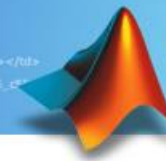




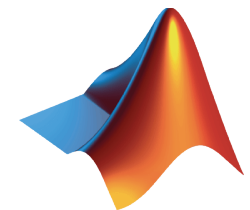
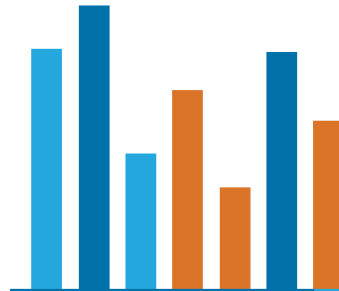
# MATLAB for Data Analysis

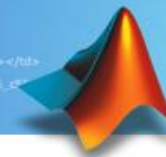
Application Engineer  
Jeffrey Liu



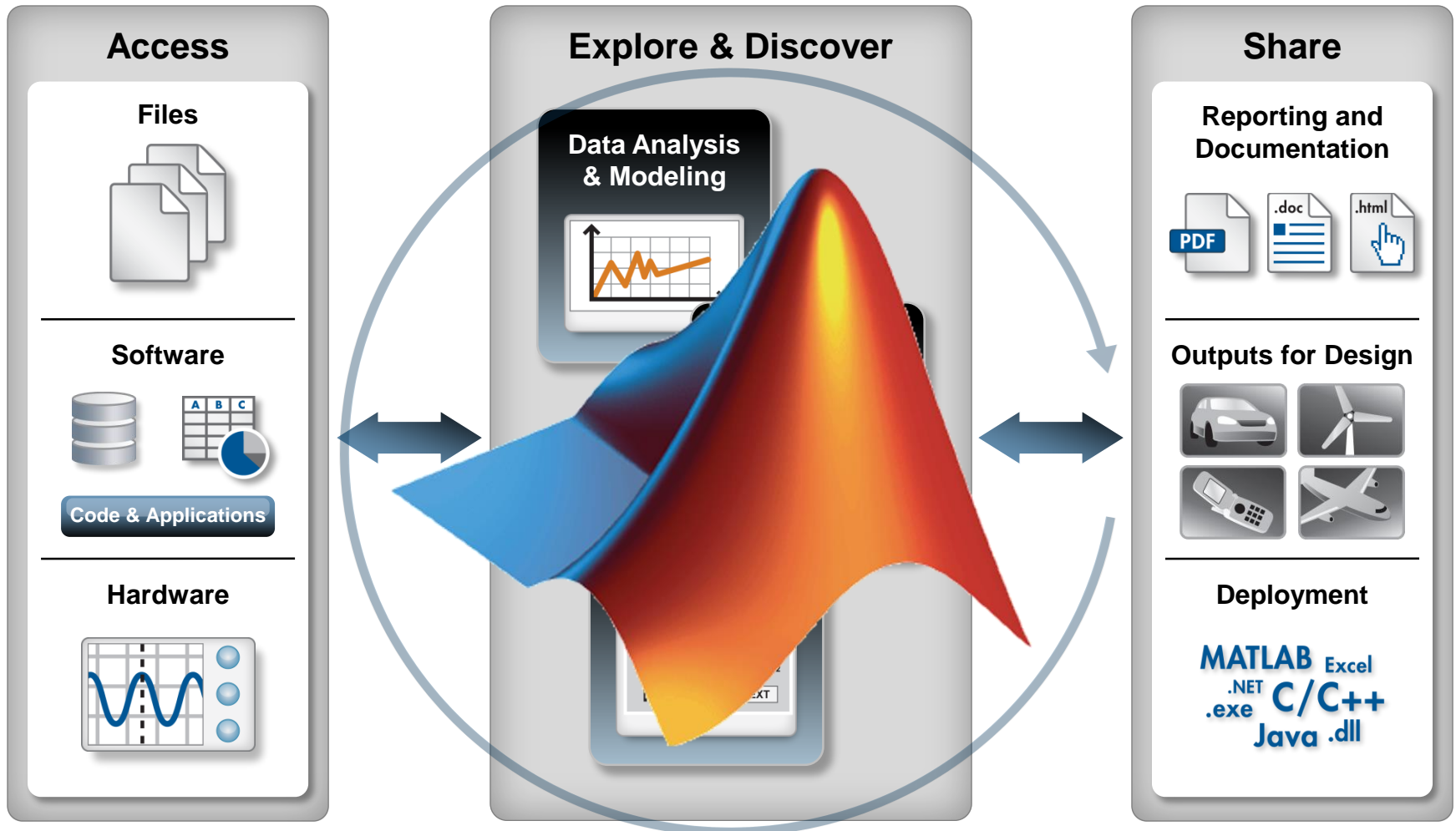
# Outline

- Importing Data to MATLAB
  - Create data in MATLAB
  - Working with data files
- Basic data analysis
  - Descriptive statistics
  - Split-apply workflow
- Data Types
  - Dataset, table
  - data types
- Fit data to models
  - Curve fitting toolbox
  - Statistics models
- Data pre-processing
  - Dealing with missing value
  - Locating data
  - Merging data
- Publish your code

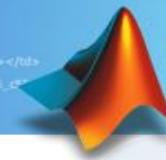




# Technical Computing Workflow



**Automate**



# Random Number generation

## ■ MATLAB

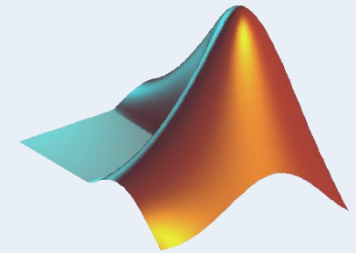
- **rand**
  - uniform distribution generation between 0~1
- **randi**
  - Uniformly distributed pseudorandom integers
- **randn**
  - Standard normal distribution generator

## ■ Statistics Toolbox

## ■ Random seed

- **rng**
- **rng shuffle**

MATLAB®



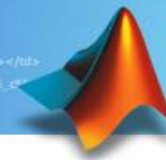
**rand**  
**randi**  
**randn**

Statistics Toolbox™

**betarnd**  
**binornd**  
**chi2rnd**  
**ncx2rnd**  
**exprrnd**  
**evrnd**  
**frnd**  
**ncfrnd**  
**gamrnd**

random





# Interactively Importing Text Files and Spreadsheets

- The Import Tool attempts to interpret the contents of spreadsheets and delimited text files, and automatically set several options.

