

# 視窗介面設計

## MATLAB進階程式語言與實作

盧家鋒 副教授

生物醫學影像暨放射科學系

[alvin4016@nycu.edu.tw](mailto:alvin4016@nycu.edu.tw)

# Teaching Materials

<http://cflu.lab.nycu.edu.tw>

**Contents → Teaching Materials → MATLAB ML (G)**

Please download **Week 9** Materials.

Compulsory Course for the Undergraduate Students  
Lecturer: Chia-Feng Lu ([alvin4016@ym.edu.tw](mailto:alvin4016@ym.edu.tw))

Matlab進階程式設計與專題實作 (碩博)  
授課教師：盧家鋒

Please set current directory to **MLmaterials\_L9**

Home Contents

## MATLAB Programming for Machine Learning (Graduate)

Compulsory Course for the Undergraduate Students  
Lecturer: Chia-Feng Lu ([alvin4016@ym.edu.tw](mailto:alvin4016@ym.edu.tw))

Matlab進階程式設計與專題實作 (碩博)  
授課教師：盧家鋒

- CV & Publications
- Members
- Research Interests
- Teaching Materials**
- Download Platforms
- Activities
- Relevant Links

- MRI (UG)
- MRM (UG)
- MRI Research (G)
- MATLAB programming (UG)
- MATLAB ML (G)**
- MATLAB GUI (G)
- Signal Processing (G)
- Computer Sci. (UG)
- Computer Arch. (UG)
- fMRI Analysis (G)
- rs-fMRI Analysis (G)
- fNIRS Basics (G)
- fNIRS Workshop (G)
- Human Dissection (UG)
- Neuroanatomy (UG)
- Image Processing (R)



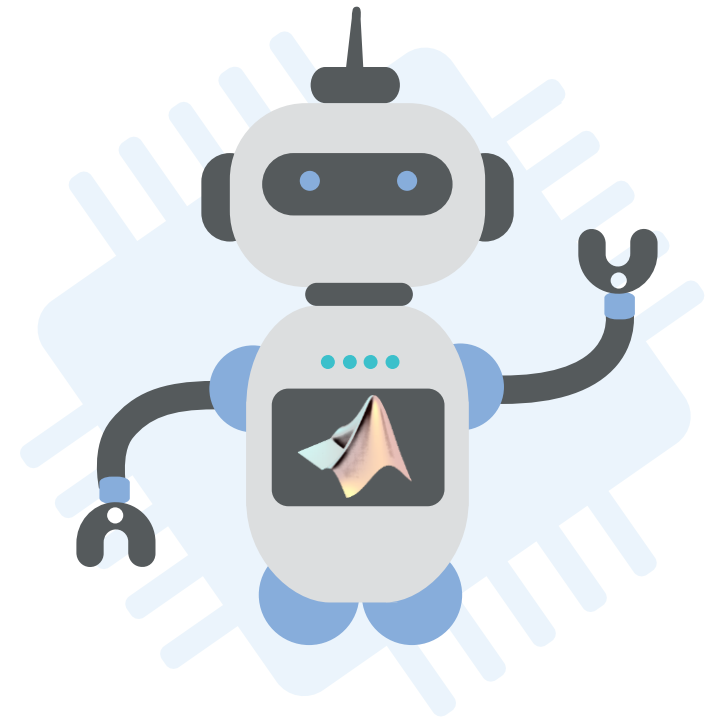
# Contents in this Week

## 01 Introduction of App Designer

Build your very first App!

