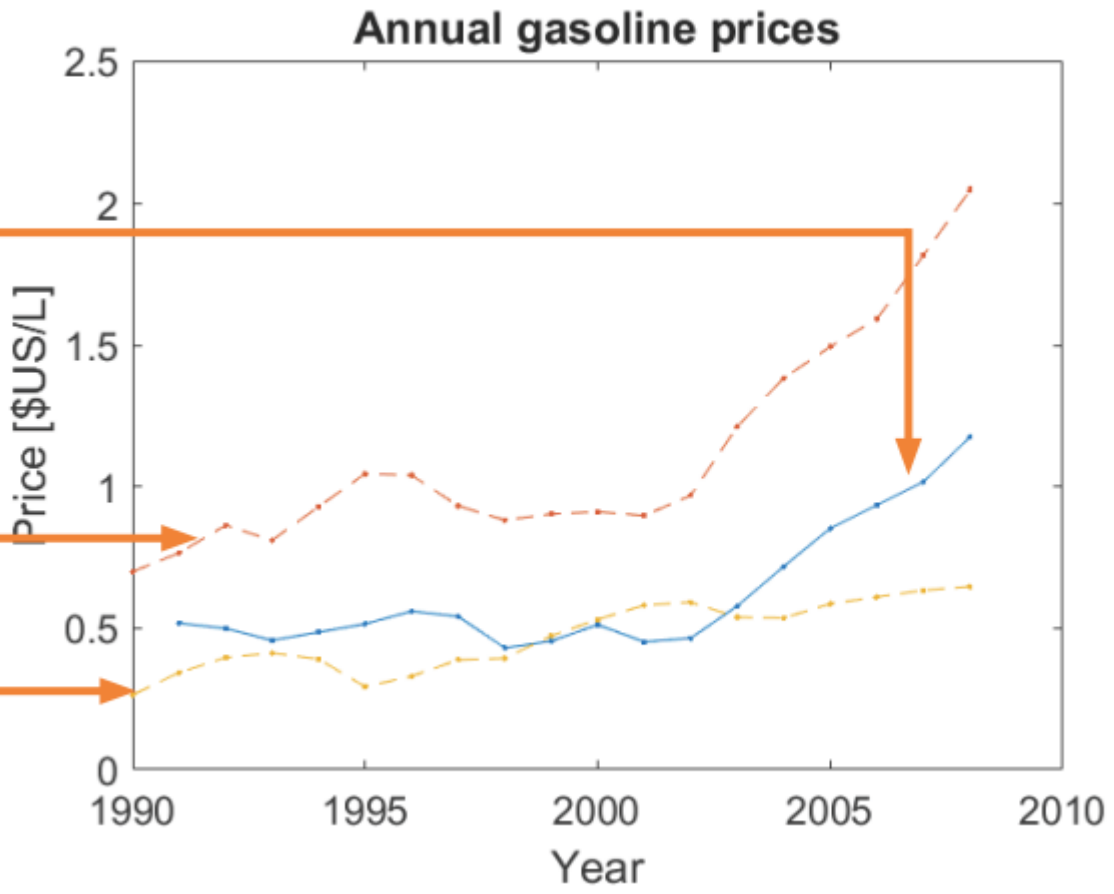


Live Scripts



Adding Plots

```
plot(...)
hold on
plot(...)
plot(...)
hold off
```

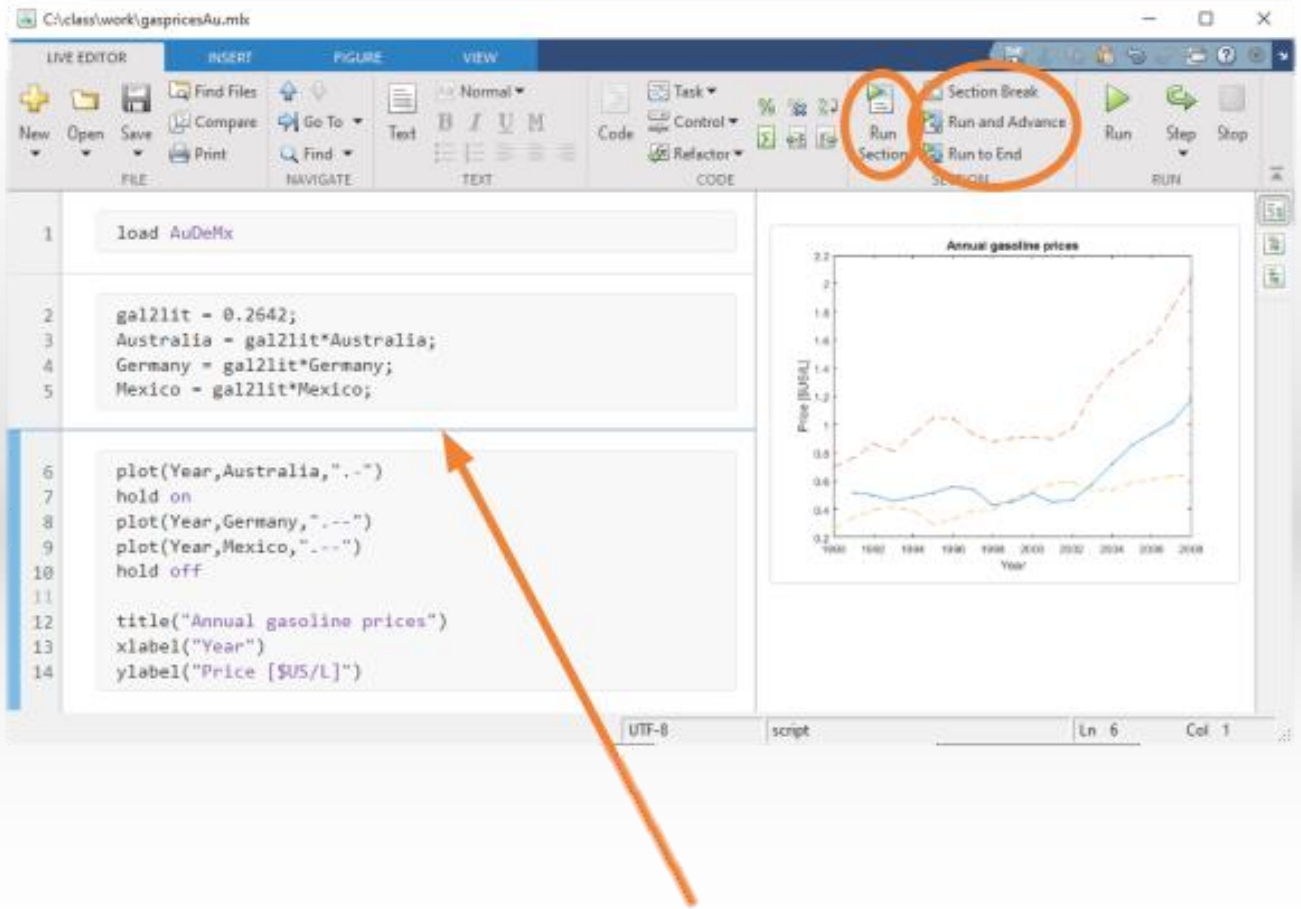


Code Sections

Load price data

Convert prices from gallons to liters

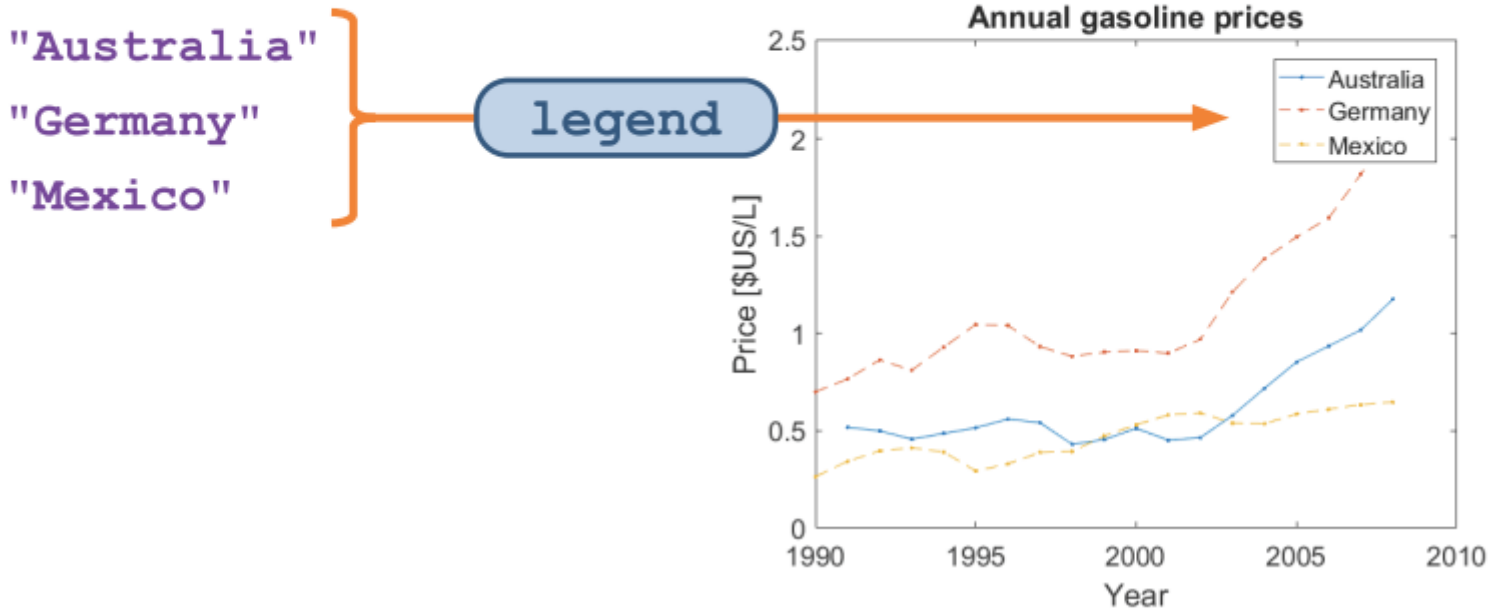
Plot results



Section Break

Adding a Plot Legend

```
legend("Australia","Germany","Mexico")
```



Chapter 2-1 Test Your Knowledge

1. (Select all that apply) Which of the following will create a matrix with three rows?
   - A. A = [zeros(2,4);ones(1,4)];
   - B. A = [1;2;3,4;5;6];
   - C. A = [1,2;3,4;5,6]';
   - D. A = rand(3);

2. Given a 5-by-5 matrix A, A(4:end,3:4) will produce a matrix of what size?
   - A. 1-by-2
   - B. 2-by-2
   - C. 2-by-3
   - D. 3-by-2

Chapter 2-2 Test Your Knowledge

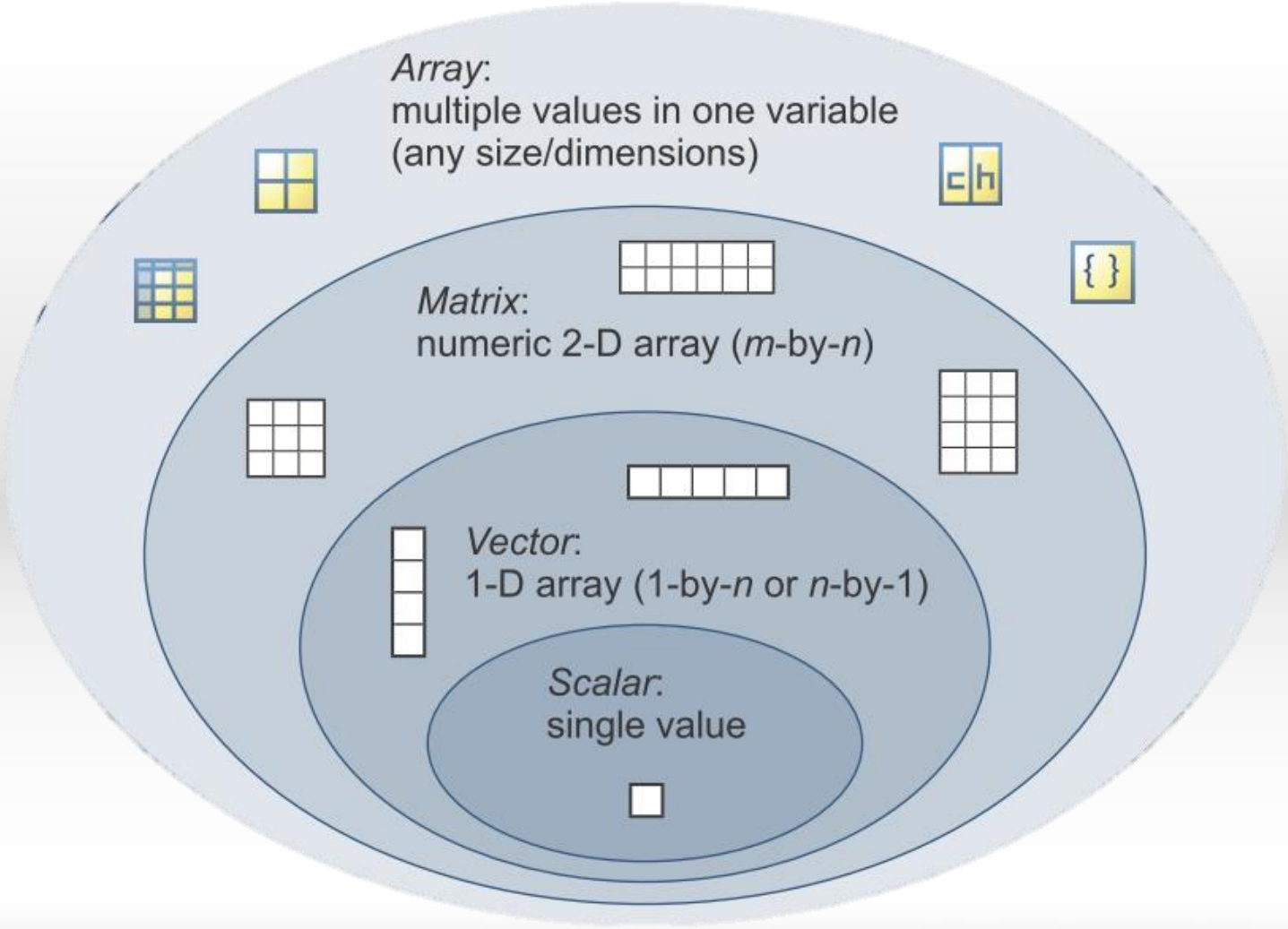
1. (Select all that apply) Which of the following will create a scatter plot of frogs on the horizontal axis and GDP on the vertical axis, with red markers at the data points?
   A. plot(GDP,frogs,'ro')
   B. plot(GDP,frogs,'o','r')
   C. plot(frogs,GDP,'ro')
   D. plot(frogs,GDP,'red')

2. If A is a 15-by-7 matrix, which of the following commands will result in five line plots on the same axes?
   A. plot(A(:))
   B. plot(A(:,3:end))
   C. plot(A(11:end,:))
   D. plot(A(2:6))

Course Outline

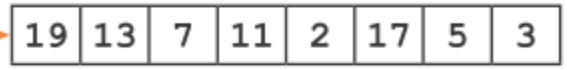
- Working with the MATLAB User Interface
- Variables, Commands, and Plots
- ➢ **Vector, Matrix, and Arrays**
- Table of Data
- Conditional Data Selection
- Analysis with Data
- Increasing Automation with Programming Constructs
- Appendix: Data Types