Import - C:\class\coursefiles\mlbe\gasprices\gasprices.txt

IMPORT VIEW

Delim: ① Tab Column delimiters: Range: A4:K22 ② Variable Names Row: 3

③ Column vector

④ Replace unimportable cells with NaN

DELIMITERS SELECTION IMPORTED DATA UNIMPORTABLE CELLS IMPORT

gasprices.txt x

	A Year	B Australia	C Canada	D France	E Germany	F Italy	G Japan	H Mexico	I SouthKo...	J UK	K USA
1	Average ...										
2	\$US per ...										
3	Year	Australia	Canada	France	Germany	Italy	Japan	Mexico	South Ko...	UK	USA
4	1990	1.87	1.87	3.63	2.65	4.59	3.16	1.00	2.05	2.82	1.16
5	1991	1.96	1.92	3.45	2.90	4.50	3.46	1.30	2.49	3.01	1.14
6	1992	1.89	1.73	3.56	3.27	4.53	3.58	1.50	2.65	3.06	1.13

# Reading Fixed structure files

## Files

- Excel, text, csv, or binary
- .xpt in SAS
- .mat
- Multimedia, scientific
- Web, XML

## Command

- load
- csvread
- xlsread
- textread
- textscan

Import - C:\Users\Jeffrey\Desktop\Jeffrey\Ae question\stepwise reg\Test data v2.xls

Range: A2:T8461  
Variable Names Row: 1

Column vectors  
Matrix  
Cell Array

Replace  
unimportable cells with NaN

Import Selection

Test data v2.xls

	Y1	Y2	X11	X12	X13	X14	X15	X16	X21	X22	X23	X24	X25
1	Y1	Y2	X11	X12	X13	X14	X15	X16	X21	X22	X23	X24	X25
2	183	185.2000	33.6000	36.6000	36.2000	37	16.8000	10.2000	370	370	350	370	
3	206.5000	146.1000	34.4000	31.8000	32.2000	35.4000	16.8000	12.8000	400	400	370	260	
4	160.5000	170	44.4000	41.2000	36.4000	44	16.2000	14.8000	320	310	320	280	
5	171.1000	156	36.6000	39	37.4000	40.8000	15.4000	15.4000	370	320	350	300	
6	168.3000	145.1000	37.6000	32	44.6000	436 Converted ToType: Number, Value: 36			300	370	270	250	
7	182.8000	215.1000	44.6000	43	39.2000	36	14.6000	13.6000	300	350	260	320	
8	197.5000	186.2000	38.2000	40	41.8000	34.2000	14.2000	13.8000	340	310	310	350	
9	223.2000	198.9000	35.4000	37.8000	31.6000	36.2000	15	19.6000	380	380	360	360	
10	173.1000	249.8000	40.4000	39.4000	36.2000	29.6000	15.4000	16.6000	210	310	400	420	
11	174.6000	240.1000	36.4000	44.6000	37.6000	39	19.2000	15.6000	310	290	380	250	
12	197	184.1000	34.4000	37.6000	38.6000	36.2000	10	16.4000	330	350	350	300	
13	241	171.4000	35.2000	36.4000	31.8000	34.6000	17.6000	15.2000	390	430	420	390	
14	128.2000	168.8000	37	39.4000	37.2000	42.8000	11.2000	11	320	300	340	290	
15	172.8000	199.1000	32.2000	37.8000	36.4000	35	12	14.4000	400	330	360	380	
16	182.6000	193.2000	32.2000	44	37.4000	35.2000	15	13.8000	400	350	310	270	
17	220	174.1000	43.4000	36.8000	32.8000	42.4000	16.6000	14	330	350	350	320	
18	206.4000	202.7000	31.6000	32.8000	36.8000	37.2000	20.2000	19.8000	390	370	340	310	

Data correlation

csv

txt

dataset

dataset

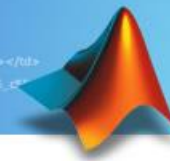
dataset

.xpt

.xls

MATLAB&SIMULINK





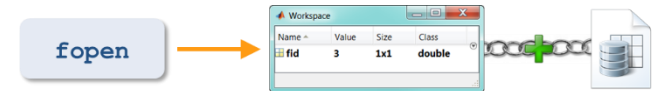
# Low-Level File I/O

## 1. Open the file

```
>> fid = fopen('gasprices.csv','r');
```

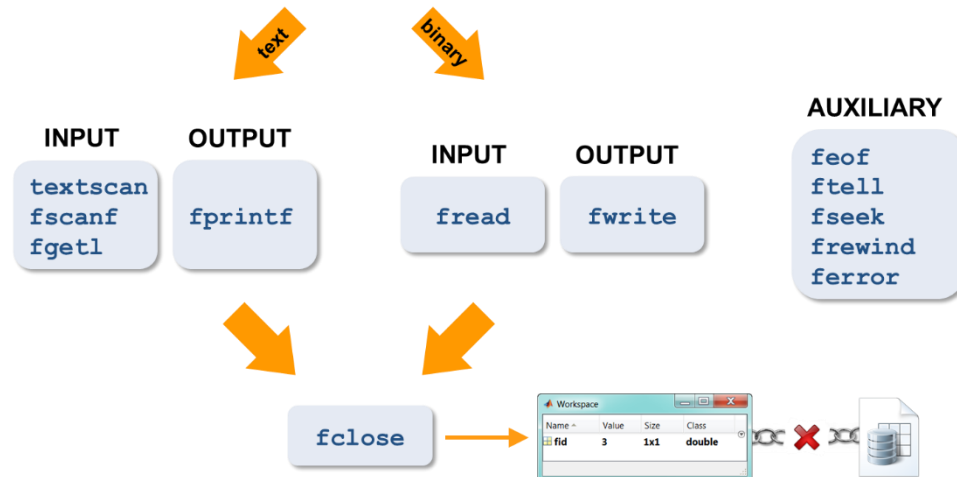
## 2. Perform the file operations

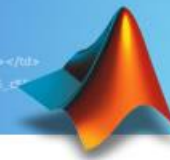
Functions for reading and writing text and binary files are shown in the diagram to the right.