## 02 Advanced Usages of App Designer

Property and package of App





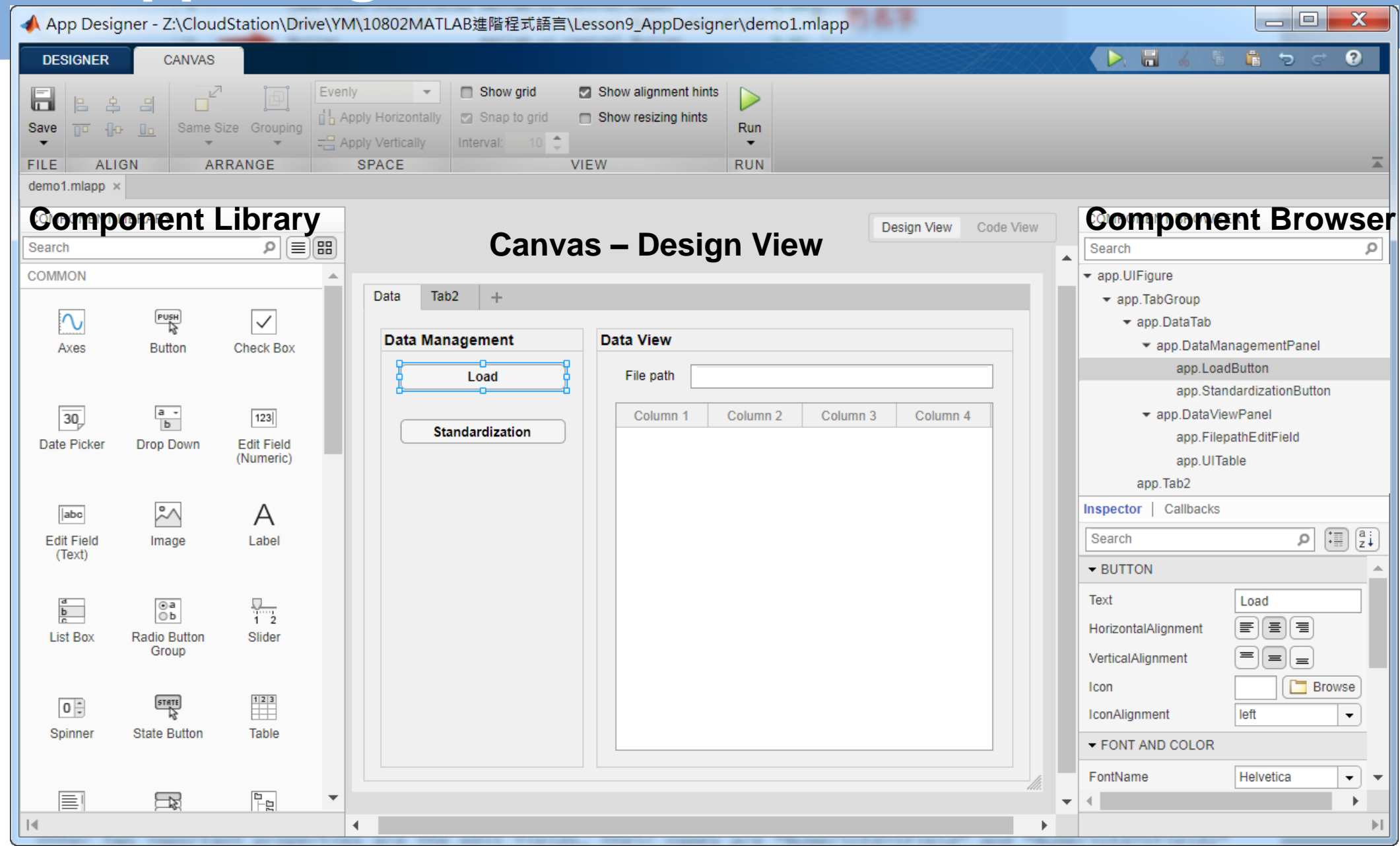
# Introduction of App Designer

Build your very first App!

# App Designer

- An alternative for GUIDE in R2016a.
- Create professional apps without having to be a professional software developer.
- Design a User Interface
  - Drag and drop visual components to lay out the design of your graphical user interface (GUI).
- Define App Behavior
  - Use the integrated editor to quickly program its behavior.
- Package Your App
















# >> appdesigner








# Component Library

COMMON

 Axes	 Button	 Check Box
 Date Picker	 Drop Down	 Edit Field (Numeric)
 Edit Field (Text)	 Image	 Label
 List Box	 Radio Button Group	 Slider
 Spinner	 State Button	 Table

 Text Area	 Toggle Button Group	 Tree
---	--	---

CONTAINERS















 Grid Layout	 Panel	 Tab Group
---	--	--

FIGURE TOOLS

 Menu Bar
--

INSTRUMENTATION

 90 Degree Gauge	 Discrete Knob	 Gauge
 Knob	 Lamp	 Linear Gauge
 Rocker Switch	 Semicircular Gauge	 Switch
 Toggle Switch		

# >> appdesigner

The screenshot displays the MATLAB App Designer interface for a file named `demo1.mlapp`. The interface is divided into several panels:

- DESIGNER / EDITOR**: The top toolbar contains icons for Save, Callback, Function, Property, App Input Arguments, Go To, Find, Comment, Indent, Enable app coding alerts, Show Tips, and Run.
- Component Library**: Located on the left, it includes tabs for Callbacks, Functions, and Properties. It contains a search bar and instructions on adding callback functions.
- Canvas – Code View**: The central area shows the MATLAB code for the app. It includes a class definition for `demo1` inheriting from `matlab.apps.AppBase`, with properties for various UI components and a `createComponents` function for initialization.
- Component Browser**: On the right, it lists available components such as `app.UIFigure`, `app.TabGroup`, `app.DataTab`, `app.DataManagementPanel`, `app.LoadButton`, `app.StandardizationButton`, `app.DataViewPanel`, `app.FilepathEditField`, `app.UITable`, and `app.Tab2`.
- Inspector**: Below the Component Browser, it shows the properties of the selected component (a `BUTTON`), including Text, HorizontalAlignment, VerticalAlignment, Icon, IconAlignment, FontName, and FontSize.
- APP LAYOUT**: At the bottom left, it provides a visual preview of the app's user interface, showing a 'Data' tab with a 'Data Management' panel and a 'Data View' panel.

```
classdef demo1 < matlab.apps.AppBase
% Properties that correspond to app components
properties (Access = public)
    UIFigure          matlab.ui.Figure
    TabGroup           matlab.ui.container.TabGroup
    DataTab            matlab.ui.container.Tab
    DataManagementPanel matlab.ui.container.Panel
    LoadButton        matlab.ui.control.Button
    StandardizationButton matlab.ui.control.Button
    DataViewPanel      matlab.ui.container.Panel
    FilepathEditFieldLabel matlab.ui.control.Label
    FilepathEditField  matlab.ui.control.EditField
    UITable            matlab.ui.control.Table
    Tab2               matlab.ui.container.Tab
end

% Component initialization
methods (Access = private)

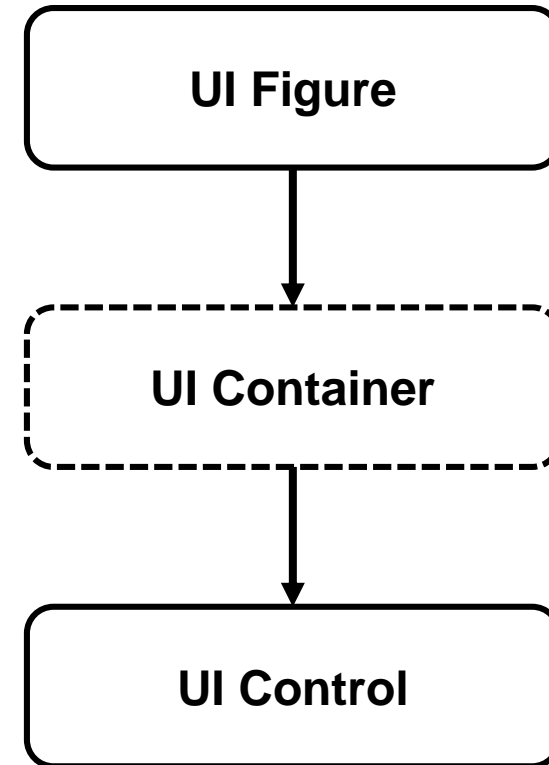
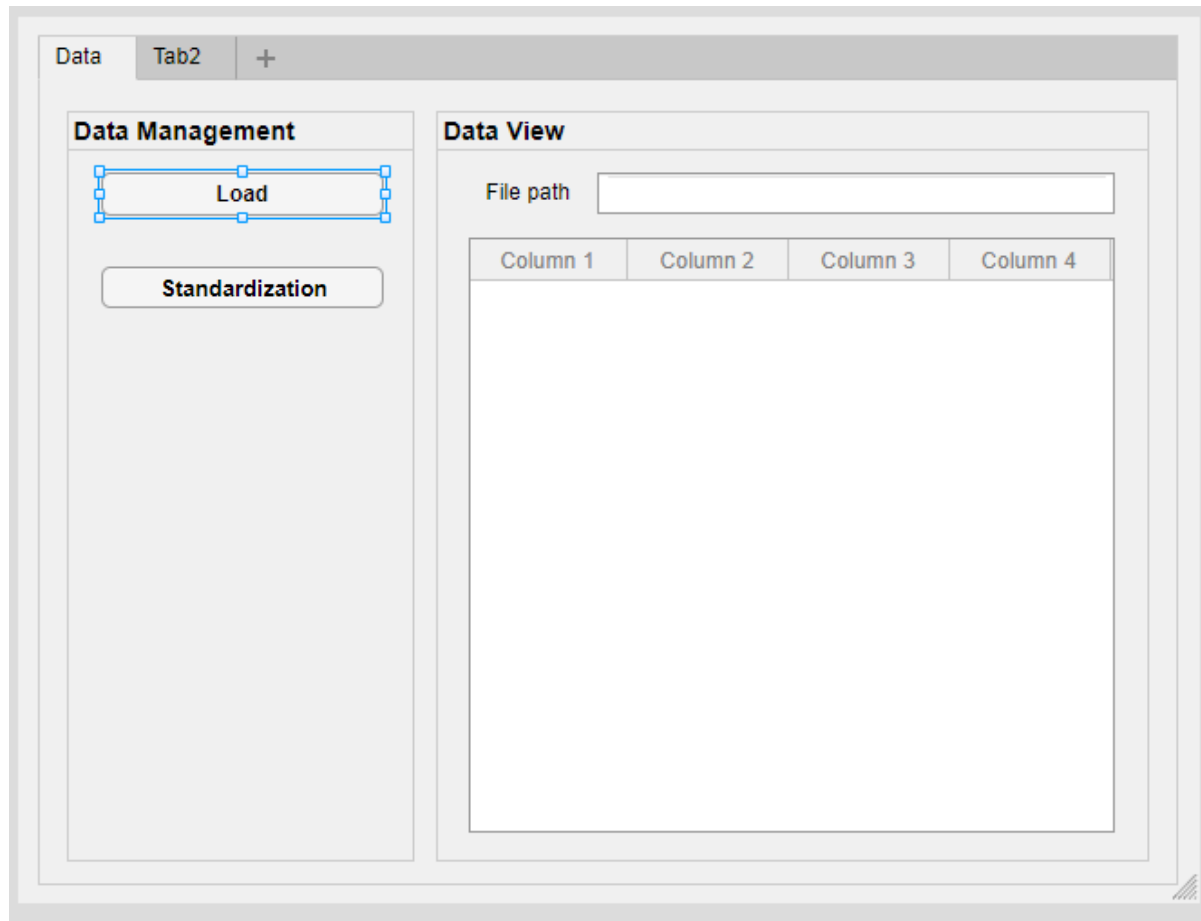
% Create UIFigure and components
function createComponents(app)

% Create UIFigure and hide until all components are created
app.UIFigure = uifigure('Visible', 'off');
app.UIFigure.Position = [100 100 640 480];
app.UIFigure.Name = 'UI Figure';

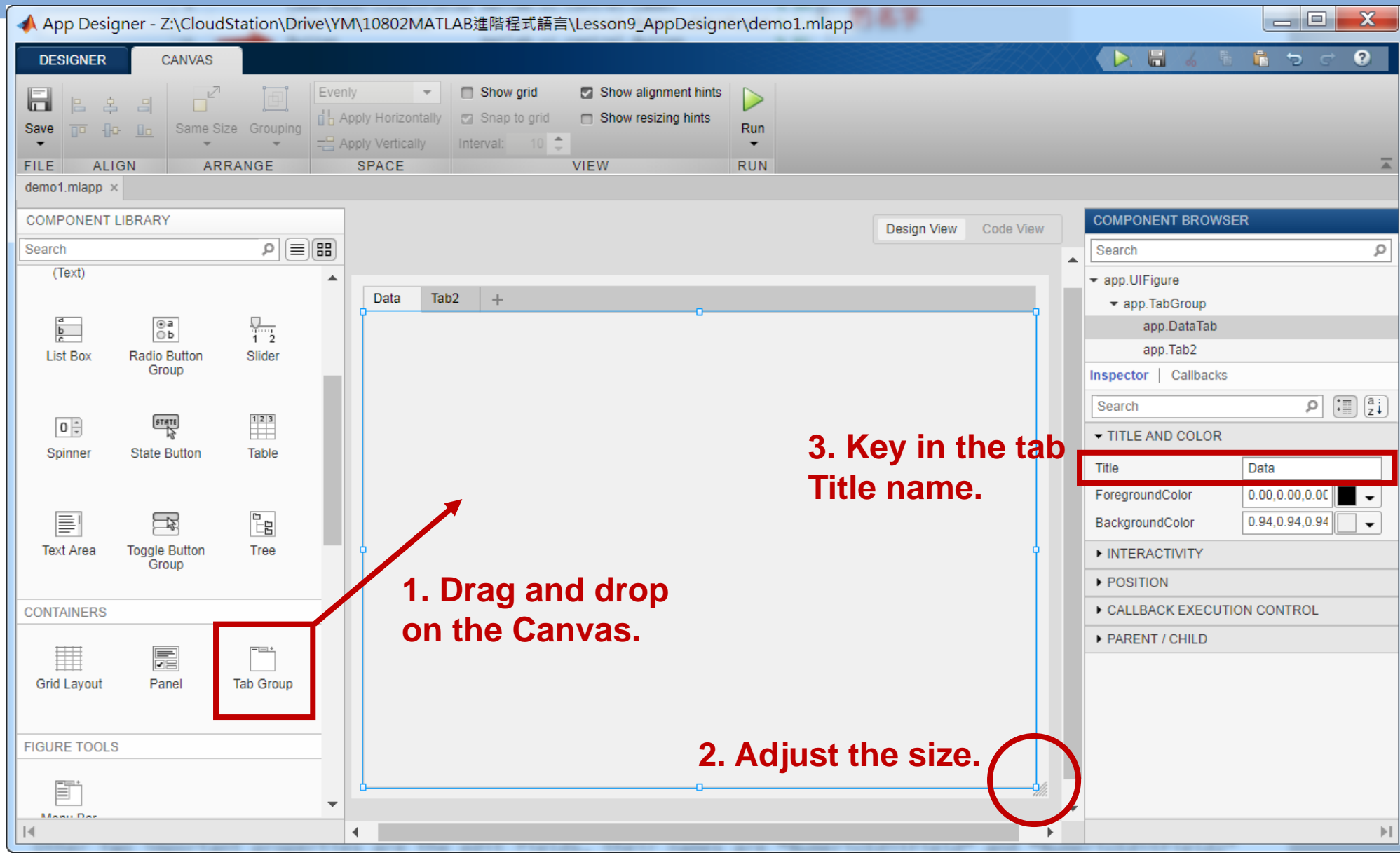
% Create TabGroup
```