Vectors, Matrices, and Arrays



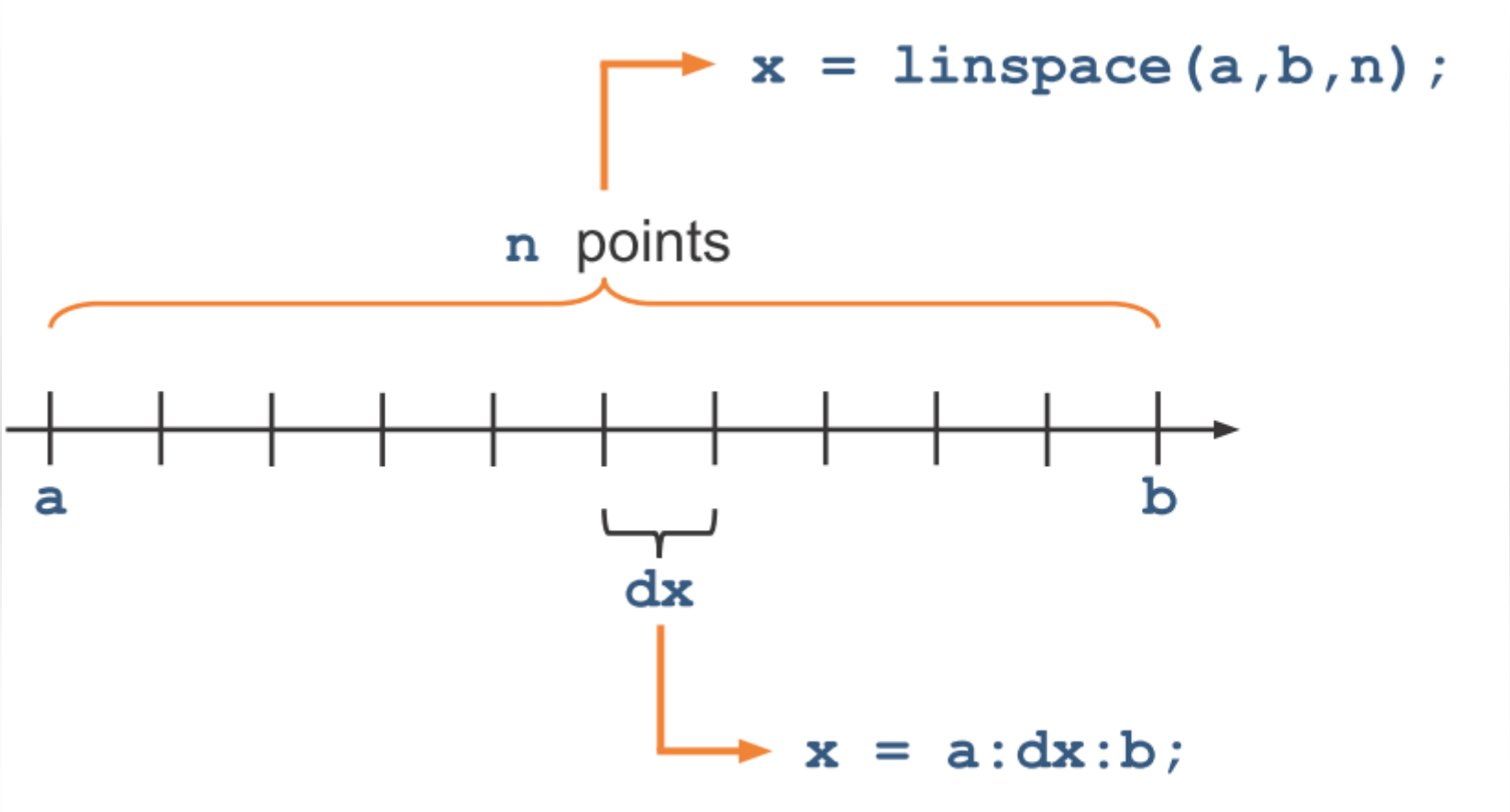
Entering Vectors Manually

```
x = [19 13 7 11 2 17 5 3]
```




```
x = [19;13;7;11;2;17;5;3]
```

Creating Vectors of Equally Spaced



Array Operations

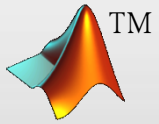
```
>> GMSum = Germany + Mexico
```

3.65	=	2.65	+	1.00
4.20		2.90		1.30
4.77		3.27		1.50
4.63		3.07		1.56
5.00		3.52		1.48
5.07		3.96		1.11
5.19		3.94		1.25
5.00		3.53		1.47
4.83		3.34		1.49
5.21		3.42		1.79
5.46		3.45		2.01
5.60		3.40		2.20
5.91		3.67		2.24
6.63		4.59		2.04
7.27		5.24		2.03
7.88		5.66		2.22
8.34		6.03		2.31
9.28		6.88		2.40
10.20		7.75		2.45

Mathematical Operations

Performed on all elements of a vector

$$\sin([a \quad b \quad c]) = [\sin(a) \quad \sin(b) \quad \sin(c)]$$

The operation is vectorized 

$$\begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \rightarrow \boxed{\sin} \rightarrow \begin{pmatrix} 0.8415 \\ 0.9093 \\ 0.1411 \\ -0.7568 \end{pmatrix}$$

- sin
- sind
- sinh
- asin
- exp
- log
- log2
- log10
- sqrt
- nthroot
- abs
- angle
- floor
- ceil
- round
- mod

Matrix Operations

Inner dimensions must be equal!!

WAvgPrices = Prices * rpop

WAvgPrices
215.4
226.6
235.0
233.4
241.4
252.4
252.1
244.7
227.1
247.7
271.9
257.1
257.5
293.1
334.9
376.0
406.2
440.1
497.1

=

Prices									
1.96	1.87	3.63	2.65	4.59	3.16	1.00	2.05	2.82	1.16
1.96	1.92	3.45	2.90	4.50	3.46	1.30	2.49	3.01	1.14
1.89	1.73	3.56	3.27	4.53	3.58	1.50	2.65	3.06	1.13
1.73	1.57	3.41	3.07	3.68	4.16	1.56	2.88	2.84	1.11
1.84	1.45	3.59	3.52	3.70	4.36	1.48	2.87	2.99	1.11
1.95	1.53	4.26	3.96	4.00	4.43	1.11	2.94	3.21	1.15
2.12	1.61	4.41	3.94	4.39	3.64	1.25	3.18	3.34	1.23
2.05	1.62	4.00	3.53	4.07	3.26	1.47	3.34	3.83	1.23
1.63	1.38	3.87	3.34	3.84	2.82	1.49	3.04	4.06	1.06
1.72	1.52	3.85	3.42	3.87	3.27	1.79	3.80	4.29	1.17
1.94	1.86	3.80	3.45	3.77	3.65	2.01	4.18	4.58	1.51
1.71	1.72	3.51	3.40	3.57	3.27	2.20	3.76	4.13	1.46
1.76	1.69	3.62	3.67	3.74	3.15	2.24	3.84	4.16	1.36
2.19	1.99	4.35	4.59	4.53	3.47	2.04	4.11	4.70	1.59
2.72	2.37	4.99	5.24	5.29	3.93	2.03	4.51	5.56	1.88
3.23	2.89	5.46	5.66	5.74	4.28	2.22	5.28	5.97	2.30
3.54	3.26	5.88	6.03	6.10	4.47	2.31	5.92	6.36	2.59
3.85	3.59	6.60	6.88	6.73	4.49	2.40	6.21	7.13	2.80
4.45	4.08	7.51	7.75	7.63	5.74	2.45	5.83	7.42	3.27

*

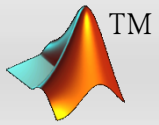
rpop
2.1
3.7
7.1
9.0
6.6
13.9
11.8
5.3
6.7
33.6

334.9 = 2.72*2.1 + 2.37*3.7 + 4.99*7.1 + 5.24*9.0 + 5.29*6.6 + 3.93*13.9 + 2.03*11.8 + 4.51*5.3 + 5.56*6.7 + 1.88*33.6

Mathematical Operations

Performed on all elements of a matrix

$$\sin\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} \sin(a) & \sin(b) \\ \sin(c) & \sin(d) \end{pmatrix}$$

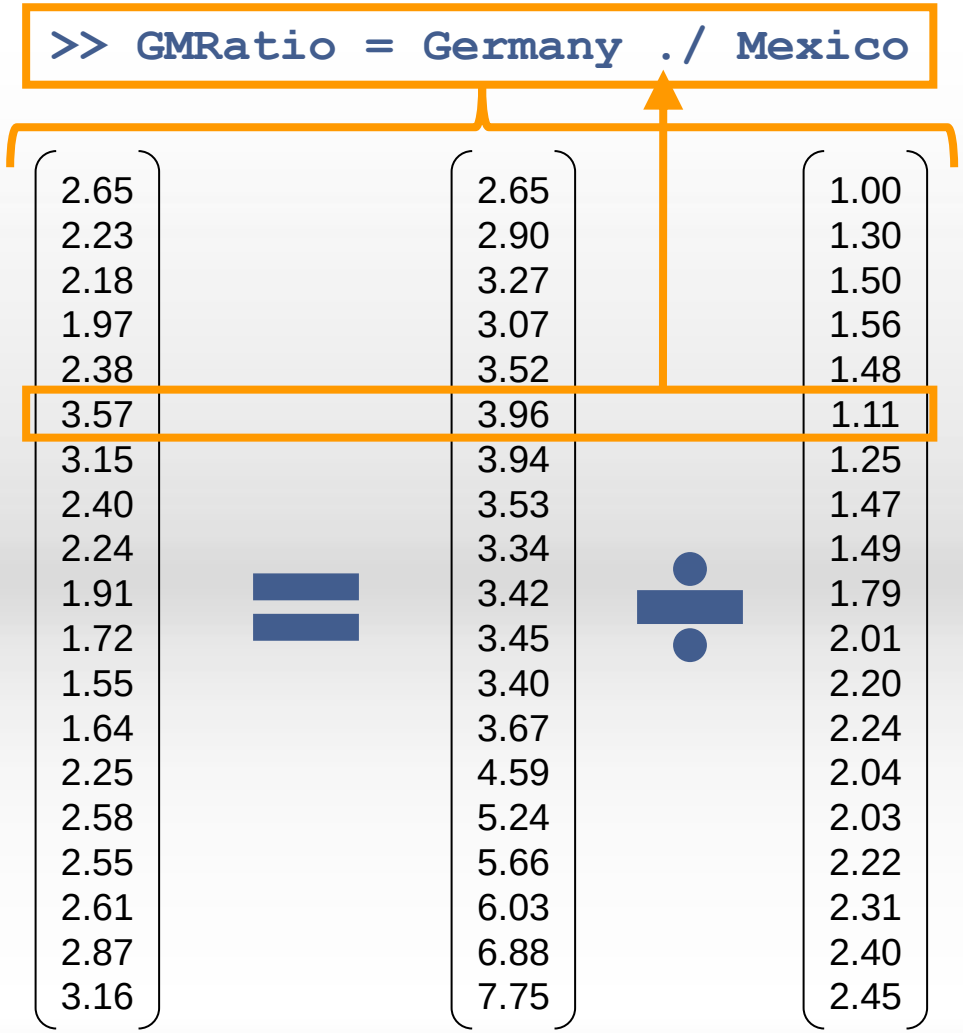
The operation is vectorized 

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \rightarrow \boxed{\sin} \rightarrow \begin{pmatrix} 0.8415 & 0.9093 \\ 0.1411 & -0.7568 \end{pmatrix}$$

- sin
- sind
- sinh
- asin
- exp
- log
- log2
- log10
- sqrt
- nthroot
- abs
- angle
- floor
- ceil
- round
- mod

Array Operations

Matrices must have the same dimensions!!



Concatenating Arrays

A

2	3
5	7
11	13

B

-1	1
1	-1

C

0
8
0

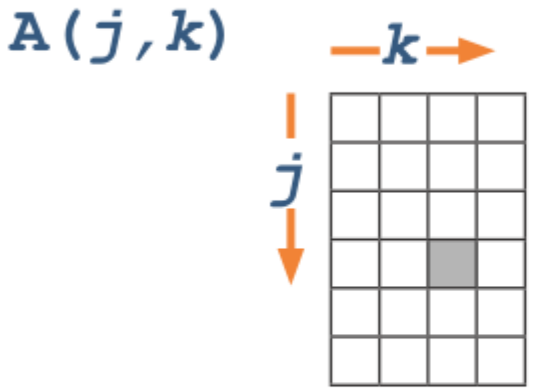
```
X = [A;B]
```

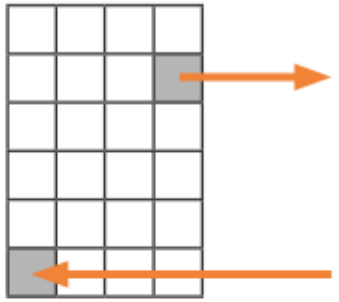
2	3
5	7
11	13
-1	1
1	-1

```
Y = [A,C]
```

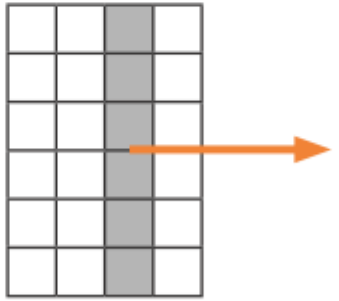
2	3	0
5	7	8
11	13	0

Accessing Data in Matrix





```
x = A(2,end)
A(end,1) = x
```

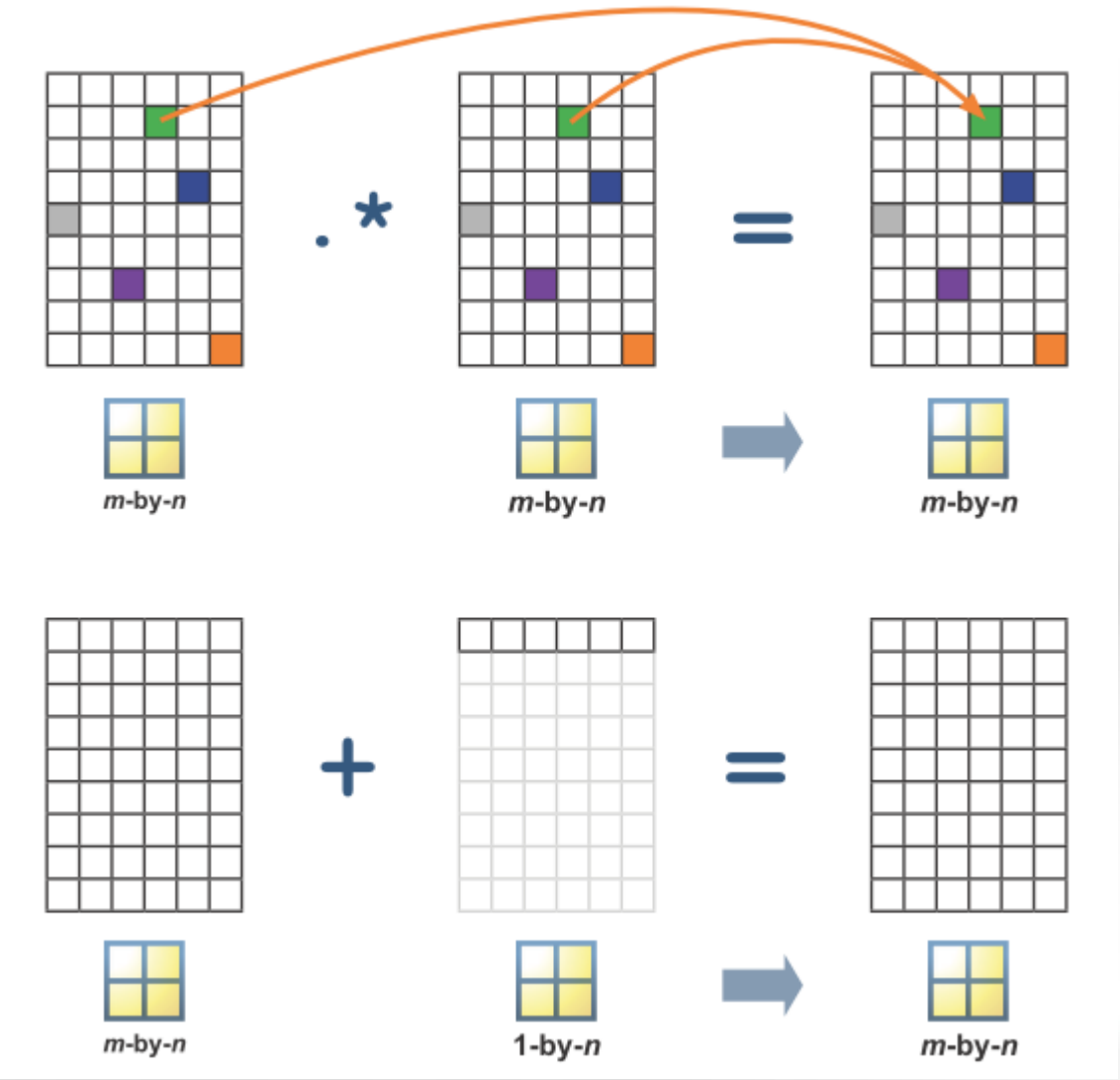


```
x = A(1:6,3)
x = A(1:end,3)
x = A(:,3)
```