## 3. Close the file

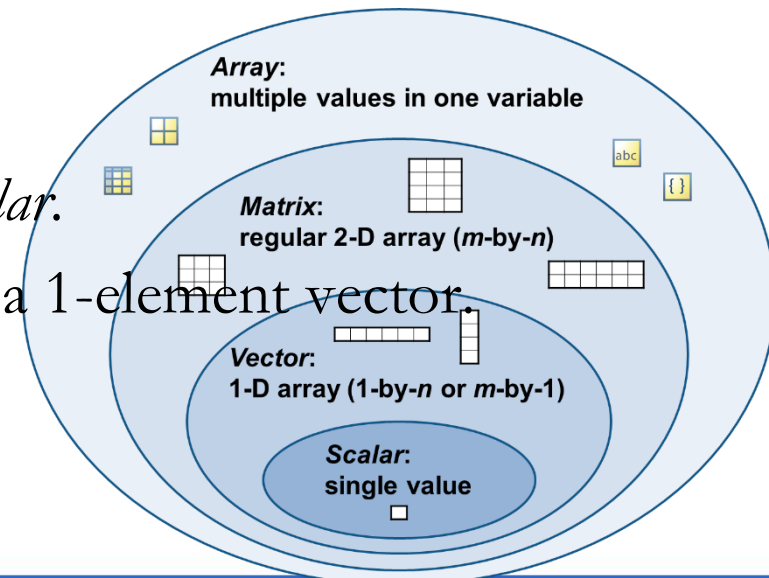
```
>> fid = fclose(fid);
```



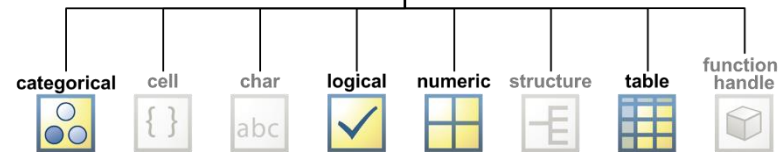


# Vectors, Matrices, and Arrays

- A table of numbers with  $m$  rows and  $n$  columns is referred to as an  $m$ -by- $n$  matrix.
- A *vector* is a single row or a single column of numbers. It is therefore a special case of a matrix, where either  $m$  or  $n$  is equal to 1.
- A single number is referred to as a *scalar*. It is a 1-by-1 matrix and, equivalently, a 1-element vector.



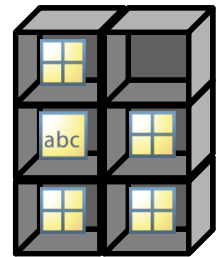




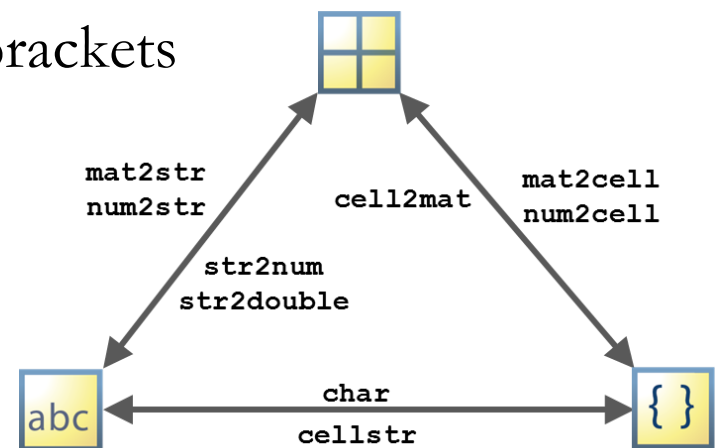
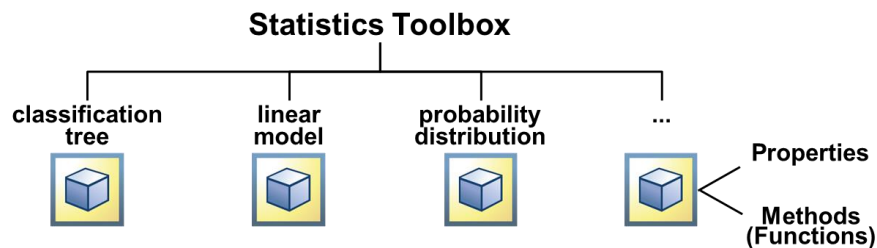
# Cell array

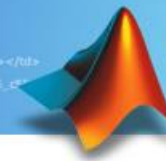
- Cell arrays are an easy way to assemble data of dissimilar types and sizes into a single “container of containers.”

- They are typically used to store strings of different length.



- To concatenate elements into a cell array, use curly braces (`{}`) instead of square brackets





# What Is a Table ?

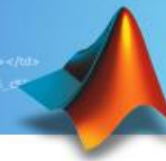
- Many data sets conform to a particular tabular arrangement with the following properties:

- Each column of data has the same type (text, numeric, logical (T/F), etc.).
- Different columns, however, can have different types.
- Each column typically has a unique name.
- Each column has the same number of rows