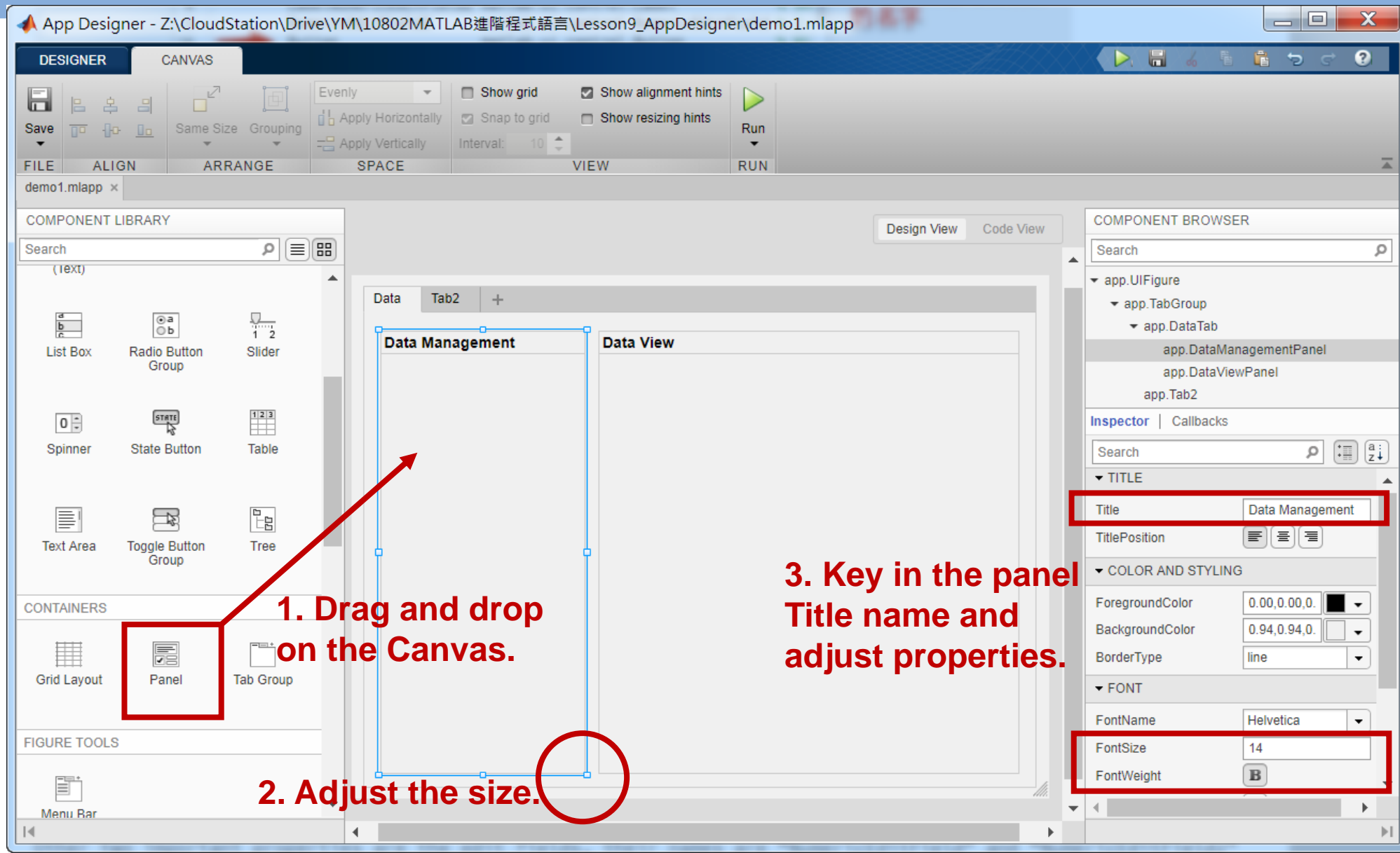
# App Components



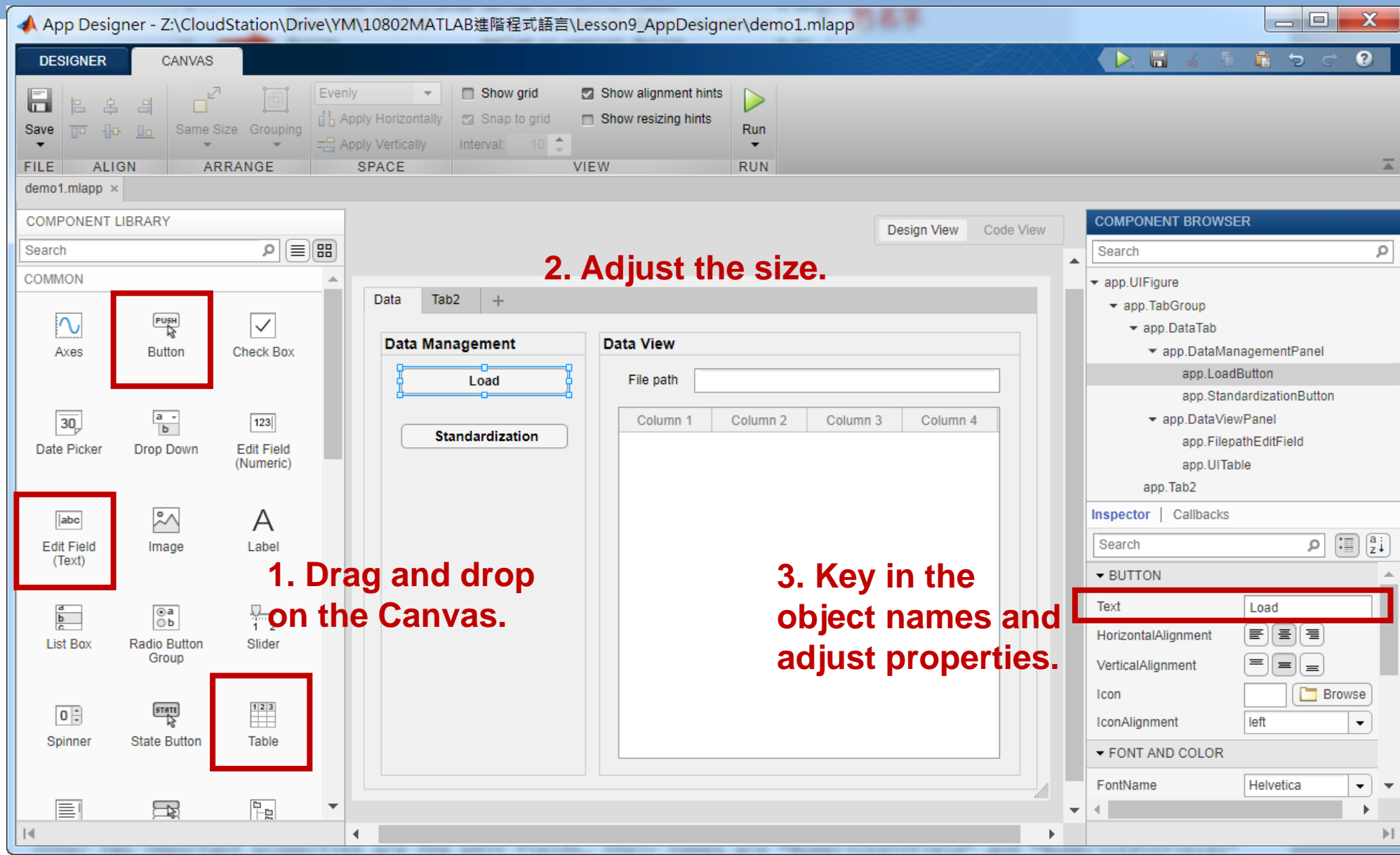
# Create Your First App – UI Container



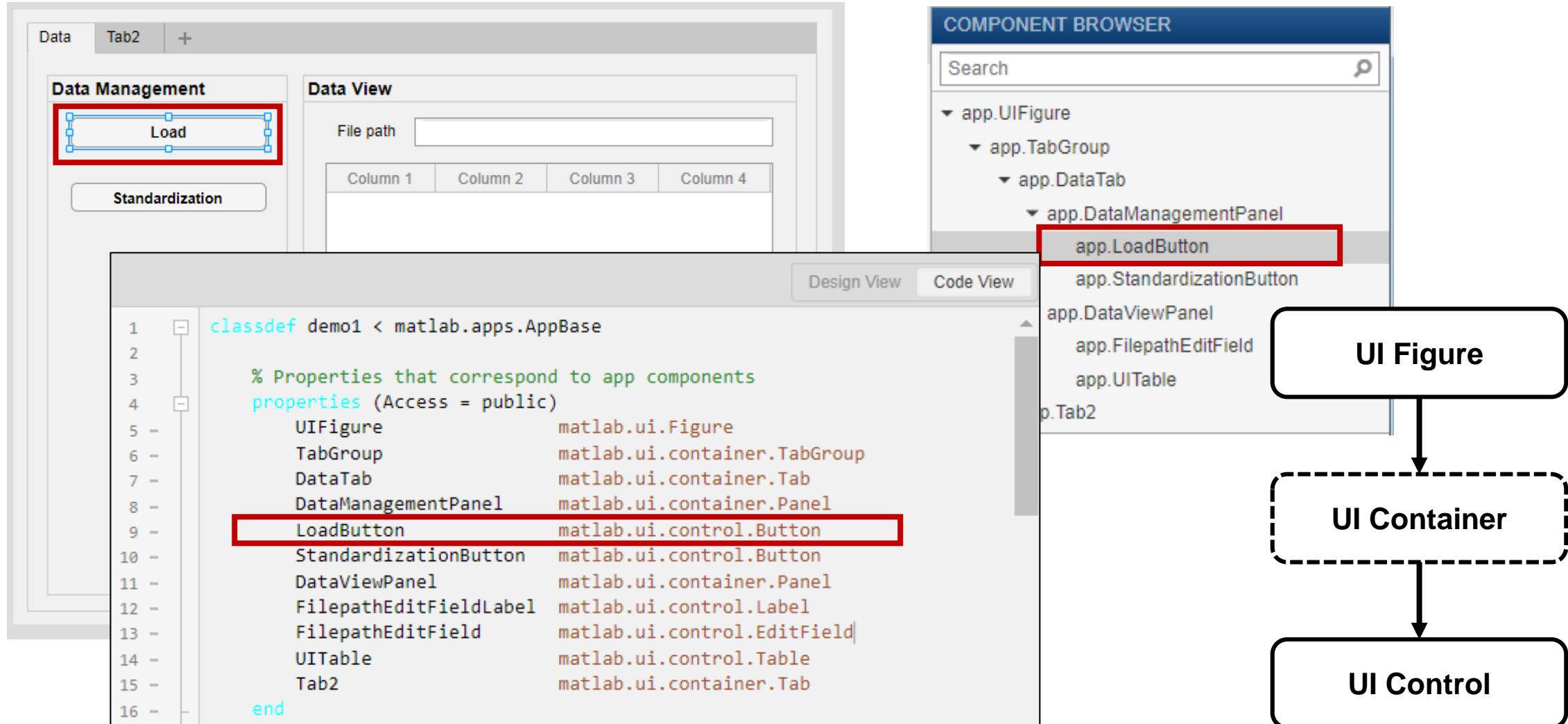
# Create Your First App – UI Container



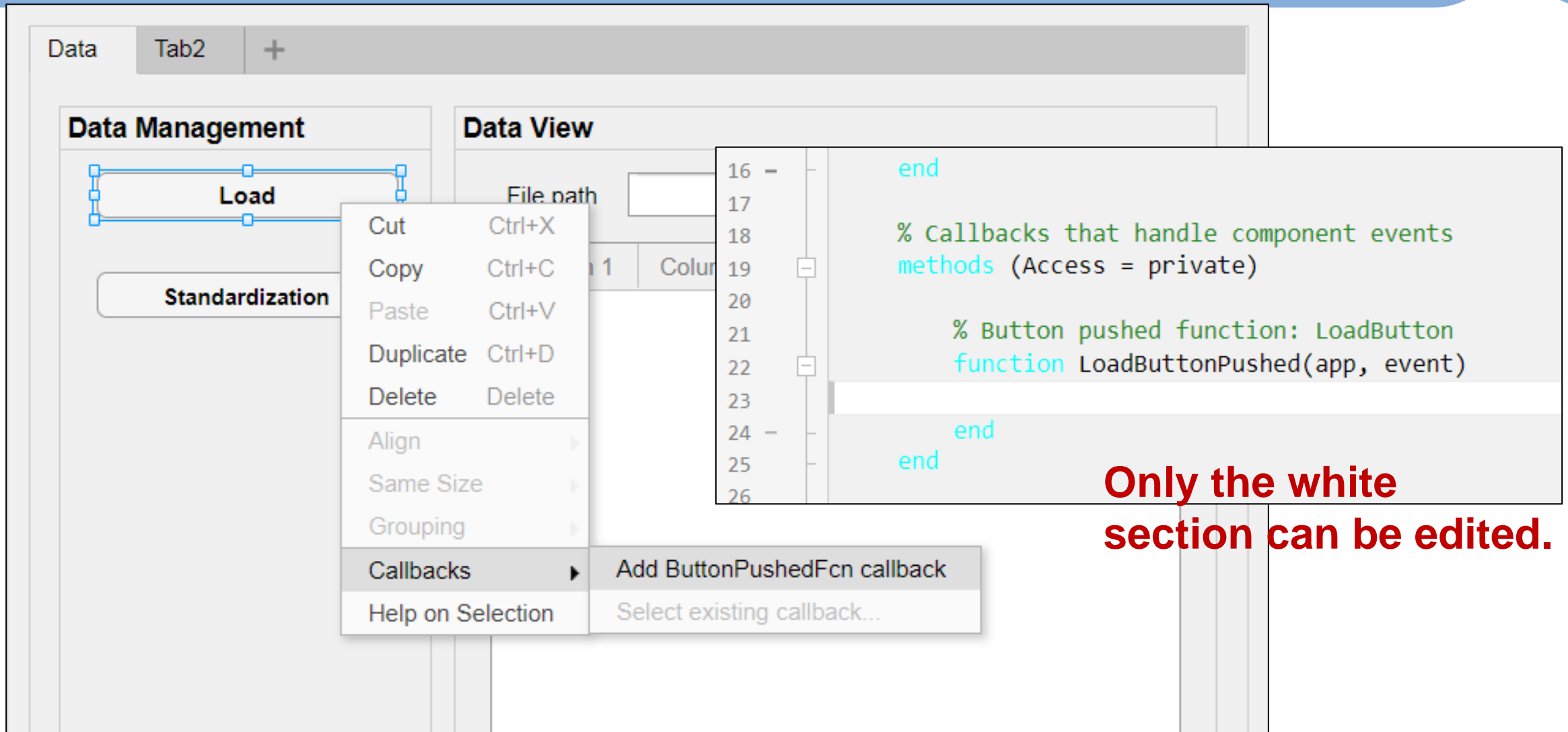
# Create Your First App – UI Control



# App Components



# Callback of UI controls – Load



The screenshot displays the MATLAB App Designer interface. On the left, the 'Data Management' section contains a 'Load' button. A context menu is