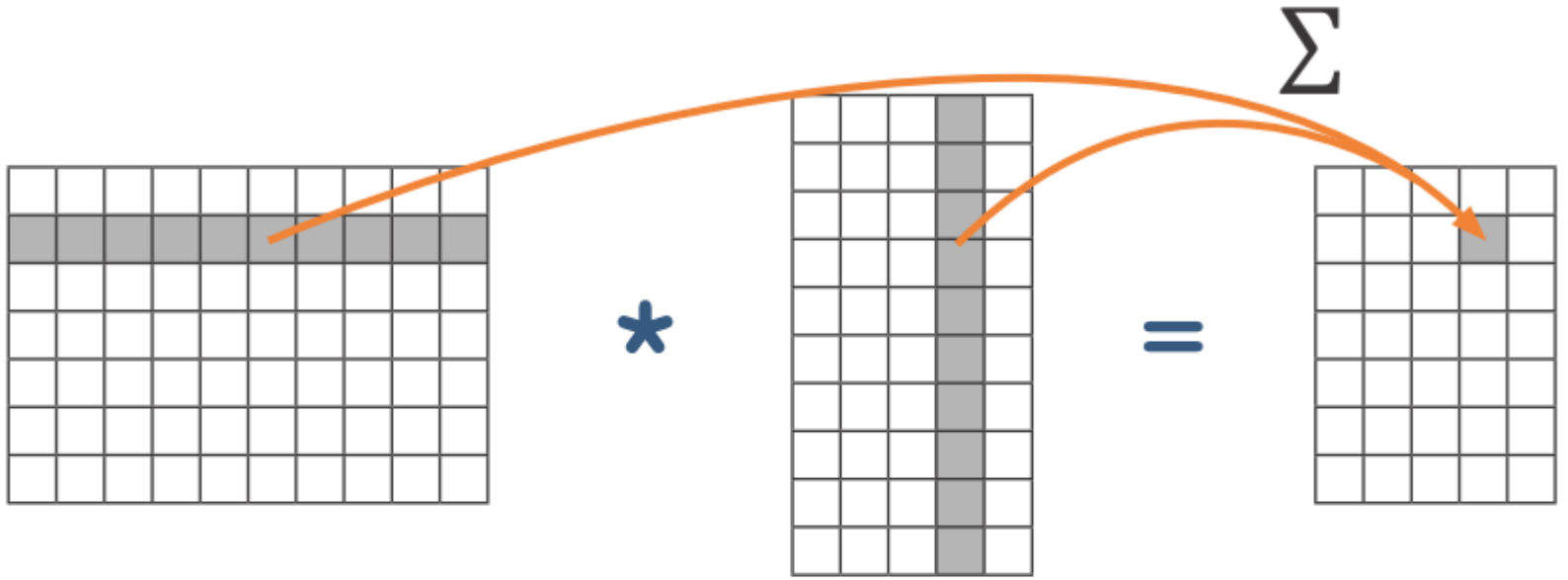
Matrix Operation

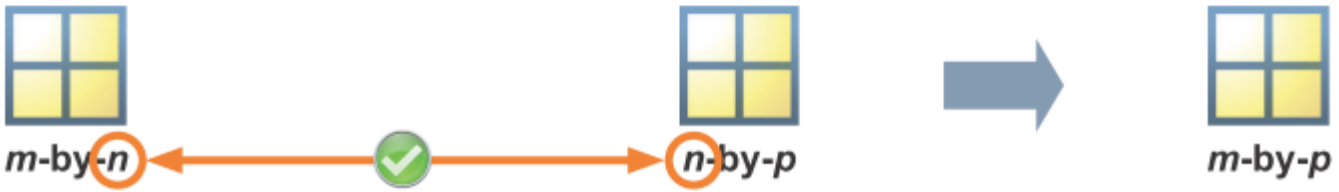
```
priceDMWh = price*10
```

Array Operation

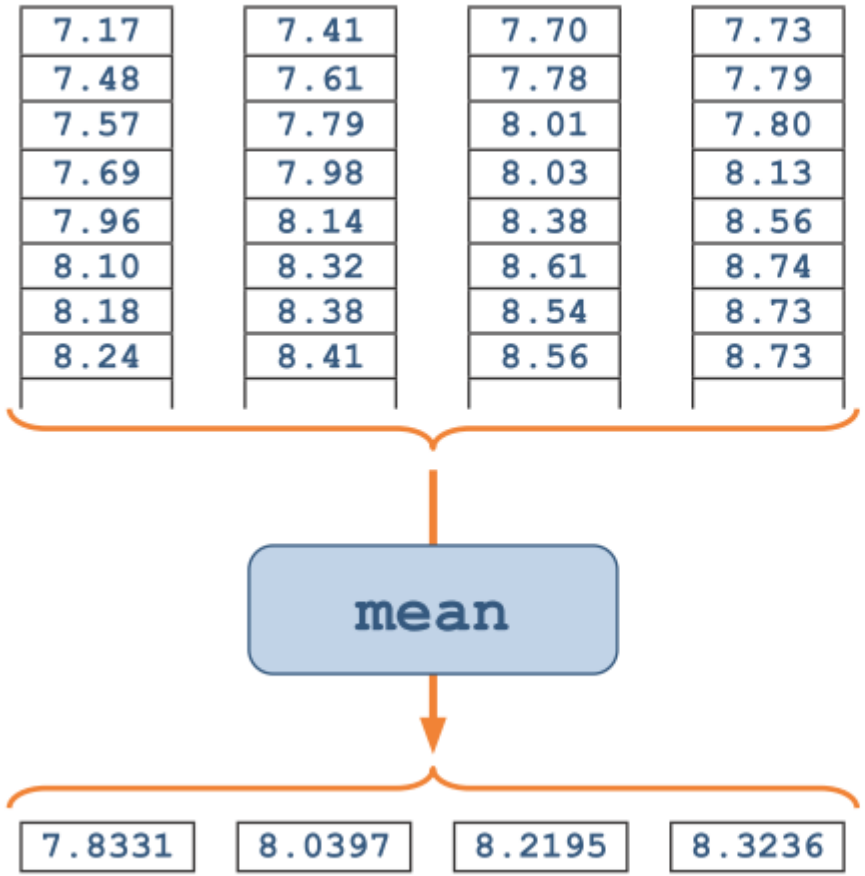


Matrix Mathematics



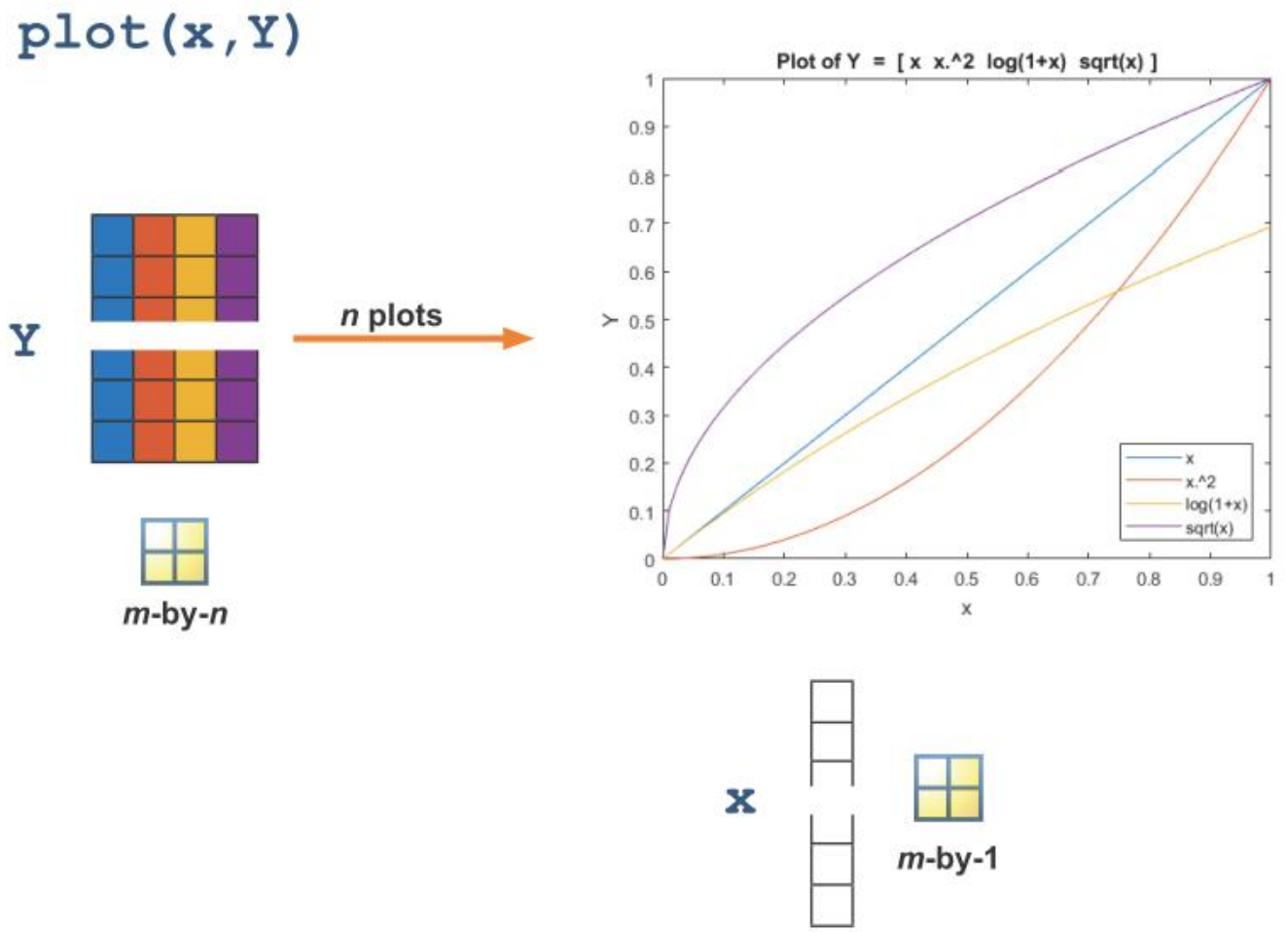


Statistical Operations

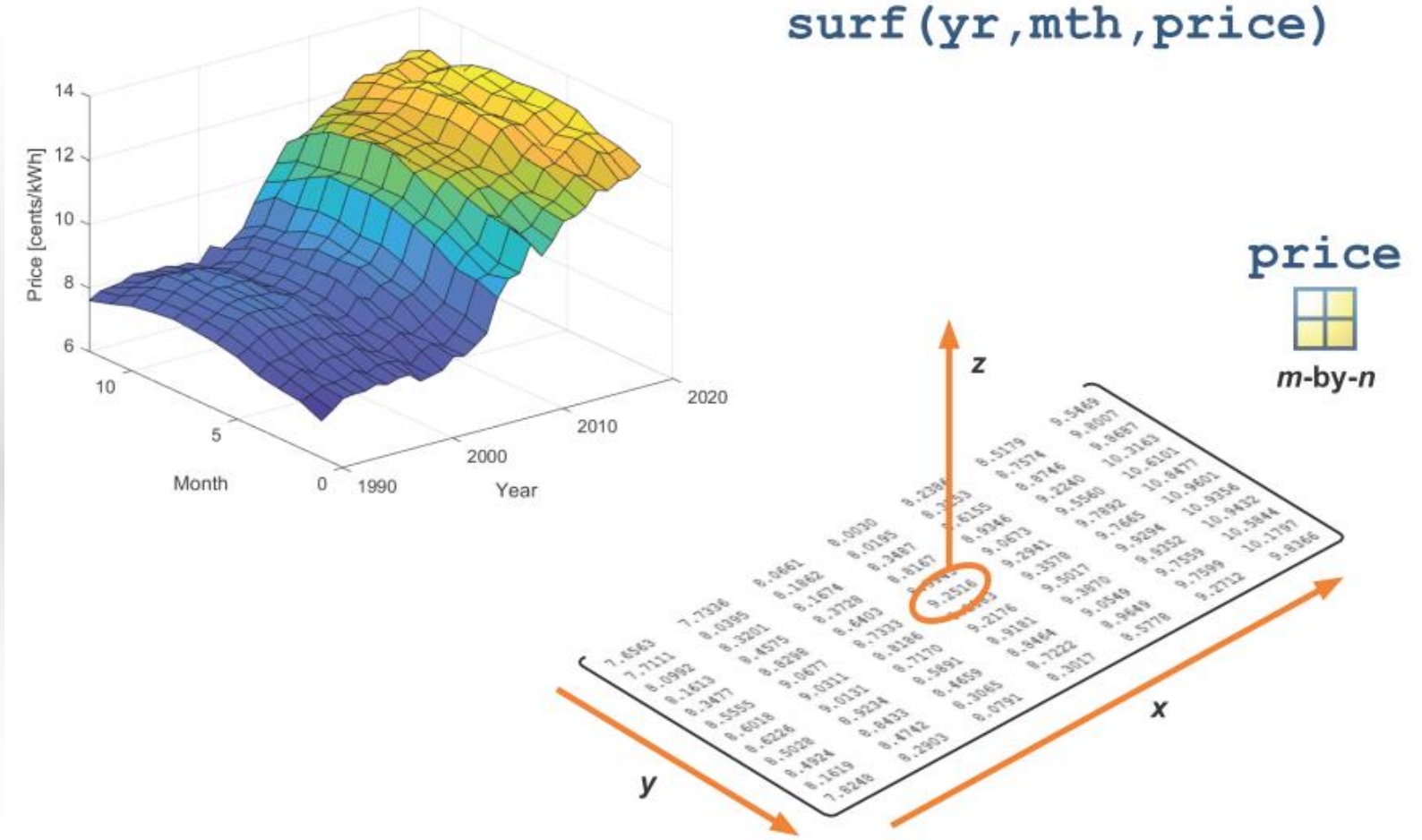



```
max
min
mean
median
std
sum
prod
diff
gradient
cumsum
cumprod
corrcoef
cov
```

Plotting Multiple Columns



Matrix Visualization



Example: Array Operations

- In most languages - use loops:

```
>> tic; for I = 1:10000
Density(I) = Mass(I)/(Length(I)*Width(I)*Height(I));
end; toc
elapsed_time =
    4.7260
```

Use TIC and TOC to measure elapsed time

- In MATLAB - use array operations:

```
>> tic; Density = Mass./(Length.*Width.*Height); toc
elapsed_time =
     0
```

Vectorized code is much faster than loops

```
>> array_examp
```

Chapter 3-1 Test Your Knowledge

1. (Select all that apply) Given two matrices, A and B, where A is a 2-by-3 matrix and B is a 3-by-2 matrix, which of the following operations are valid?
   A. A+B
   B. A.+B
   C. A*B
   D. A.*B

2. Given a vector x, what is the command to add 3 to each element, double that value, then sum all the resulting values?
   A. sum(2*x+3)
   B. sum(2*[x(k)+3])
   C. sum[2*x+3]
   D. sum(2*(x+3))

Chapter 3-2 Test Your Knowledge

3. (Select all that apply) Which commands are equivalent to the command `x = 1.4:2:6.8` ?

A. `x = [1.4 2 6.8]`
B. `x = [1.4 6.8]`
C. `x = [1.4 3.4 5.4]`
D. `x = [1.4 3.4 5.4 6.8]`
E. `x = [3.4 5.4]`

Course Outline

- Working with the MATLAB User Interface
- Variables, Commands, and Plots
- Vector, Matrix, and Arrays
- **Table of Data**
- Conditional Data Selection
- Analysis with Data
- Increasing Automation with Programming Constructs
- Appendix: Data Types

Course Example : Premier League Scoring

Team	Home Wins	Home Draws	Home Losses	Home GF	Home GA	Away Wins	Away Draws	Away Losses	Away GF	Away GA
Arsenal	12	4	3	31	11	8	7	4	34	25
Aston Villa	2	5	12	14	35	1	3	15	13	41
Bournemouth	5	5	9	23	34	6	4	9	22	33
Chelsea	5	9	5	32	30	7	5	7	27	23
Crystal Palace	6	3	10	19	23	5	6	8	20	28
Everton	6	5	8	35	30	5	9	5	24	25
Leicester City	12	6	1	35	18	11	6	2	33	18
Liverpool	8	8	3	33	22	8	4	7	30	28
Manchester City	12	2	5	47	21	7	7	5	24	20
Manchester United	12	5	2	27	9	7	4	8	22	26
Newcastle United	7	7	5	32	24	2	3	14	12	41
Norwich City	6	5	8	26	30	3	2	14	13	37
Southampton	11	3	5	39	22	7	6	6	20	19
Stoke City	8	4	7	22	24	6	5	8	19	31
Sunderland	6	6	7	23	20	3	6	10	25	
Swansea City	8	6	5	20	20	4	5	10	22	
Tottenham Hotspur	10	6	3	35	15	9	7	3	34	
Watford	6	6	7	20	19	6	3	10	20	
West Bromwich Albion	6	5	8	20	26	4	8	7	14	
West Ham United	9	7	3	34	26	7	7	5	31	

What is a Table?