	A	B	C	D	E	F	G	H	I	J	K	L
1	Date	AAPL	CSCO	DELL	EBAY	HPQ	IBM	INTC	MSFT	ORCL	YHOO	DJIA
2	1/3/2008	0.0005	0.0079	-0.0283	0.0107	0.0000	0.0020	-0.0272	0.0042	0.0272	0.0050	0.0010
3	1/4/2008	-0.0794	-0.0238	-0.0708	-0.0480	-0.0576	-0.0366	-0.0845	-0.0284	-0.0479	-0.0289	-0.0198
4	1/7/2008	-0.0135	0.0004	-0.0383	-0.0279	-0.0330	-0.0107	0.0092	0.0067	0.0099	0.0009	0.0021
5	1/8/2008	-0.0366	-0.0272	-0.0248	-0.0142	-0.0488	-0.0249	-0.0275	-0.0341	-0.0507	-0.0249	-0.0188
6	1/9/2008	0.0465	0.0314	0.0091	-0.0047	0.0285	0.0074	0.0218	0.0292	0.0215	-0.0022	0.0115
7	1/10/2008	-0.0077	0.0000	0.0005	0.0163	0.0200	0.0162	-0.0093	-0.0032	0.0032	0.0656	0.0092
8	1/11/2008	-0.0304	-0.0142	-0.0086	-0.0227	-0.0075	-0.0228	-0.0247	-0.0123	-0.0271	-0.0308	-0.0194
9	1/14/2008	0.0347	0.0161	0.0327	-0.0153	0.0248	0.0525	0.0				0.0135
10	1/15/2008	-0.0560	-0.0169	-0.0260	-0.0448	-0.0237	-0.0107	-0.0				-0.0219
11	1/16/2008	-0.0572	-0.0275	-0.0106	0.0050	-0.0159	-0.0020	-0.1				-0.0028
12	1/17/2008	0.0078	-0.0331	0.0087	0.0011	-0.0307	-0.0052	-0.0				-0.0249
13	1/18/2008	0.0029	-0.0012	0.0105	0.0074	0.0173	0.0225	-0.0				-0.0049
14	1/22/2008	-0.0361	-0.0343	-0.0402	-0.0396	-0.0238	-0.0213	-0.0				-0.0106
15	1/23/2008	-0.1126	0.0232	-0.0020	0.0609	0.0109	0.0471	0.0710	-0.0009	0.0246	0.0075	0.0247
16	1/24/2008	-0.0253	0.0440	0.0426	-0.0627	0.0386	0.0076	0.0349	0.0405	0.0000	0.0806	0.0088
17	1/25/2008	-0.0421	-0.0369	-0.0501	-0.0130	-0.0260	-0.0226	-0.0339	-0.0094	-0.0161	0.0115	-0.0139
18	1/28/2008	0.0000	-0.0041	0.0144	0.0015	0.0014	0.0044	0.0144	-0.0067	-0.0010	-0.0543	0.0144
19	1/29/2008	0.0117	-0.0017	0.0103	-0.0283	-0.0096	0.0106	0.0103	-0.0037	-0.0094	0.0014	0.0078
20	1/30/2008	0.0049	0.0124	-0.0063	0.0069	0.0037	-0.0043	0.0092	-0.0123	0.0099	-0.0884	-0.0030
21	1/31/2008	0.0238	0.0057	-0.0193	0.0222	0.0041	0.0137	0.0196	0.0123	0.0137	0.0068	0.0165
22	2/1/2008	-0.0120	0.0178					0.0313	-0.0682	0.0063	0.3918	0.0073
23	2/4/2008	-0.0158	-0.0459					0.0265	-0.0086	-0.0235	0.0329	-0.0085
24	2/5/2008	-0.0175	-0.0238					0.0523	-0.0378	-0.0477	-0.0120	-0.0297

Each row  
is a set of  
observations

Each column is a  
named variable



# Categorical array

- When text labels intended to represent finite set of possibilities, cell array is unnecessary. Instead, you can use *categorical* array

```
>> x = {'C', 'B', 'C', 'A', 'B', 'A', 'C'};
```

```
>> y = categorical(x);
```

**x**

'C'	'B'	'C'	'A'	'B'	'A'	'C'
-----	-----	-----	-----	-----	-----	-----



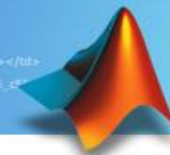
1-by-7

**y**

C	B	C	A	B	A	C
---	---	---	---	---	---	---



1-by-7



# Extracting Data from a Table

- Index into variables in a table using dot (.) notation to reference the variables by name:

```
>> dates = stockPrices.Date;  
>> admPrices = stockPrices.ADM;
```

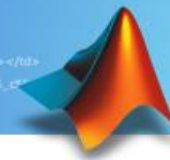
- You can also use curly braces ({} ) to index into variables in a table. You can index numerically

```
>> twoFin = stockPrices{:[4 5]};  
or by name  
>> twoFin = stockPrices{:[ 'ADM', 'ADN' ]};
```

Date	AAL	ABF	ADM	ADN	AGK	...
'1/3/2006'	2241.76	840.50	466.75	132.75	295.74	
'1/4/2006'	2251.65	857.50	481.00	133.00	291.10	
'1/5/2006'	2193.41	855.00	477.25	135.00	292.90	
'1/6/2006'	2235.17	857.00	477.25	135.50	289.03	
'1/9/2006'	2212.09	862.00	478.25	138.00	283.10	

Diagram illustrating data extraction from a table:

- Two columns (Date and ADM) are highlighted with a box, labeled "505-by-1" and containing curly braces {}.
- Two columns (ADM and ADN) are highlighted with a box, labeled "505-by-1" and containing curly braces {}.
- A larger box highlights the entire table, labeled "505-by-93".



# Merging Data from two tables

- Use “join” to merge “dataset” variables

- Import files with dataset array

```
>> Variables = dataset('XLSFile', 'filename');
```

- Merge two datasets

```
>> New = join(A,B, 'Keys', 'date', 'Type', 'outer',  
'MergeKeys',true);
```

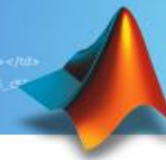
- Use “innerjoin”, “outerjoin” to combine “table” variables

- Import files with table array

```
>> Variables = readtable(date, data, 'var_name');
```

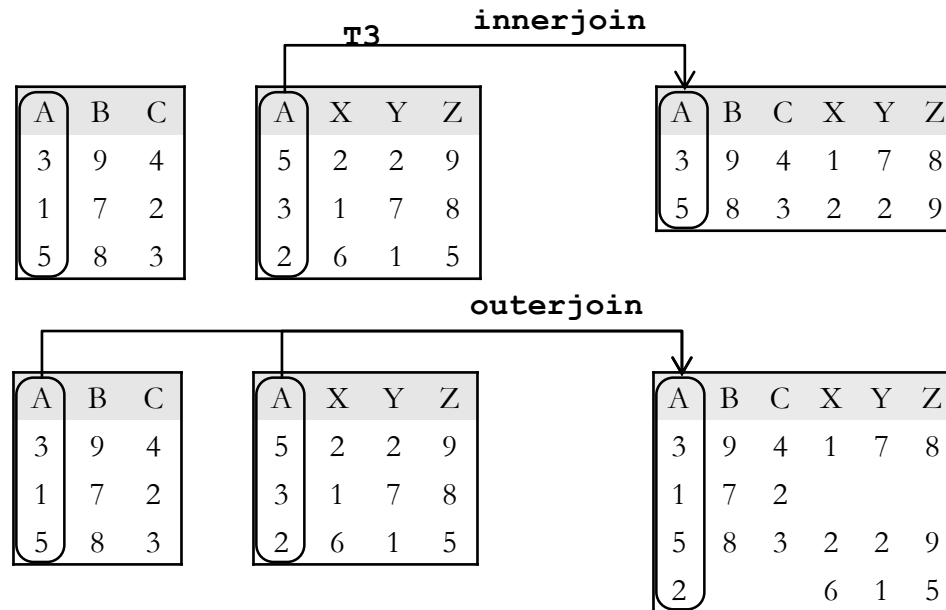
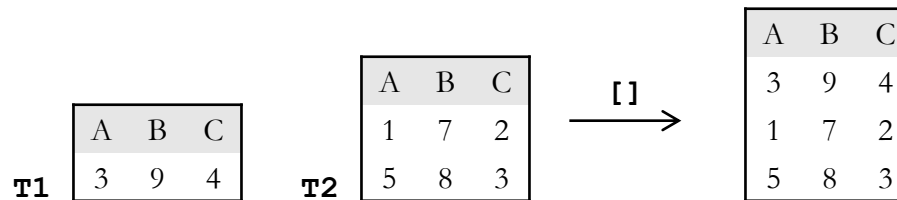
- Merge two tables

✓ innerjoin, outerjoin

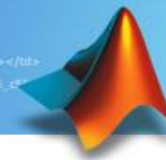


# Exercise 1

- Merging two tables by time in excel file.







# Dealing with missing value

- Possible strategies

3
NaN
NaN
2
4
NaN
1
6

Ignore

3
2
4
1
6

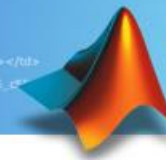


Delete

3
2.67
2.33
2
4
2.50
1
6



Replace



# Avoiding NaNs in Calculations

- Several functions are designed to ignore NaNs in calculations

`nancov`  
`nanmax`  
`nanmean`

`nanmedian`  
`nanmin`  
`nanstd`

`nansum`  
`nanvar`

- if a column contains all NaNs, ignoring them will result in applying the desired function to an empty array.

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

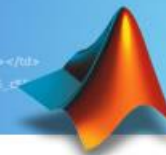
↓  
`nanmean`

2	2	4	1
---	---	---	---

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

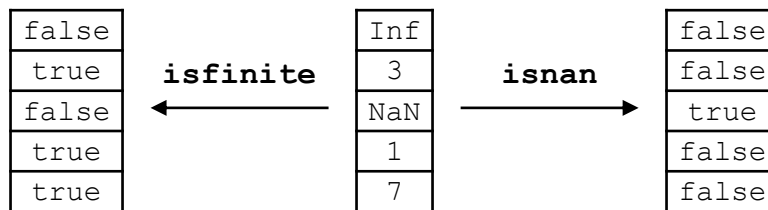
↓  
`nanmean`

2	NaN	4	1
---	-----	---	---



# Locating Missing(Other) Values

- MATLAB provides numerous “is\*” functions that take an array as input and return a logical output that signifies if the input has a certain characteristic.



- You can use logical indexing to remove elements from an array.

```
>> idx = isnan(x);
>> x(idx) = [];
```

3
1
7

**x**



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

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

Numerical comparison

```
>> x == NaN
```

Not a Number!

F	F	F	F
F	F	F	F
F	F	F	F



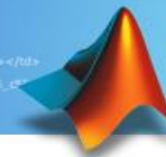
```
>> isnan(x)
```

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

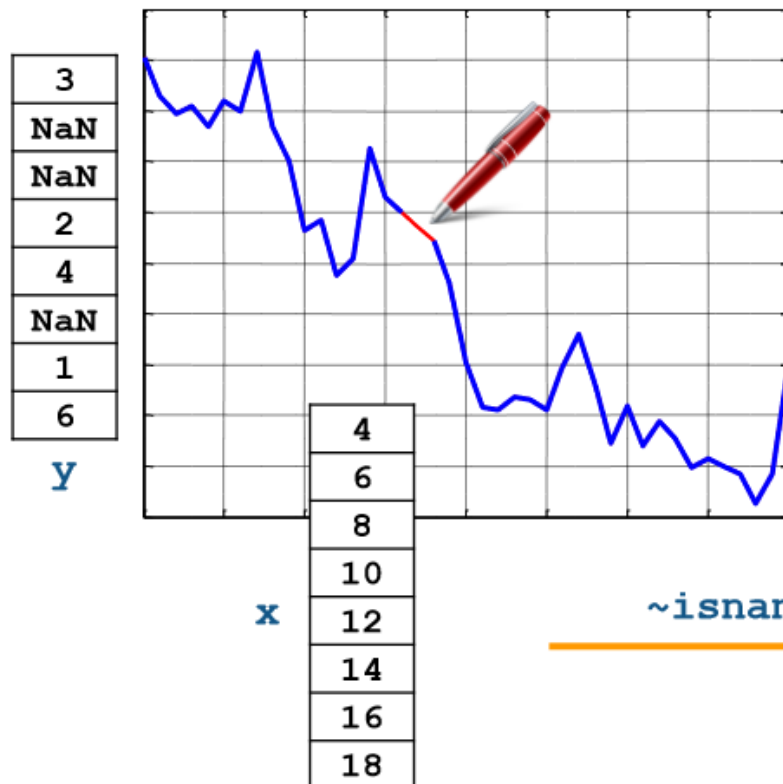


```
>> all(isnan(x))
```

F	T	F	F
---	---	---	---

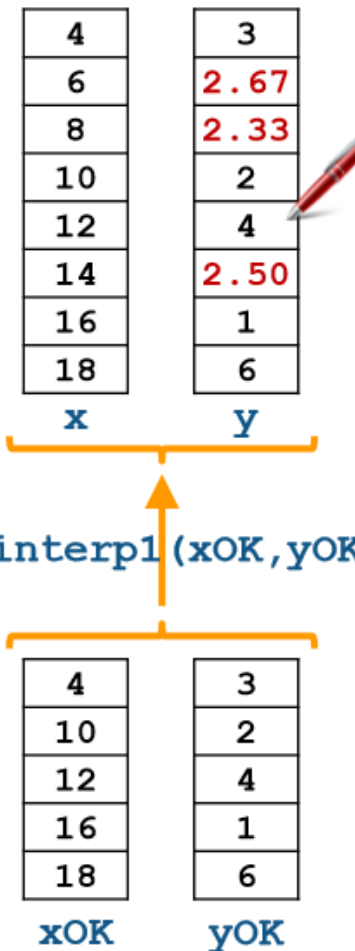


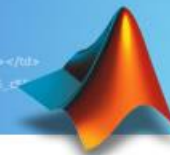
# Replacing Missing Values in Matrices



`~isnan(y)`

`>> y = interp1(xOK,yOK,x);`





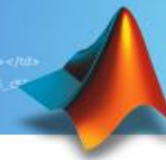
# Test Your Knowledge

1. (Select all that apply.) If  $x$  is a (numeric) vector, which commands will remove all the NaNs from  $x$ ?