Each column is a named variable

Each row is an observation

Team	Home Wins	Home Draws	Home Losses	Home GF	Home GA	Away Wins	Away Draws	Away Losses	Away GF	Away GA
Arsenal	12	4	3	31	11	8	7	4	34	25
Aston Villa	2	5	12	14	35	1	3	15	13	41
Bournemouth	5	5	9	23	34	6	4	9	22	33
Chelsea	5	9	5	32	30	7	5	7	27	23
Crystal Palace	6	3	10	19	23	5	6	8	20	28
Everton	6	5	8	35	30	5	9	5	24	25
Leicester City	12	6	1	35	18	11	6	2	33	18
Liverpool	8	8	3	33	22	8	4	7	30	28
Manchester City	12	2	5	47	21	7	7	5	24	20
Manchester United	12	5	2	27	9	7	4	8	22	26
Newcastle United	7	7	5	32	24	2	3	14	12	41
Norwich City	6	5	8	26	30	3	2	14	13	37
Southampton	11	3	5	39	22	7	6	6	20	19
Stoke City	8	4	7	22	24	6	5	8	19	31
Sunderland	6	6	7	23	20	3	6	10	25	42
Swansea City	8	6	5	20	20	4	5	10	22	32
Tottenham Hotspur	10	6	3	35	15	9	7	3	34	20
Watford	6	6	7	20	19	6	3	10	20	31
West Bromwich Albion	6	5	8	20	26	4	8	7	14	22
West Ham United	9	7	3	34	26	7	7	5	31	25

Operating on tables

```
byHomeWins = sortrows(EPL,"HomeWins","descend");
```

EPL
20x11 table

	Team	HomeWins	HomeDraws	HomeLosse	HomeGF	HomeGA	AwayWins	AwayDraws	AwayLosses	AwayGF	AwayGA
1	'Arsenal'	12	4	3	31	11	8	7	4	34	25
2	'Aston Villa'	2	5	12	14	35	1	3	15	13	41
3	'Bournemouth'	5	5	9	23	34	6	4	9	22	33
4	'Chelsea'	5	9	5	32	30	7	5	7	27	23
5	'Crystal Palace'	6	3	10	19	23	5	6	8	20	28
6	'Everton'	6	5	8	35	30	5	9	5	24	25
7	'Leicester City'	12	6	1	35	18	11	6	2	33	18
8	'Liverpool'	8	8	3	33	22	8	4	7	30	28
9	'Manchester City'	12	2	5	47	21	7	7	5	24	20
10	'Manchester United'	12	5	2	27	9	7	4	8	22	26
11	'Newcastle United'	7	7	5	32	24	2	3	14	12	41
12	'Norwich City'	6	5	8	26	30	3	2	14	13	37
13	'Southampton'	11	3	5	39	22	7	5	6	20	19
14	'Stoke City'	8	4	7	22	24	6	5	8	19	31
15	'Sunderland'	6	6	7	23	20	3	6	10	25	42
16	'Swansea City'	8	6	5	20	20	4	5	10	22	32
17	'Tottenham Hotspur'	10	6	3	35	15	9	7	3	34	20
18	'Watford'	6	6	7	20	19	6	3	10	20	31
19	'West Bromwich Albi...	6	5	8	20	26	4	8	7	14	22
20	'West Ham United'	9	7	3	34	26	7	7	5	31	25
21											

20-by-11

Observations stay together

		HomeWins	HomeDraws	HomeLosses	HomeGF	HomeGA	AwayWins	AwayDraws	AwayLosses	AwayGF	AwayGA
		12	4	3	31	11	8	7	4	34	25
		12	6	1	35	18	11	6	2	33	18
		12	2	5	47	21	7	7	5	24	20
	ed'	12	5	2	27	9	7	4	8	22	26
		11	3	5	39	22	7	6	6	20	19
	pur'	10	6	3	35	15	9	7	3	34	20
7	'West Ham United'	9	7	3	34	26	7	7	5	31	25
8	'Liverpool'	8	8	3	33	22	8	4	7	30	28
9	'Stoke City'	8	4	7	22	24	6	5	8	19	31
10	'Swansea City'	8	6	5	20	20	4	5	10	22	32
11	'ewcastle United'	7	7	5	32	24	2	3	14	12	41
12	'Crystal Palace'	6	3	10	19	23	5	6	8	20	28
13	'Everton'	6	5	8	35	30	5	9	5	24	25
14	'Norwich City'	6	5	8	26	30	3	2	14	13	37
15	'Sunderland'	6	6	7	23	20	3	6	10	25	42
16	'Watford'	6	6	7	20	19	6	3	10	20	31
17	'West Bromwich Alb...	6	5	8	20	26	4	8	7	14	22
18	'Bournemouth'	5	5	9	23	34	6	4	9	22	33
19	'Chelsea'	5	9	5	32	30	7	5	7	27	23
20	'Aston Villa'	2	5	12	14	35	1	3	15	13	41

20-by-11

Extracting Portions of a Table

```
Top10Wins = EPL(1:10,[1,2,7]);
```

10-by-3

EPL

20x11 table

	1 Team	2 HomeWins	3 HomeDraws	4 HomeLosses	5 HomeGF	6 HomeGA	7 AwayWins	8 AwayDraws	9 AwayLosses	10 AwayGF	11 AwayGA
1	'Leicester City'	12	6	1	35	1	11	6	2	33	18
2	'Arsenal'	12	4	3	31	1	8	7	4	34	25
3	'Manchester City'	12	2	5	47	2	7	7	5	24	20
4	'Manchester United'	12	5	2	27		7	4	8	22	26
5	'Southampton'	11	3	5	39	2	7	6	6	20	19
6	'Tottenham Hotspur'	10	6	3	35	1	9	7	3	34	20
7	'West Ham United'	9	7	3	31	2	7	7	5	31	25
8	'Liverpool'	8	8	3	33	2	8	4	7	30	28
9	'Stoke City'	8	4	7	22	2	6	5	8	19	31
10	'Swansea City'	8	6	5	20	2	4	5	10	22	32
11	'Newcastle United'	7	7	5	32	24	2	3	14	12	41
12	'Watford'	6	6	7	20	19	6	3	10	20	31
13	'Crystal Palace'	6	3	10	19	23	5	6	8	20	28
14	'Everton'	6	5	8	35	30	5	9	5	24	25
15	'West Bromwich Albion'	6	5	8	20	26	4	8	7	14	22
16	'Norwich City'	6	5	8	26	30	3	2	14	13	37
17	'Sunderland'	6	6	7	23	20	3	6	10	25	42
18	'Chelsea'	5	9	5	32	30	7	5	7	27	23
19	'Bournemouth'	5	5	9	23	34	6	4	9	22	33
20	'Aston Villa'	2	5	12	14		1	3	15	13	41

20-by-11

Extracting Data from a Table

```
homeWins = EPL.HomeWins;
```

20-by-1

20-by-11

EPL

20x11 table

	1 Team	2 HomeWins	3 HomeDraws	4 HomeLosses	5 HomeGF	6 HomeGA	7 AwayWins	8 AwayDraws	9 AwayLosses	10 AwayGF	11 AwayGA
1	'Leicester City'	12	6		35	18	11	6	2	33	18
2	'Arsenal'	12	4		31	11	8	7	4	34	25
3	'Manchester City'	12	2		47	21	7	7	5	24	20
4	'Manchester United'	12	5		27	9	7	4	8	22	26
5	'Southampton'	11	3			2	7	6	6	20	19
6	'Tottenham Hotspur'	10	6			5	9	7	3	34	20
7	'West Ham United'	9	7		34	26	7	7	5	31	25
8	'Liverpool'	8	8		33	22	8	4	7	30	28
9	'Stoke City'	8	4		22	24	6	5	8	19	31
10	'Swansea City'	8	6		20	20	4	5	10	22	32
11	'Newcastle United'	7	7	5	32	24	2	3	14	12	41
12	'Watford'	6	6	7	20	19	6	3	10	20	31
13	'Crystal Palace'	6	3	10	19	23	5	6	8	20	28
14	'Everton'	6	5	8	35	30	5	9	5	24	25
15	'West Bromwich Albion'	6	5	8	20	26	4	8	7	14	22
16	'Norwich City'	6	5	8	26	30	3	2	14	13	37
17	'Sunderland'	6	6	7	23	20	3	6	10	25	42
18	'Chelsea'	5	9	5	32	30	7	5	7	27	23
19	'Bournemouth'	5	5	9	23	34	6	4	9	22	33
20	'Aston Villa'	2	5	12	14	35	1	3	15	13	41

```
topHomeScoring = EPL{1:10,5:6};
```

Modifying Tables

```
EPL.Points = points;
```

20-by-1

EPL

20x12 table

	1 Team	2 HomeWins	3 HomeDraws	4 HomeLosses	5 HomeGF	6 HomeGA	7 AwayWins	8 AwayDraws	9 AwayLosses	10 AwayGF	11 AwayGA	12 Points
1	'Leicester City'	12	6	1	35	18	11	6	2	33	18	81
2	'Arsenal'	12	4	3	31	11	8	7	4	34	25	71
3	'Manchester City'	12	2	5	47	21	7	7	5	24	20	66
4	'Manchester United'	12	5	2	27	9	7	4	8	22	26	66
5	'Southampton'	11	3	5	39	22	7	6	6	20	19	63
6	'Tottenham Hotspur'	10	6	3	35	15	9	7	3	34	20	70
7	'West Ham United'	9	7	3	34	26	7	7	5	31	25	62
8	'Liverpool'	8	8	3	33	22	8	4	7	30	28	60
9	'Stoke City'	8	4	7	22	24	6	5	8	19	31	51
10	'Swansea City'	8	6	5	20	20	4	5	10	22	32	47
11	'Newcastle United'	7	7	5	32	24	2	3	14	12	41	37
12	'Watford'	6	6	7	20	19	6	3	10	20	31	45
13	'Everton'	6	5	8	35	30	5	9	5	24	25	47
14	'Crystal Palace'	6	3	10	19	23	5	6	8	20	28	42
15	'West Bromwich Albion'	6	5	8	20	26	4	8	7	14	22	43
16	'Sunderland'	6	6	7	23	20	3	6	10	25	42	39
17	'Norwich City'	6	5	8	26	30	3	2	14	13	37	34
18	'Chelsea'	5	9	5	32	30	7	5	7	27	23	50
19	'Bournemouth'	5	5	9	23	34	6	4	9	22	33	42
20	'Aston Villa'	2	5	12	14	35	1	3	15	13	41	17
21												

20-by-11 → 20-by-12

Chapter 4-1 Test Your Knowledge

1. If x is a 20-by-5 table with variables “A”, “B”, “C”, “D”, and “E”, which are all numeric vectors, the command

```
y = x.C
```

will return:

A. A 20-by-1 numeric vector
B. A 20-by-1 table
C. A 20-by-5 numeric array
D. A 1-by-1 table
E. An error message

Chapter 4-2 Test Your Knowledge

2. If x is a 20-by-5 table with variables “A”, “B”, “C”, “D”, and “E”, which are all numeric vectors, the command

```
y = x(:,"C")
```

will return:

A. A 20-by-1 numeric vector
B. A 20-by-1 table
C. A 20-by-5 numeric array
D. A 1-by-1 table
E. An error message

Course Outline

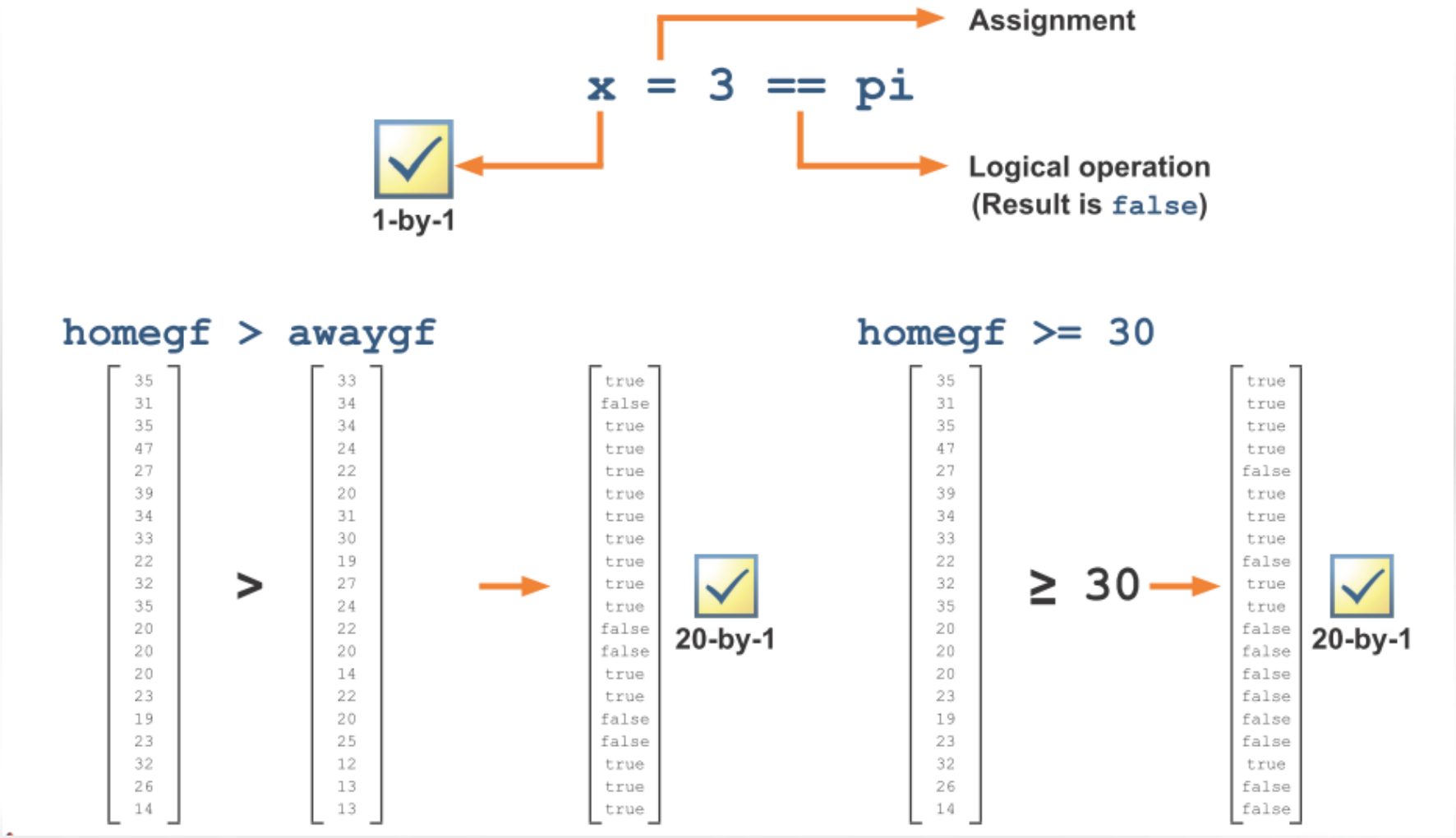
- Working with the MATLAB User Interface
- Variables, Commands, and Plots
- Vector, Matrix, and Arrays
- Table of Data
- ➢ **Conditional Data Selection**
- Analysis with Data
- Increasing Automation with Programming Constructs
- Appendix: Data Types

Course Example : Investigating Premier League Scoring

Team	Home Wins	Home Draws	Home Losses	Home GF	Home GA	Away Wins	Away Draws	Away Losses	Away GF	Away GA	Total Wins	Total Draws	Total Losses	Points
Leicester City	12	6	1	35	18	11	6	2	33	18	23	12	3	81
Arsenal	12	4	3	31	11	8	7	4	34	25	20	11	7	71
Tottenham Hotspur	10	6	3	35	15	9	7	3	34	20	19	13	6	70
Manchester City	12	2	5	47	21	7	7	5	24	20	19	9	10	66
Manchester United	12	5	2	27	9	7	4	8	22	26	19	9	10	66
Southampton	11	3	5	39	22	7	6	6	20	19	18	9	11	63
West Ham United	9	7	3	34	26	7	7	5	31	25	16	14	8	62
Liverpool	8	8	3	33	22	8	4	7	30	28	16	12	10	60
Stoke City	8	4	7	22	24	6	5	8	19	31	14	9	15	51
Chelsea	5	9	5	32	30	7	5	7	27	23	12	14	12	50
Everton	6	5	8	35	30	5	9	5	24	25	11	14	13	47
Swansea City	8	6	5	20	20	4	5	10	22	32	12	11	15	47
Watford	6	6	7	20	19	6	3	10	20	31	12	9	17	45
West Bromwich Albion	6	5	8	20	26	4	8	7	14	22	10	13	15	43
Bournemouth	5	5	9	23	34	6	4	9	22	33	11	9	18	42
Crystal Palace	6	3	10	19	23	5	6	8	20	28	11	9	18	42
Sunderland	6	6	7	23	20	3	6	10	25	42	9	12	17	9
Newcastle United	7	7	5	32	24	2	3	14	12	41	9	10	19	37
Norwich City	6	5	8	26	30	3	2	14	13	37	9	7	22	
Aston Villa	2	5	12	14	35	1	3	15	13	41	3	8	27	17