- A.  $x = x(\text{isnan}(x));$
- B.  $x = x(x \sim \text{NaN});$
- C.  $x(\text{isnan}(x)) = [];$
- D.  $x(x == \text{NaN}) = [];$
- E.  $x = x(\text{isfinite}(x));$

2. Suppose  $x$  is a 6-by-3 (numeric) matrix of ones. Three values in the first column are missing (NaN), as are six values in the third column. What will `>> nanmean(x)` return?

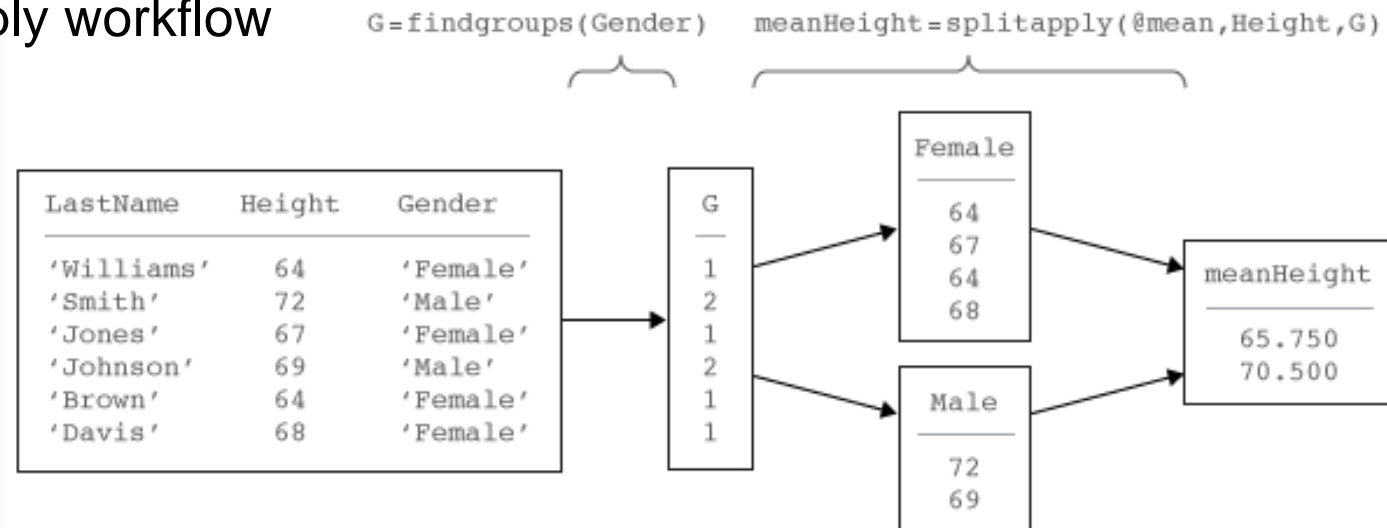
- A. 1
- B.  $[0.5 \ 1 \ 0]$
- C.  $[0.5 \ 1 \ \text{NaN}]$
- D.  $[1 \ 1 \ \text{NaN}]$
- E.  $[1 \ 1 \ 0]$
- F. An error message.



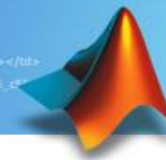
# Calculate the data by group

- Function 'grpstats' : Summary statistics organized by group
  - Syntax : `statarray = grpstats(tbl,groupvar,whichstats)`
    - tbl : data in table or dataset array
    - groupvar : Column name for grouping in tbl
    - whichstats : Types of summary statistics ( numel, std, max, sum...)

- New Split-apply workflow

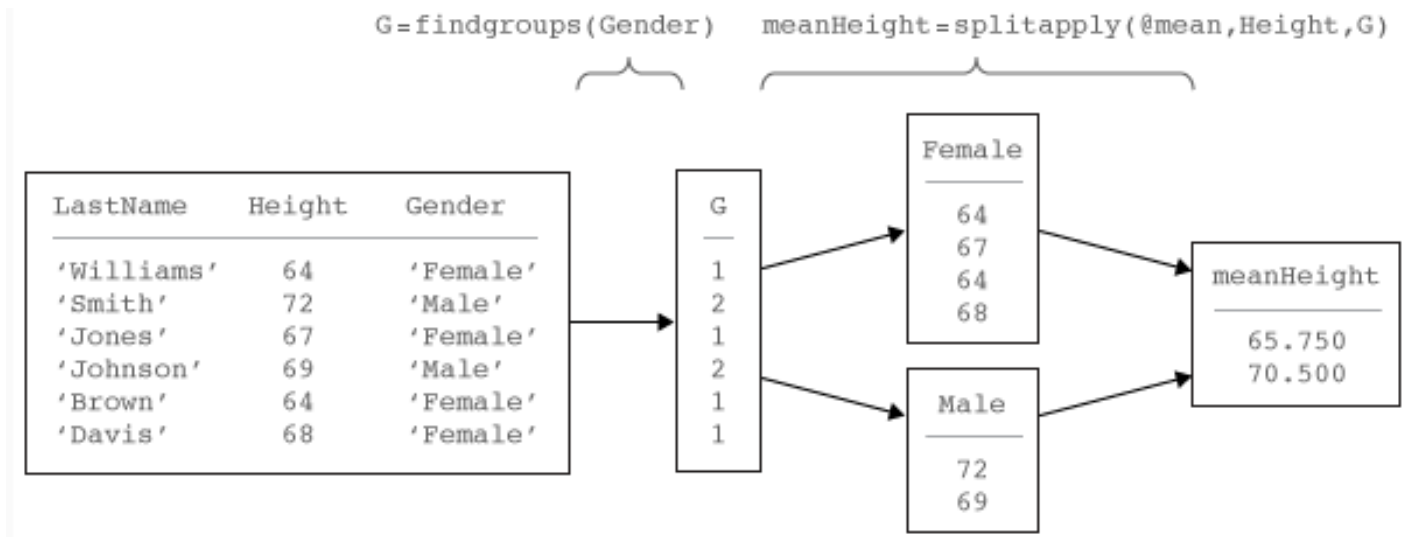


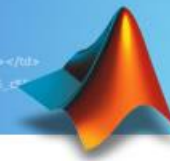




## Exercise 2

- 在bank-full.csv中依照三種類別(工作、婚姻、購買與否)區分客戶，並計算類別中的平均收入





# LinearModel Variables

- Given data in vectors  $x$  (predictor) and  $y$  (response), you can perform a least-squares linear regression using `fitlm`:

```
>> linmodel = fitlm(x,y)
```

- You can visually inspect the fit using the `plot` method:

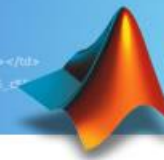
```
>> plot(linmodel)
```

- Evaluate the fitted model at chosen predictor values using the **predict** method:

```
>> predict(linmodel,xnew)
```

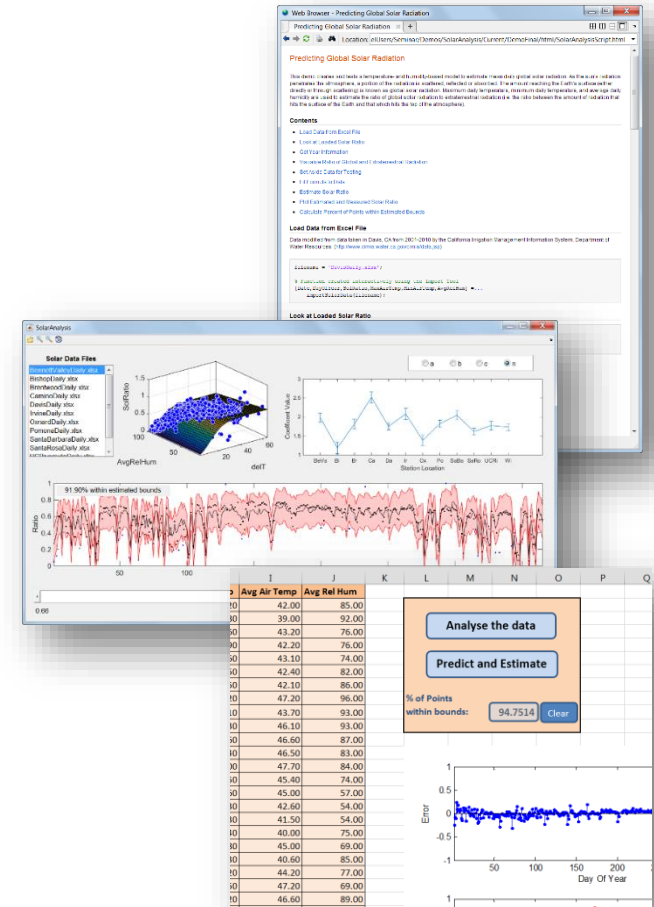
- This value is one of the properties of a **LinearModel** variable:

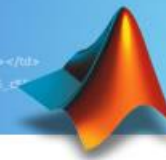
```
>> linmodel.Rsquared
```



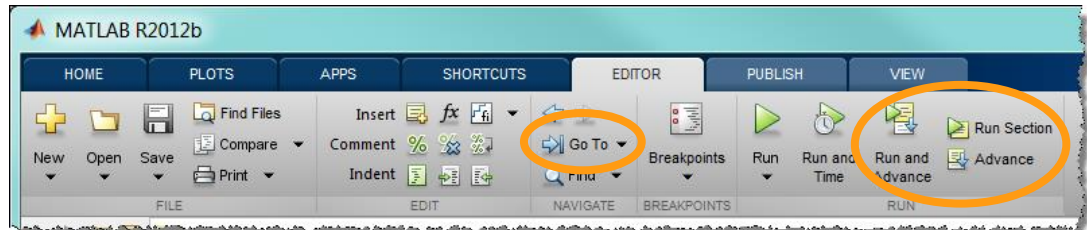
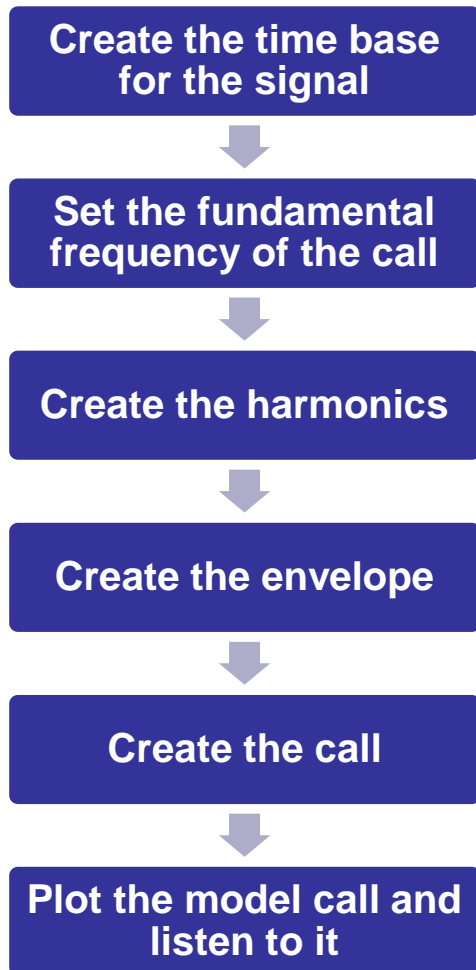
# Sharing Results from MATLAB

- Automatically generate reports
- Create and package applications
- Deploy to other environments





# Code Sections



%%

```

%% Set the fundamental frequency of the call.
f0 = 175;

%% Create the harmonics.
y0 = sin(2*pi*f0*t) + sin(2*pi*2*f0*t) + sin(2*pi*3*f0*t);

%% Create the envelope
% Set the additional parameters in the model.
A0 = 2; % Initial amplitude.
B = 1.5; % Amplitude decay rate.
fm = 0.65; % Frequency of the modulating envelope.
% Create the envelope
A = A0*exp(-B*t).*sin(2*pi*fm*t);

%% Create the call.
call = A.*y0;
    
```

# Publishing Code

```
%% Set the fundamental frequency of the call.
```

```
f0 = 175;
```

```
%% Create the harmonics.
```

```
y0 = sin(2*pi*f0*t) + sin(2*pi*2*f0*t) + sin(2*pi*3*f0*t);
```

```
%% Create the envelope
```

```
% Set the additional parameters in the model.
```

```
A0 = 2; % Initial amplitude.
```

```
B = 1.5; % Amplitude decay rate.
```