Home vs. Away Scoring:
Teams with Winning Home Records

Logical Operations and Variables



Finding and Counting

```
x = homewinning & ~awaywinning
```

```
x =
  false
  false
  false
  false
   true
  false
  false
  false
   true
  false
  false
   true
  false
  false
  false
  false
   true
  false
  false
```

```
find(x)  →  [ 5; 9; 12; 18 ]

nnz(x)   →  4
```

Logical Indexing

```
EPL.Team(homegf >= 30)
```

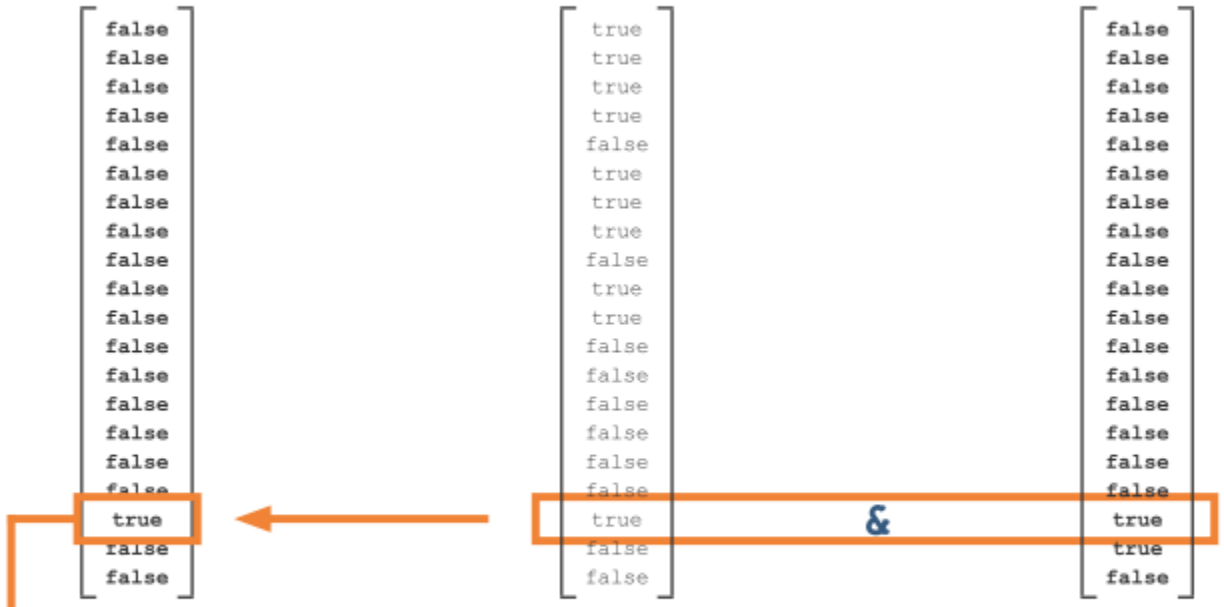
Leicester City	true
Arsenal	true
Tottenham Hotspur	true
Manchester City	true
Manchester United	false
Southampton	true
West Ham United	true
Liverpool	true
Stoke City	false
Chelsea	true
Everton	true
Swansea City	false
Watford	false
West Bromwich Albion	false
Bournemouth	false
Crystal Palace	false
Sunderland	false
Newcastle United	true
Norwich City	false
Aston Villa	false

~~idx = find(...)~~
~~EPL.Team(idx)~~

```
Leicester City
Arsenal
Tottenham Hotspur
Manchester City
Southampton
West Ham United
Liverpool
Chelsea
Everton
Newcastle United
```

Combining Logical Condition

```
bigHFA = (homegf >= 30) & (homegf > 2*awaygf)
```



any(bigHFA) ☑ true

1-by-1

Chapter 5-1 Test Your Knowledge

1. If `x` and `y` are both 20-by-1 (numeric) vectors and half of the elements of `y` are greater than 0.5, the command

```
y > 0.5
```

will return:

A. 10-by-1 numeric vector of y values
B. A 20-by-1 logical vector
C. A 20-by-1 numeric vector of y values
D. An error message

Chapter 5-2 Test Your Knowledge

2. If `x` and `y` are both 20-by-1 (numeric) vectors and half of the elements of `y` are greater than 0.5, the command

```
x(y > 0.5)
```

will return:

A. A 10-by-1 numeric vector of `x` values

B. A 10-by-1 numeric vector of `y` values

C. A 20-by-1 logical vector

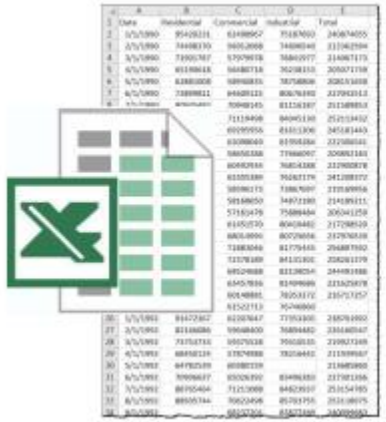
D. A 20-by-1 numeric vector of `x` values

E. An error message

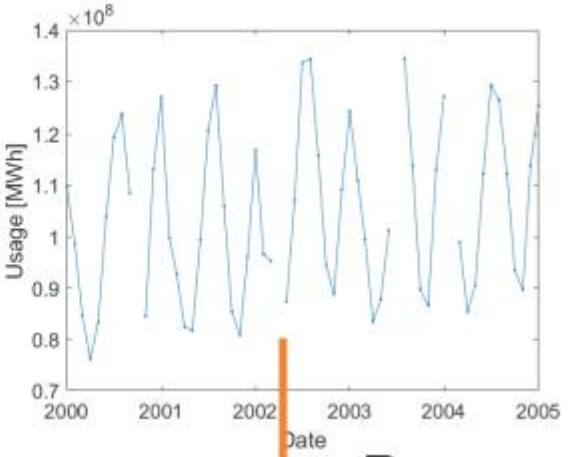
Course Outline

- Working with the MATLAB User Interface
- Variables, Commands, and Plots
- Vector, Matrix, and Arrays
- Table of Data
- Conditional Data Selection
- **Analysis with Data**
- Increasing Automation with Programming Constructs
- Appendix: Data Types

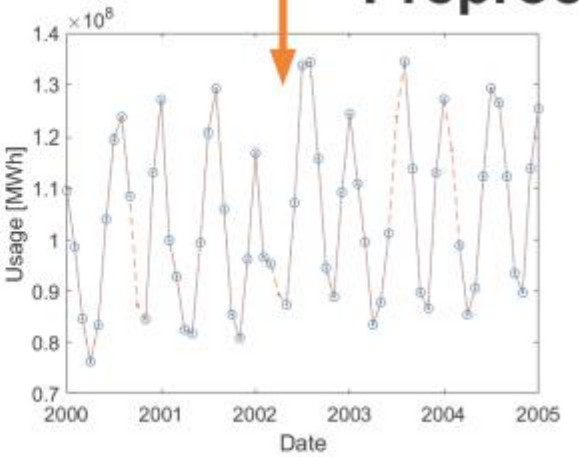
Course Example : Modeling Electricity Consumption



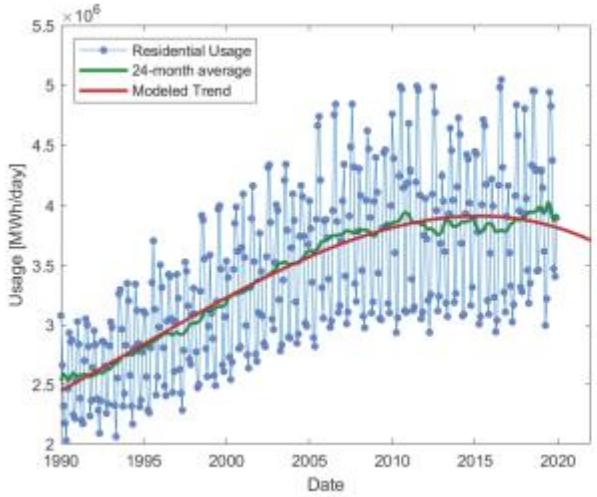
Import



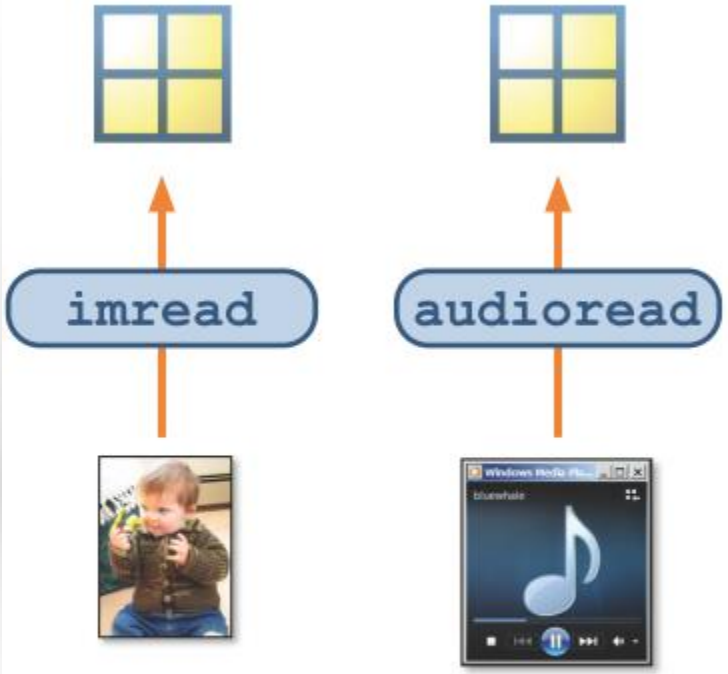
Preprocess

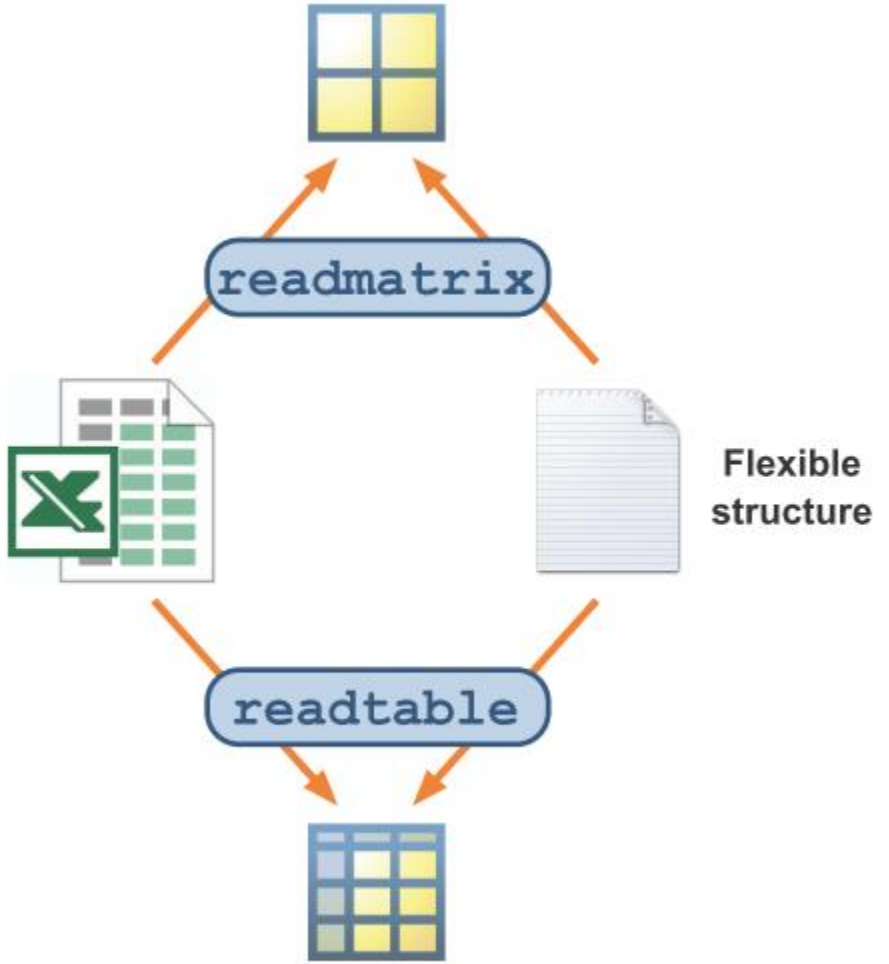


Model

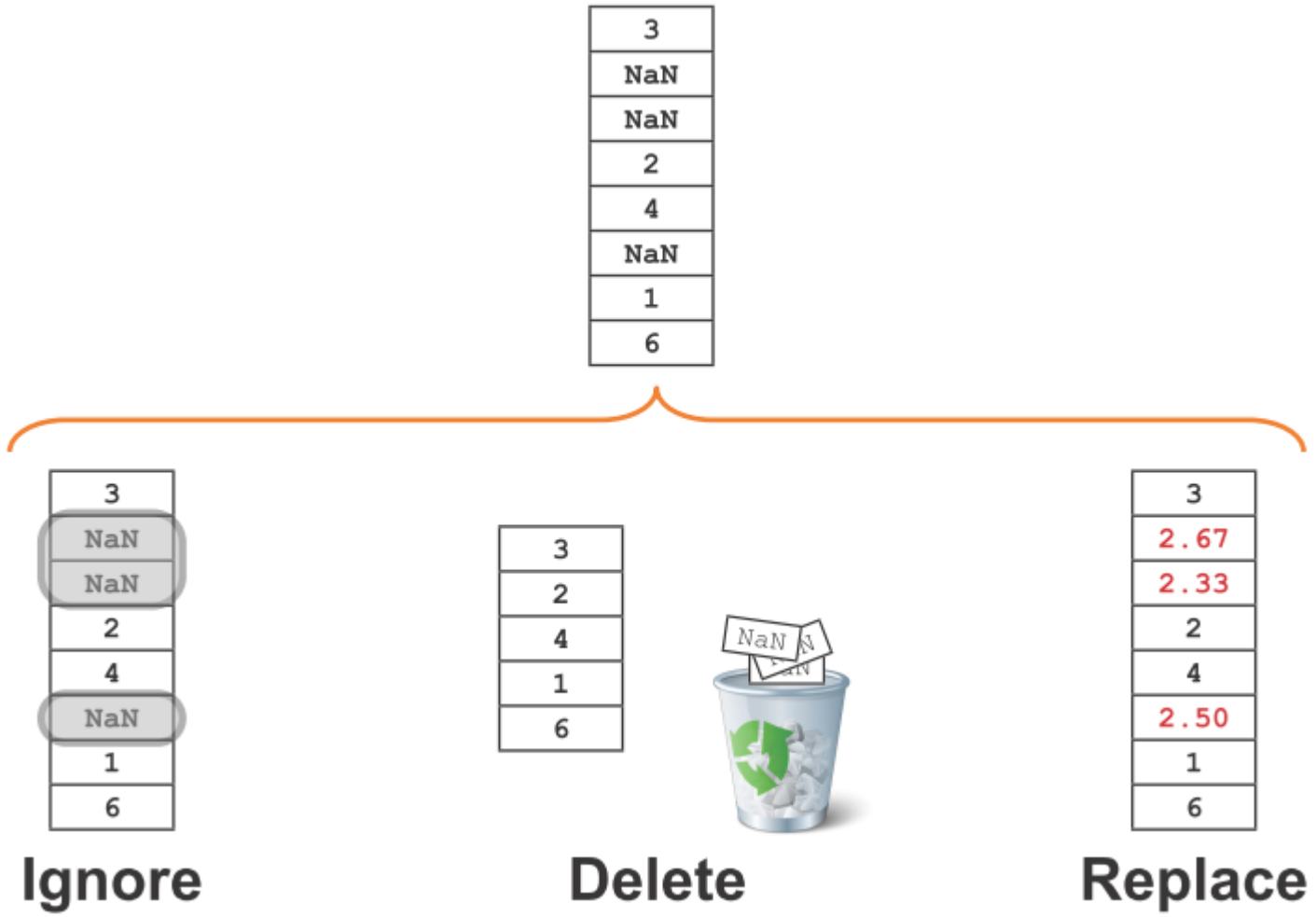


Importing Data Programmatically

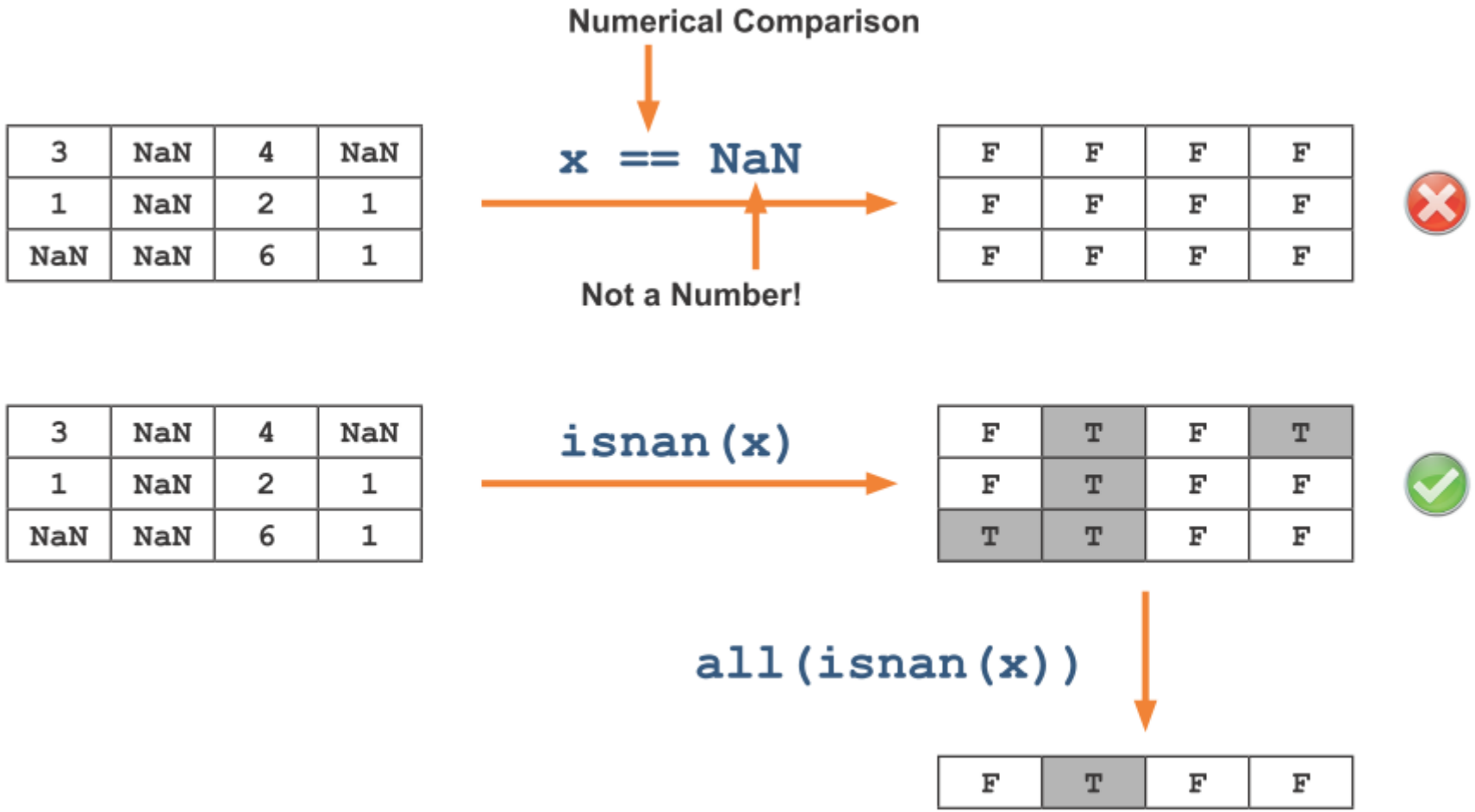




Dealing with Missing Data



Locating Missing Values



Removing Missing Values

NaN	NaN
1	4
6	NaN
3	1
2	7

x

rmmissing

1	4
3	1
2	7

x

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

x

```
idx1 = ismissing(x)
```

F	T	F	T
F	T	F	F
T	T	F	F

```
x(idx1) = []
```

3	4
1	1
	6	1

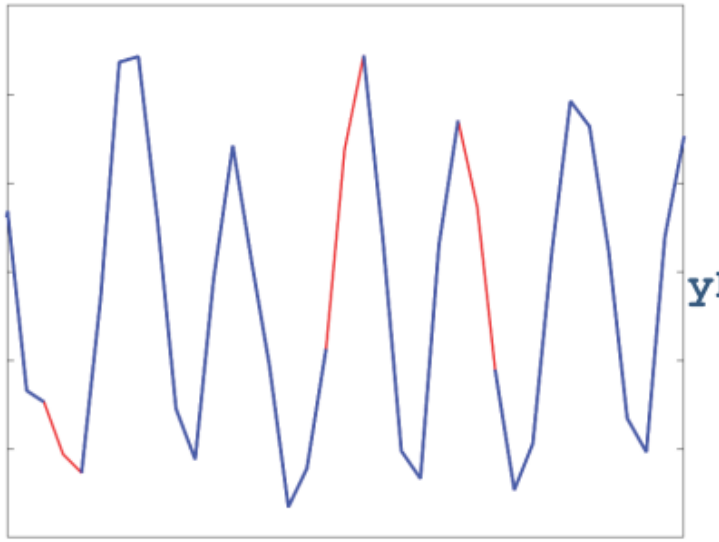
```
idx2 = all(ismissing(x))
```

F	T	F	F

```
x(:,idx2) = []
```

3	4	NaN
1	2	1
NaN	6	1

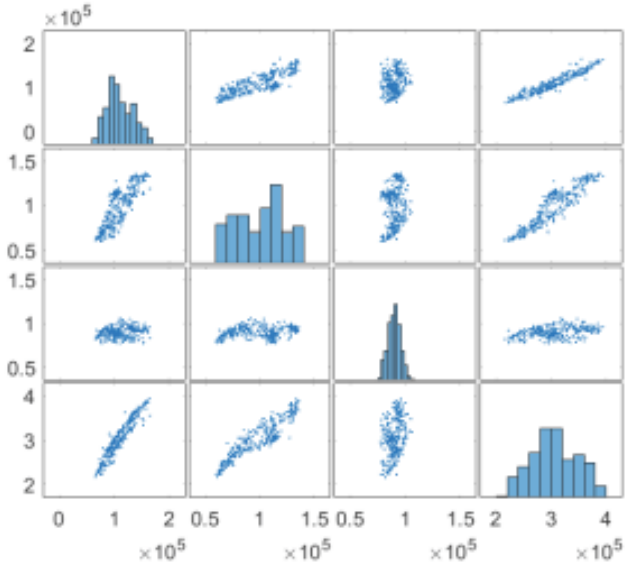
Replacing Missing Values



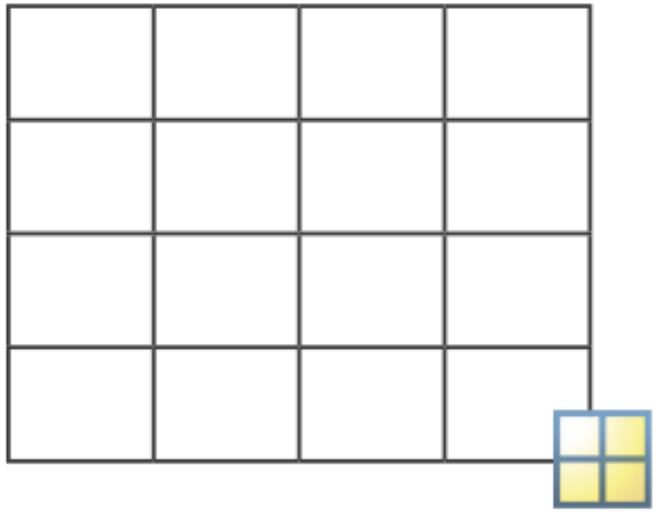
```
y = [3; NaN; NaN; 2; 4; NaN; 1; 6]

yFill = fillmissing(y,"linear")

yFill = [3; 2.67; 2.33; 2; 4; 2.50; 1; 6]
```

Linear Correlation

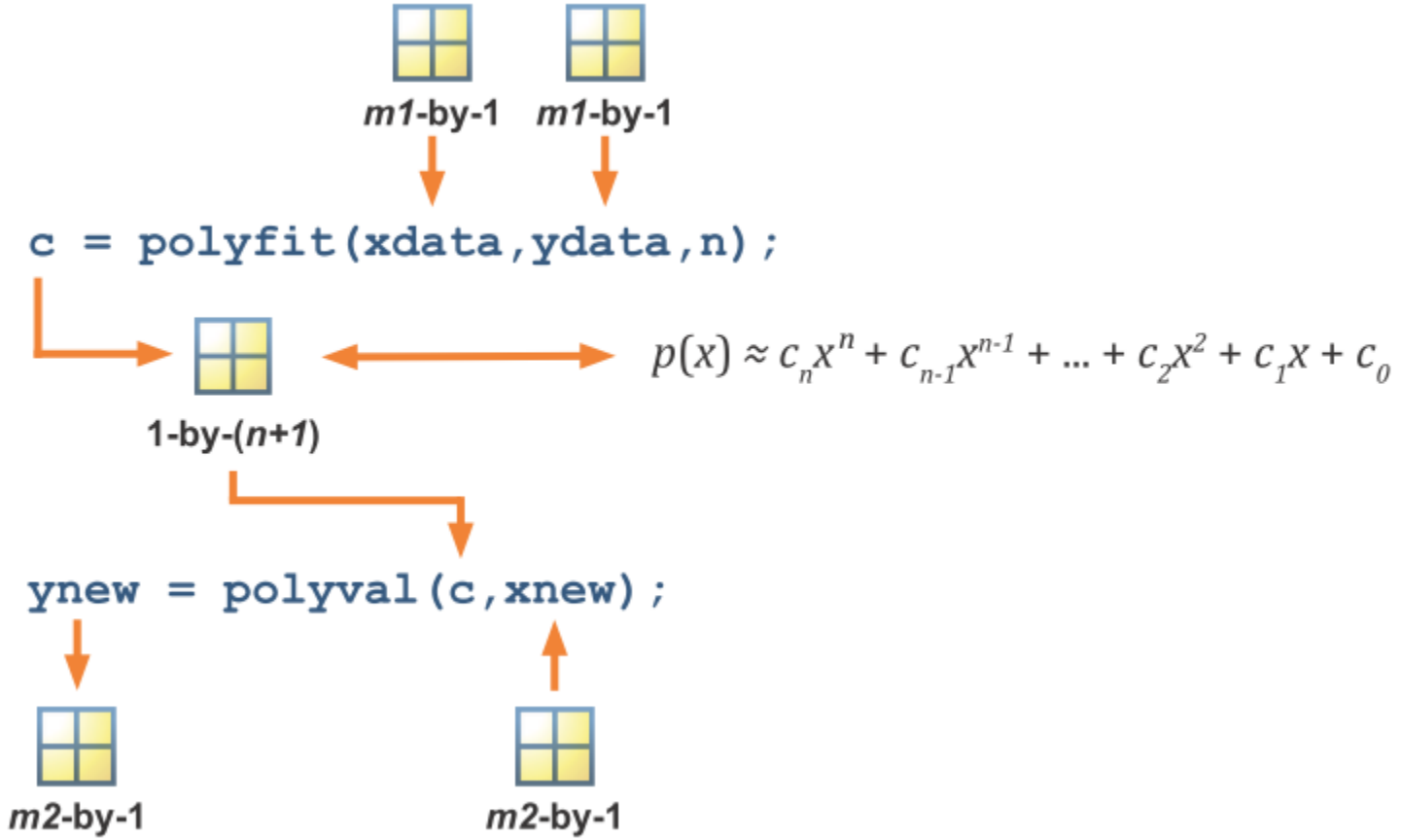
`plotmatrix`



`corrcoef`



Fitting a Polynomial



Chapter 6 Test Your Knowledge

1. **Given 1-by-50 vectors `x` and `y`, what is the result of the command `z = polyfit(x,y,3)`?**

   A. A 1-by-3 vector of points interpolating `y` as a function of `x`

   B. A 1-by-4 vector representing the coefficients of a cubic polynomial fitted to `y` as a function of `x`

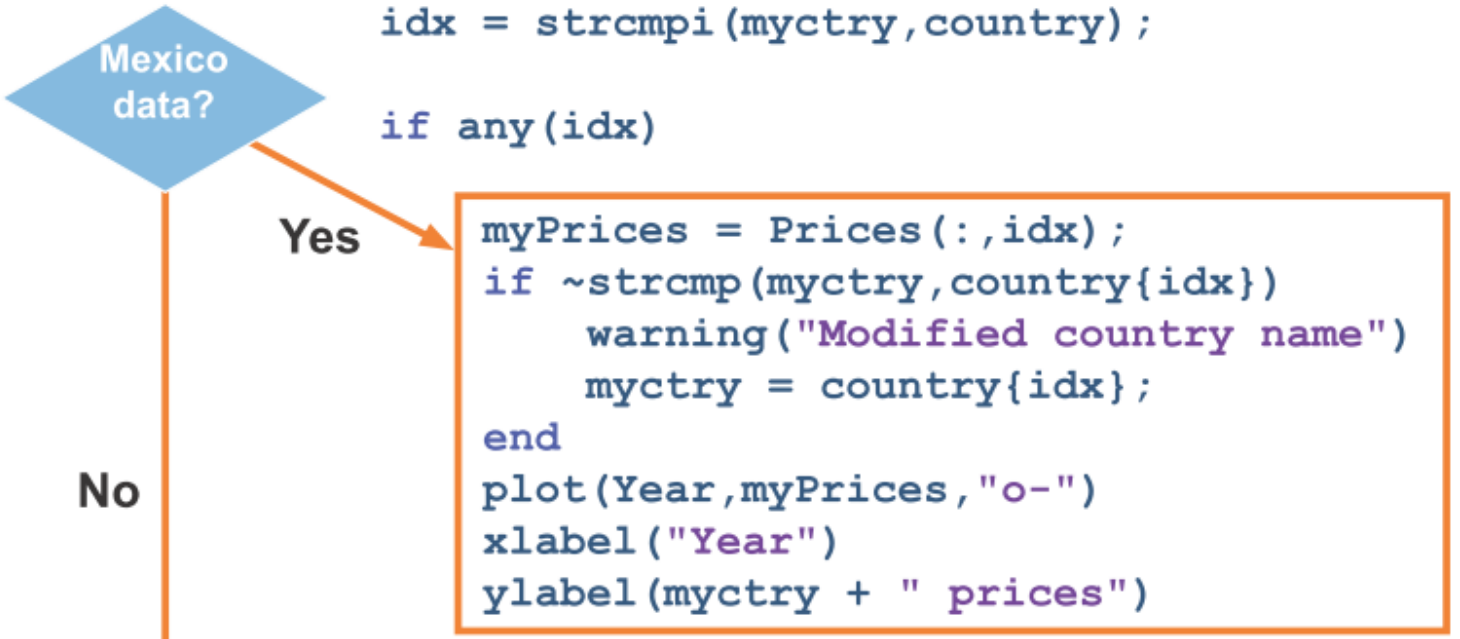
   C. A 1-by-50 vector of the values of a cubic polynomial fitted to `y` as a function of `x`

   D. An error message

Course Outline

- Working with the MATLAB User Interface
- Variables, Commands, and Plots
- Vector, Matrix, and Arrays
- Table of Data
- Conditional Data Selection
- Analysis with Data
- **Increasing Automation with Programming Constructs**
- Appendix: Data Types

Decision Branching



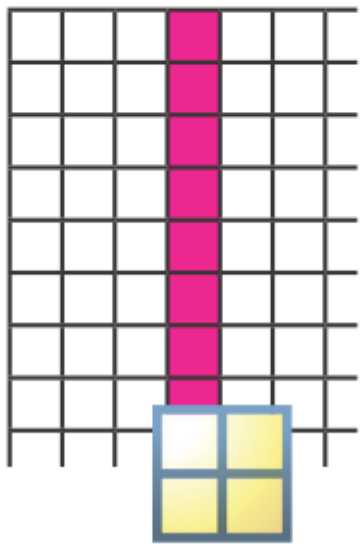

```
myctry = inputdlg("Enter a country:");
myctry = myctry{1};
idx = strcmpi(myctry,country);

if any(idx)
    myPrices = Prices(:,idx);
    if ~strcmp(myctry,country{idx})
        warning("Modified country name")
        myctry = country{idx};
    end
    plot(Year,myPrices,"o-")
    xlabel("Year")
    ylabel(myctry + " prices")
else
    error("No data for " + myctry)
end
```