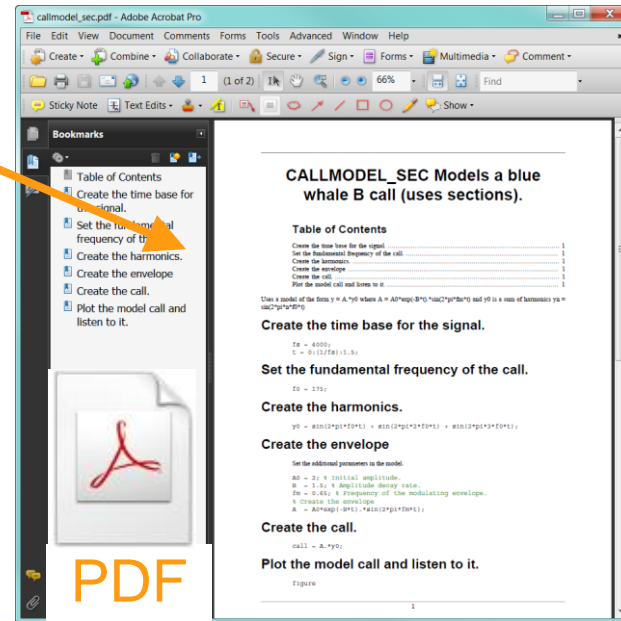
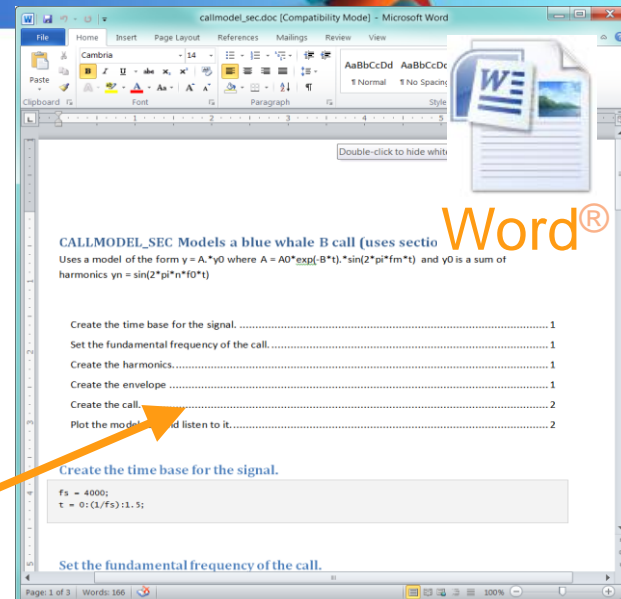
```
fm = 0.65; % Frequency of the modulating envelope.
```

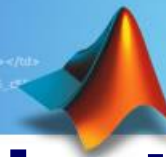
```
% Create the envelope
```

```
A = A0*exp(-B*t).*sin(2*pi*fm*t);
```

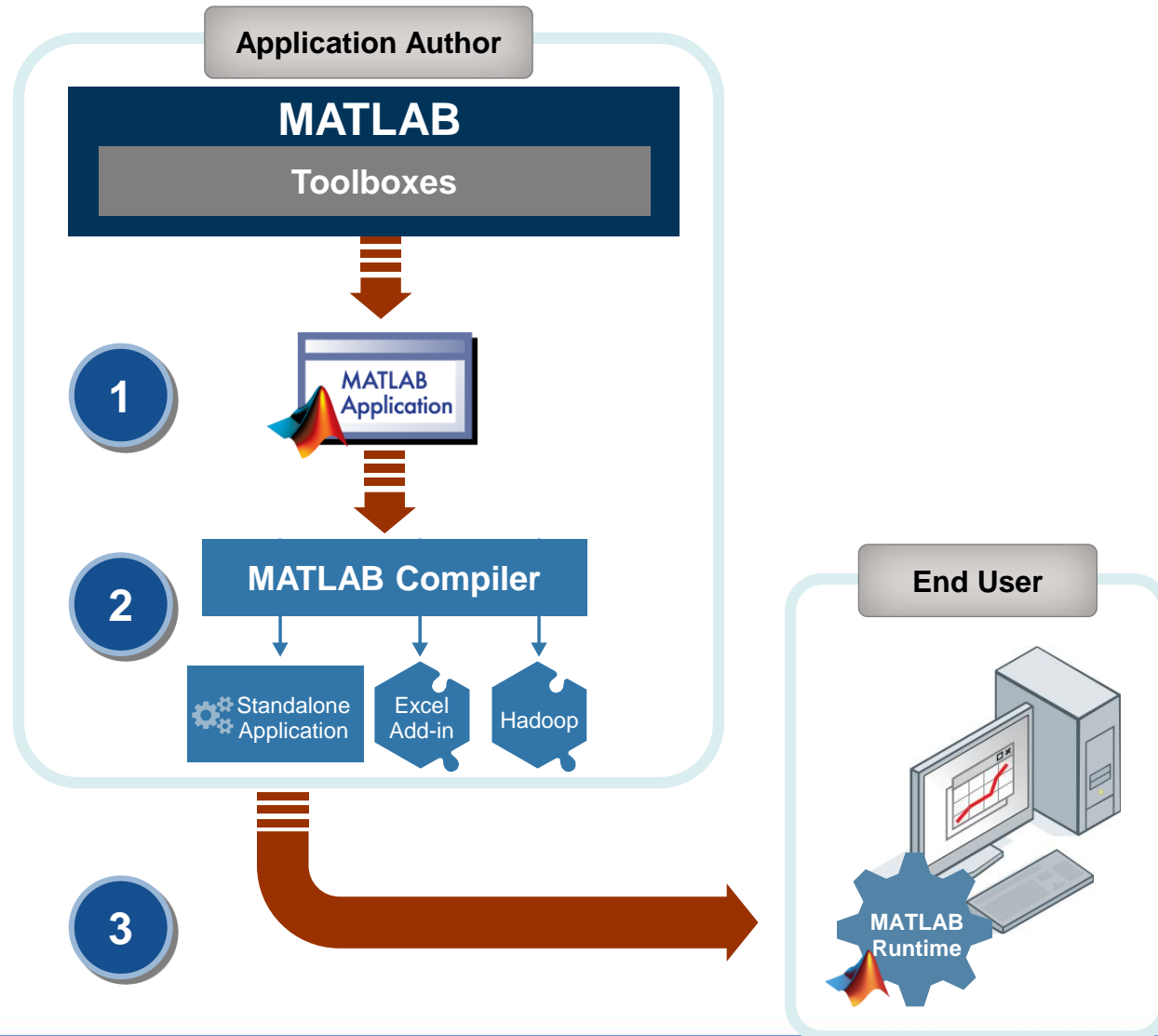
```
%% Create the call.
```

```
call = A.*y0;
```

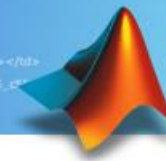




# Sharing Standalone Applications







# Thanks for your attention !