Mutually exclusive options

For-Loops

```
for k = 1:length(country)
    c = polyfit(myPrices,Prices(:,k),1);
    disp(country(k) + "/" + myctry + " slope: " + c(1))
end
```

```
Australia/France slope: 0.69351
Canada/France slope: 0.63727
France/France slope: 1
Germany/France slope: 1.1991
Italy/France slope: 0.93439
Japan/France slope: 0.4621
Mexico/France slope: 0.24761
South Korea/France slope: 0.91372
UK/France slope: 1.1382
USA/France slope: 0.54434
```



Prices

The `break` statement in `for` and `while` loop

- **break terminates the execution of a for or while loop.**
- **Statements in the loop after the break statement do not execute.**
- **In nested loops, break exits only from the loop in which it occurs.**

```
limit = 0.8;
s = 0;

while 1
        tmp = rand;
        if tmp > limit
                break
        end
        s = s + tmp;
end
```

The `continue`

- **`continue` passes control to the next iteration of a for or while loop.**

- It skips any remaining statements in the body of the loop for the current iteration.

```
for n = 1:50
        if mod(n,7)
                continue
        end
        disp(['Divisible by 7: ' num2str(n)])
end
```

The switch, case, and otherwise

- switch *switch_expression*, case *case_expression*, end

- *For numbers, case_expression == switch_expression.*

- *For character vectors, strcmp(case_expression,switch_expression) == 1.*

- *For a cell array case_expression, at least one of the elements of the cell array matches switch_expression, as defined above for numbers, character vectors, and objects.*

```
x = [12 64 24];

plottype = 'pie3';


switch plottype
        case 'bar'
                bar(x)
                title('Bar Graph')
        case {'pie','pie3'}
                pie3(x)
                title('Pie Chart')
        otherwise
                warning(No plot created.')
end
```

Q & A

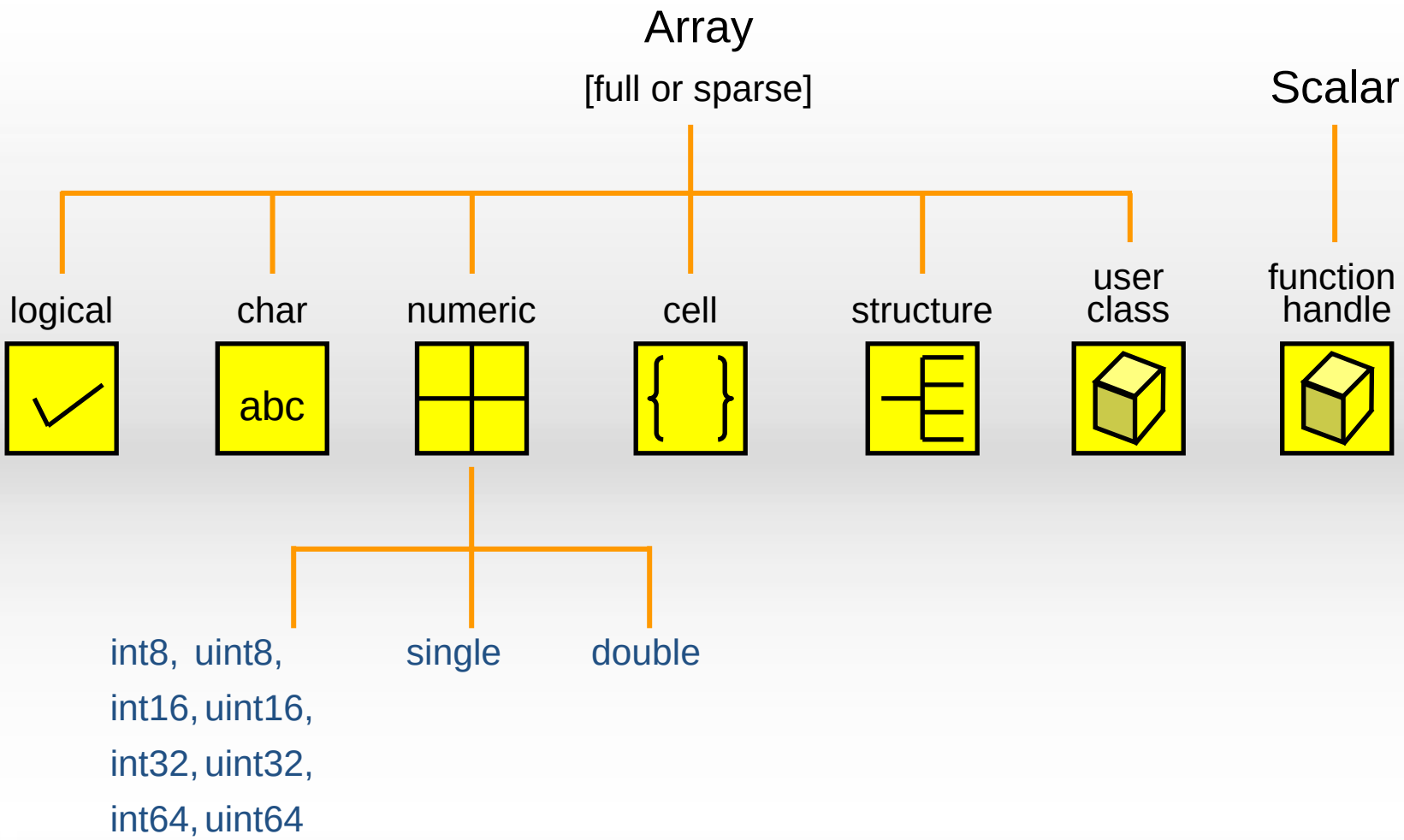
Thank you very much

Have a good time

Course Outline

- Working with the MATLAB User Interface
- Variables, Commands, and Plots
- Vector, Matrix, and Arrays
- Table of Data
- Conditional Data Selection
- Analysis with Data
- Increasing Automation with Programming Constructs
- ➢ **Appendix: Data Types**

MATLAB® Data Types

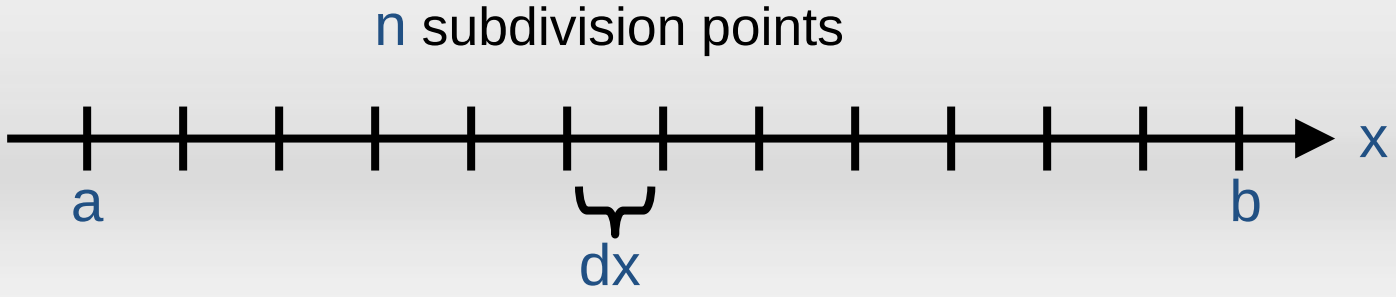


Creating Vectors

Arbitrary data

```
>> x = [2,3,5,7,11,13];
```

Equally-spaced values




```
>> x = a:dx:b;
```

```
>> x = linspace(a,b,n);
```

Creating Matrices

```
>> A = [1,2,3; 4,5,6; 7,8,9];
```

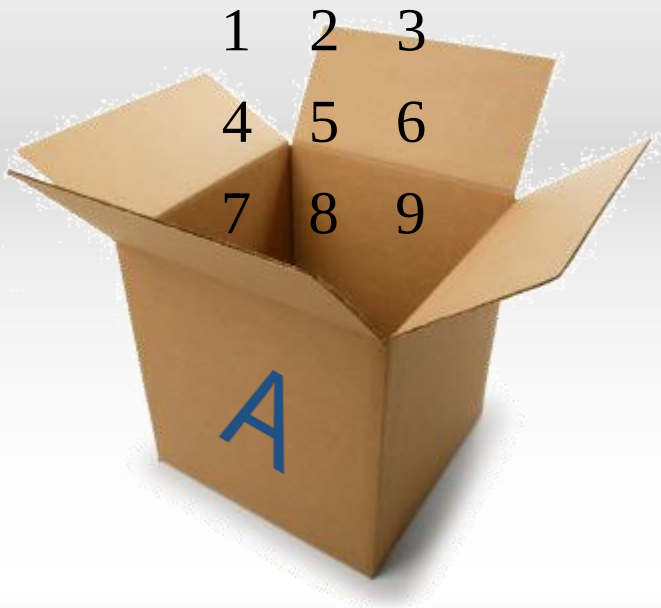
or

```
>> A = [1 2 3; 4 5 6; 7 8 9];
```

or

```
>> A = [1 2 3
4 5 6
7 8 9]
```

} data entry mode



Concatenation

```
>> A
A =
     1  2  3
     4  5  6
     7  8  9

>> B
B =
     1  0  0
     0  1  0
     0  0  1
```

```
>> C = [A,B]
C =
     1  2  3  1  0  0
     4  5  6  0  1  0
     7  8  9  0  0  1
```

```
>> D = [A;B]
D =
     1  2  3
     4  5  6
     7  8  9
     1  0  0
     0  1  0
     0  0  1
```

!!!Please notice the dimension

Row, Column Indexing

```
>> gpdata(1,2)
```

```
>> gpdata(3,6)
```

```
>> gpdata(2,end)
```

Variable Editor - gpdata

File Edit View Graphics Debug Desktop Window Help

Stack: Base No valid plots for: gpdata(...

gpdata <19x11 double>

	1	2	3	4	5	6	7	8	9	10	11	12
1	1990	NaN	1.8700	3.6300	2.6500	4.5900	3.1600	1	2.0500	2.8200	1.1600	
2	1991	1.9600	1.9200	3.4500	2.9000	4.5000	3.4600	1.3000	2.4900	3.0100	1.1400	
3	1992	1.8900	1.7300	3.5600	3.2700	4.5300	3.5800	1.5000	2.6500	3.0600	1.1300	
4	1993	1.7300	1.5700	3.4100	3.0700	3.6800	4.1600	1.5600	2.8800	2.8400	1.1100	
5	1994	1.8400	1.4500	3.5900	3.5200	3.7000	4.3600	1.4800	2.8700	2.9900	1.1100	
6	1995	1.9500	1.5300	4.2600	3.9600	4	4.4300	1.1100	2.9400	3.2100	1.1500	
7	1996	2.1200	1.6100	4.4100	3.9400	4.3900	3.6400	1.2500	3.1800	3.3400	1.2300	
8	1997	2.0500	1.6200	4	3.5300	4.0700	3.2600	1.4700	3.3400	3.8300	1.2300	
9	1998	1.6300	1.3800	3.8700	3.3400	3.8400	2.8200	1.4900	3.0400	4.0600	1.0600	
10	1999	1.7200	1.5200	3.8500	3.4200	3.8700	3.2700	1.7900	3.8000	4.2900	1.1700	
11	2000	1.9400	1.8600	3.8000	3.4500	3.7700	3.6500	2.0100	4.1800	4.5800	1.5100	
12	2001	1.7100	1.7200	3.5100	3.4000	3.5700	3.2700	2.2000	3.7600	4.1300	1.4600	
13	2002	1.7600	1.6900	3.6200	3.6700	3.7400	3.1500	2.2400	3.8400	4.1600	1.3600	
14	2003	2.1900	1.9900	4.3500	4.5900	4.5300	3.4700	2.0400	4.1100	4.7000	1.5900	
15	2004	2.7200	2.3700	4.9900	5.2400	5.2900	3.9300	2.0300	4.5100	5.5600	1.8800	
16	2005	3.2300	2.8900	5.4600	5.6600	5.7400	4.2800	2.2200	5.2800	5.9700	2.3000	
17	2006	3.5400	3.2600	5.8800	6.0300	6.1000	4.4700	2.3100	5.9200	6.3600	2.5900	
18	2007	3.8500	3.5900	6.6000	6.8800	6.7300	4.4900	2.4000	6.2100	7.1300	2.8000	
19	2008	4.4500	4.0800	7.5100	7.7500	7.6300	5.7400	2.4500	5.8300	7.4200	3.2700	
20												

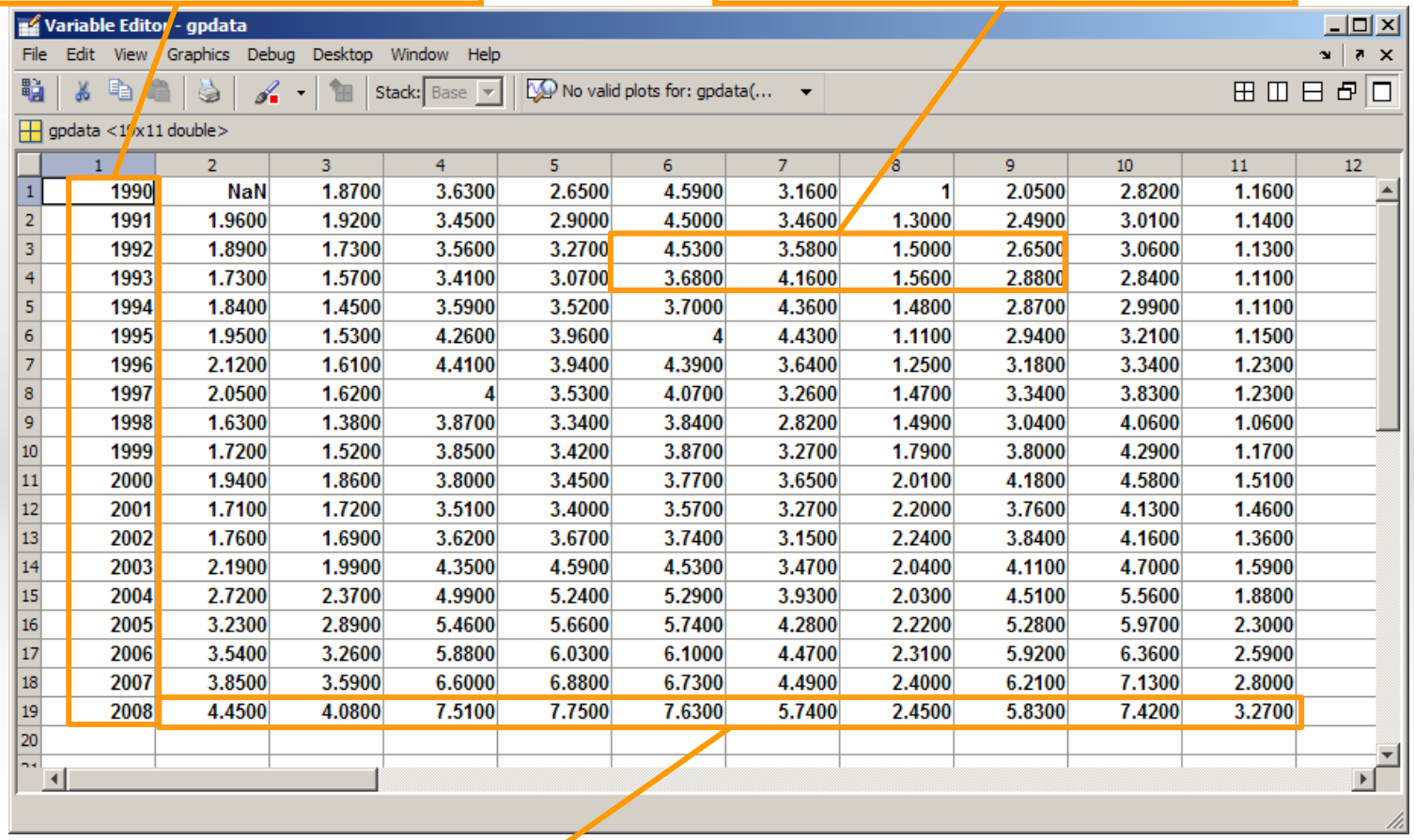
```
>> gpdata(end,2)
```

```
>> gpdata(end,end)
```

Multiple Row, Column Indices

```
>> Year = gpdata(:,1)
```

```
>> gpdata([3,4],6:9)
```



	1	2	3	4	5	6	7	8	9	10	11	12
1	1990	NaN	1.8700	3.6300	2.6500	4.5900	3.1600	1	2.0500	2.8200	1.1600	
2	1991	1.9600	1.9200	3.4500	2.9000	4.5000	3.4600	1.3000	2.4900	3.0100	1.1400	
3	1992	1.8900	1.7300	3.5600	3.2700	4.5300	3.5800	1.5000	2.6500	3.0600	1.1300	
4	1993	1.7300	1.5700	3.4100	3.0700	3.6800	4.1600	1.5600	2.8800	2.8400	1.1100	
5	1994	1.8400	1.4500	3.5900	3.5200	3.7000	4.3600	1.4800	2.8700	2.9900	1.1100	
6	1995	1.9500	1.5300	4.2600	3.9600	4	4.4300	1.1100	2.9400	3.2100	1.1500	
7	1996	2.1200	1.6100	4.4100	3.9400	4.3900	3.6400	1.2500	3.1800	3.3400	1.2300	
8	1997	2.0500	1.6200	4	3.5300	4.0700	3.2600	1.4700	3.3400	3.8300	1.2300	
9	1998	1.6300	1.3800	3.8700	3.3400	3.8400	2.8200	1.4900	3.0400	4.0600	1.0600	
10	1999	1.7200	1.5200	3.8500	3.4200	3.8700	3.2700	1.7900	3.8000	4.2900	1.1700	
11	2000	1.9400	1.8600	3.8000	3.4500	3.7700	3.6500	2.0100	4.1800	4.5800	1.5100	
12	2001	1.7100	1.7200	3.5100	3.4000	3.5700	3.2700	2.2000	3.7600	4.1300	1.4600	
13	2002	1.7600	1.6900	3.6200	3.6700	3.7400	3.1500	2.2400	3.8400	4.1600	1.3600	
14	2003	2.1900	1.9900	4.3500	4.5900	4.5300	3.4700	2.0400	4.1100	4.7000	1.5900	
15	2004	2.7200	2.3700	4.9900	5.2400	5.2900	3.9300	2.0300	4.5100	5.5600	1.8800	
16	2005	3.2300	2.8900	5.4600	5.6600	5.7400	4.2800	2.2200	5.2800	5.9700	2.3000	
17	2006	3.5400	3.2600	5.8800	6.0300	6.1000	4.4700	2.3100	5.9200	6.3600	2.5900	
18	2007	3.8500	3.5900	6.6000	6.8800	6.7300	4.4900	2.4000	6.2100	7.1300	2.8000	
19	2008	4.4500	4.0800	7.5100	7.7500	7.6300	5.7400	2.4500	5.8300	7.4200	3.2700	
20												

```
>> gp08 = gpdata(end,2:end)
```

Linear Indexing

```
>> A = magic(5)
A =
```

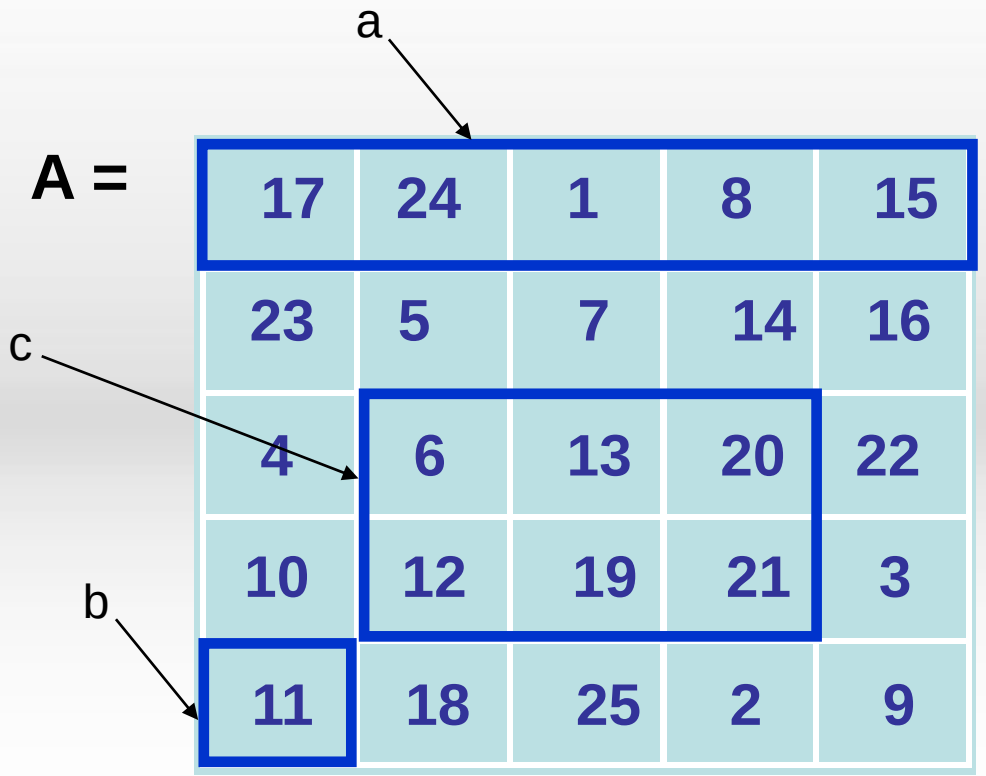
1	17	6	24	11	1	16	8	21	15
2	23	7	5	12	7	17	14	22	16
3	4	8	6	13	13	18	20	23	22
4	10	9	12	14	19	19	21	24	3
5	11	10	18	15	25	20	2	25	9

Indices Data

end

Practice

- `A = magic(5)`
- `a=A(?)`
- `b=A(?)`
- `c=A(?)`



Integer Arrays

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

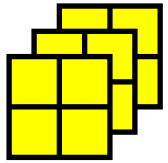
Construct

```
>> A = uint8([1, 2; 3, 4]);
```

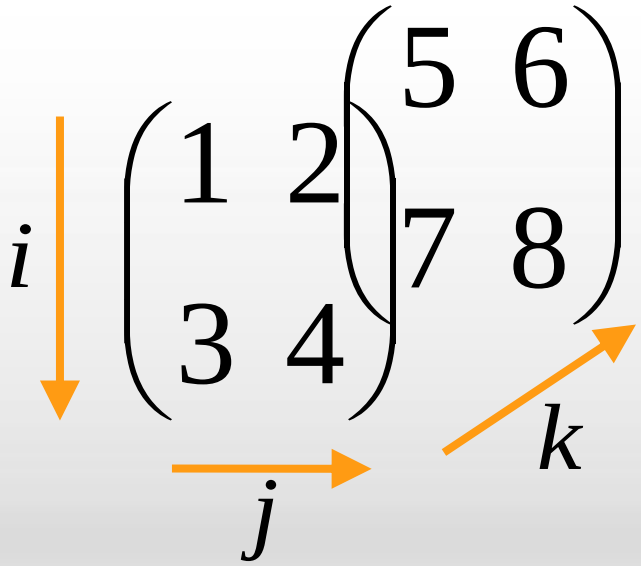
Access

```
>> A(:,2)
ans =
   2
   4
```

Multidimensional Arrays



`A(i,j,k)`



Construct

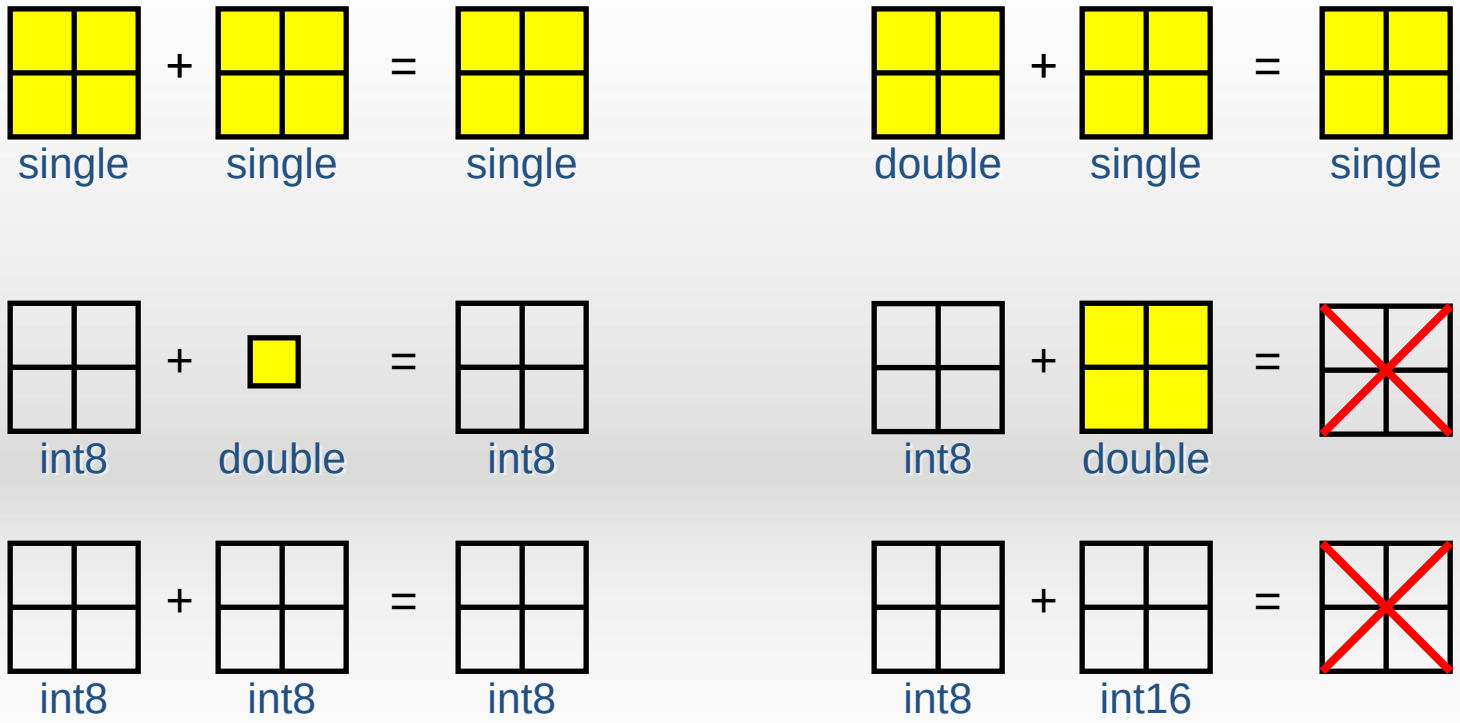
```
>> A(:,:,2) = [5, 6; 7, 8];
>> B = rand(2,2,3);
```

Access

```
>> A(:,2,1)
ans =
     2
     4
```

back

Nondouble Arithmetic



Characters and Strings

back

```
>> y = x
>> y = 'x'
```

variable

character

```
>> MarkA = 'Friends, Romans, countrymen, lend me your ears';
```

1-by-46 char array

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
'F'	'r'	'i'	'e'	'n'	'd'	's'	','	' '	'R'	'o'	'm'	'a'	'n'	's'	','	' '	'c'

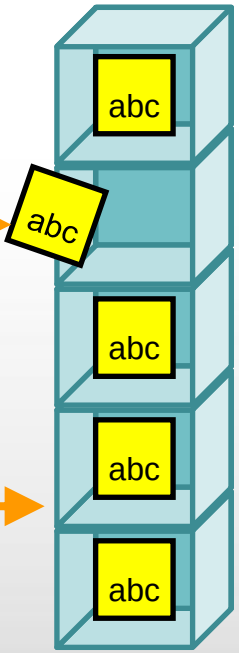
```
>> FriendNationality = MarkA(10:15)
```

Cell Arrays

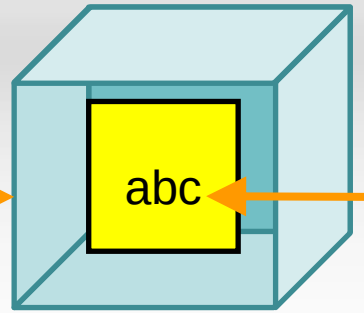
```
>> x = {'Optima'
       'Corolla'
       'Civic'
       'Focus'
       'Legacy'}
```

5-by-1 cell

1-by-7 char




```
>> y = x(3)
```




```
>> y = x{3}
```

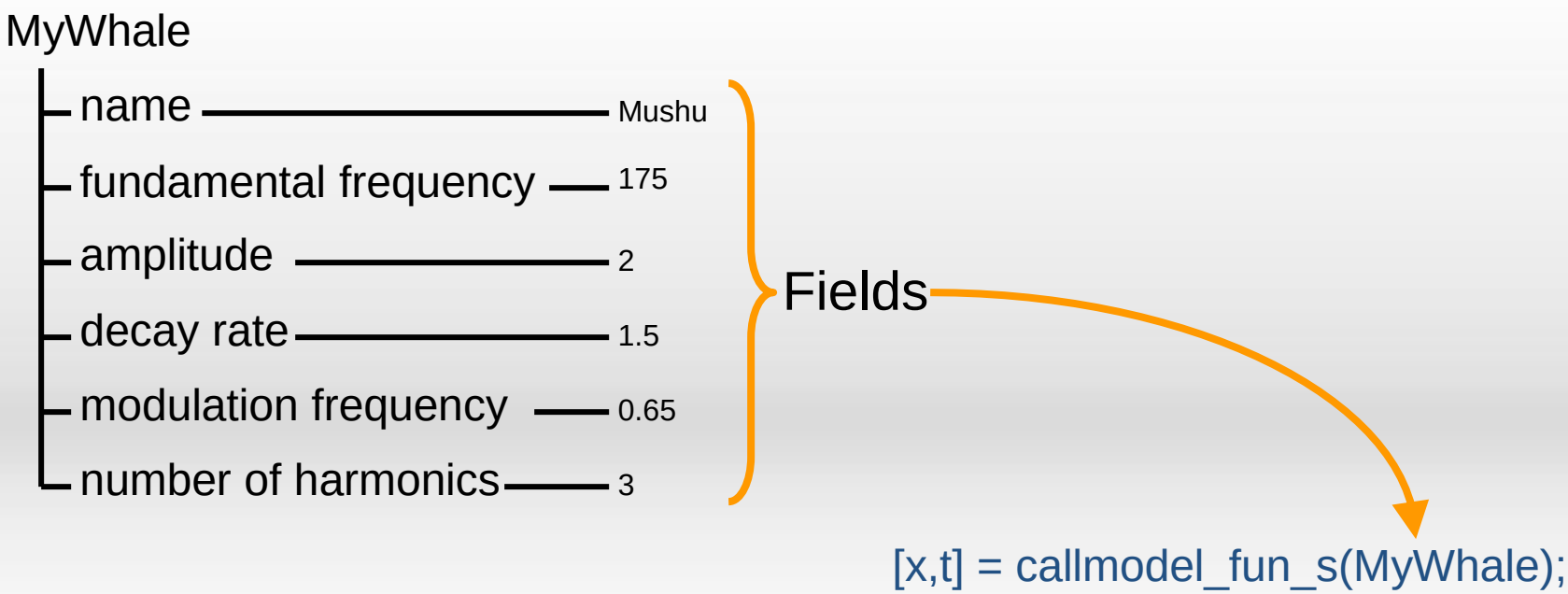
Workspace

Name	Value	Size	Class
x	<5x1 cell>	5x1	cell
y	<1x1 cell>	1x1	cell

Workspace

Name	Value	Size	Class
x	<5x1 cell>	5x1	cell
y	'Civic'	1x5	char

Structures


MyWhale
name
Mushu
fundamental frequency
175
amplitude
2
decay rate
1.5
modulation frequency
0.65
number of harmonics
3
Fields
[x,t] = callmodel_fun_s(MyWhale);

Structure Arrays

```
>> aquarium

aquarium =
1x3 struct array with fields:
    name
    fundamentalfreq
    amplitude
    decayrate
    modulationfreq
    harmonics
```

aquarium(1)
- name — Mushu
- fundamental frequency — 175
- amplitude — 2
- decay rate — 1.5
- modulation frequency — 0.65
- number of harmonics — 3

aquarium(2)
- name — Willy
- fundamental frequency — 180
- 2.5
- 1.9
- ency — 0.5
- onics — 5

aquarium(3)
- name — Shamu
- fundamental frequency — 165
- 1
- 0.8
- ency — 1.0
- onics — 4

Indexing into Structure Arrays