open over this button, with the 'Callbacks' option selected, revealing a sub-menu with 'Add ButtonPushedFcn callback' and 'Select existing callback...'. On the right, the 'Data View' section shows the corresponding MATLAB code. The code is as follows:

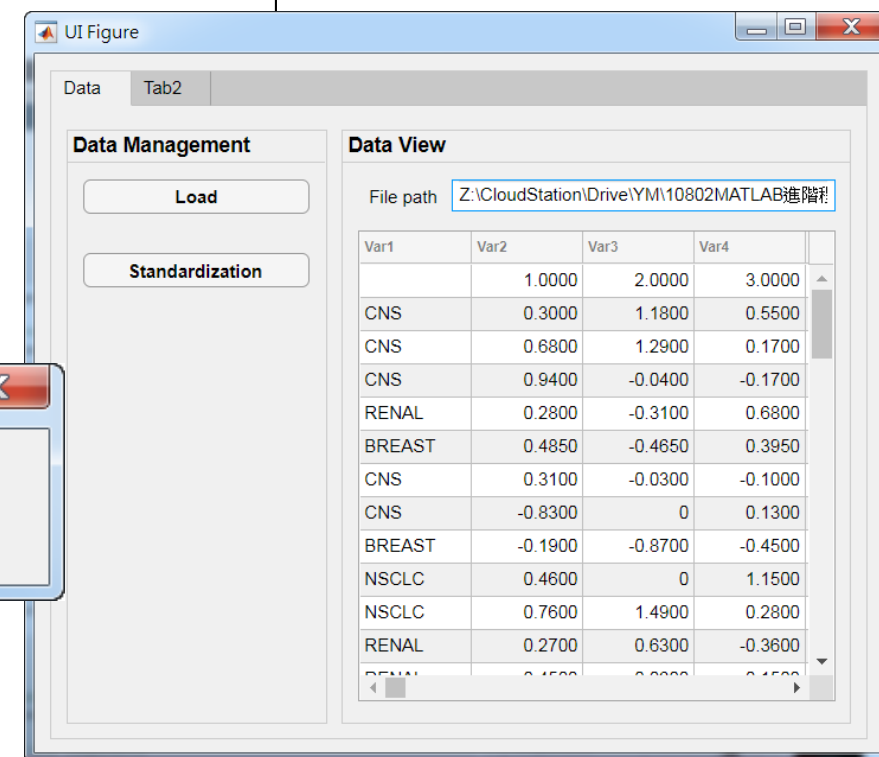
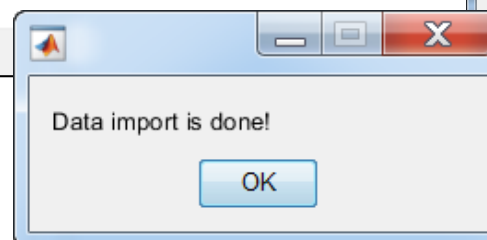
```
16 - end
17
18 % Callbacks that handle component events
19 methods (Access = private)
20
21 % Button pushed function: LoadButton
22 function LoadButtonPushed(app, event)
23
24 end
25
26 end
```

The code is divided into two sections: a grey section for comments and a white section for the function definition. The white section, containing the function definition, is highlighted with a red box and the text 'Only the white section can be edited.'



# Load data and display in UITable

```
18 % Callbacks that handle component events
19 methods (Access = private)
20
21 % Button pushed function: LoadButton
22 function LoadButtonPushed(app, event)
23     [filename,filepath] = uigetfile('*.csv','Please select a file to import. ');
24     if filename
25         app.FilepathEditField.Value = [filepath, filename];
26         data = readtable([filepath,filename]);
27         app.UITable.Data = data;
28         app.UITable.ColumnName = data.Properties.VariableNames;
29         msgbox('Data import is done!')
30
31         assignin('base','app',app) % assign 'app' to workspace
32     end
33 end
```



Load MLmaterials\_L9\NCI60data.csv

# Callback of UI controls – Standardization

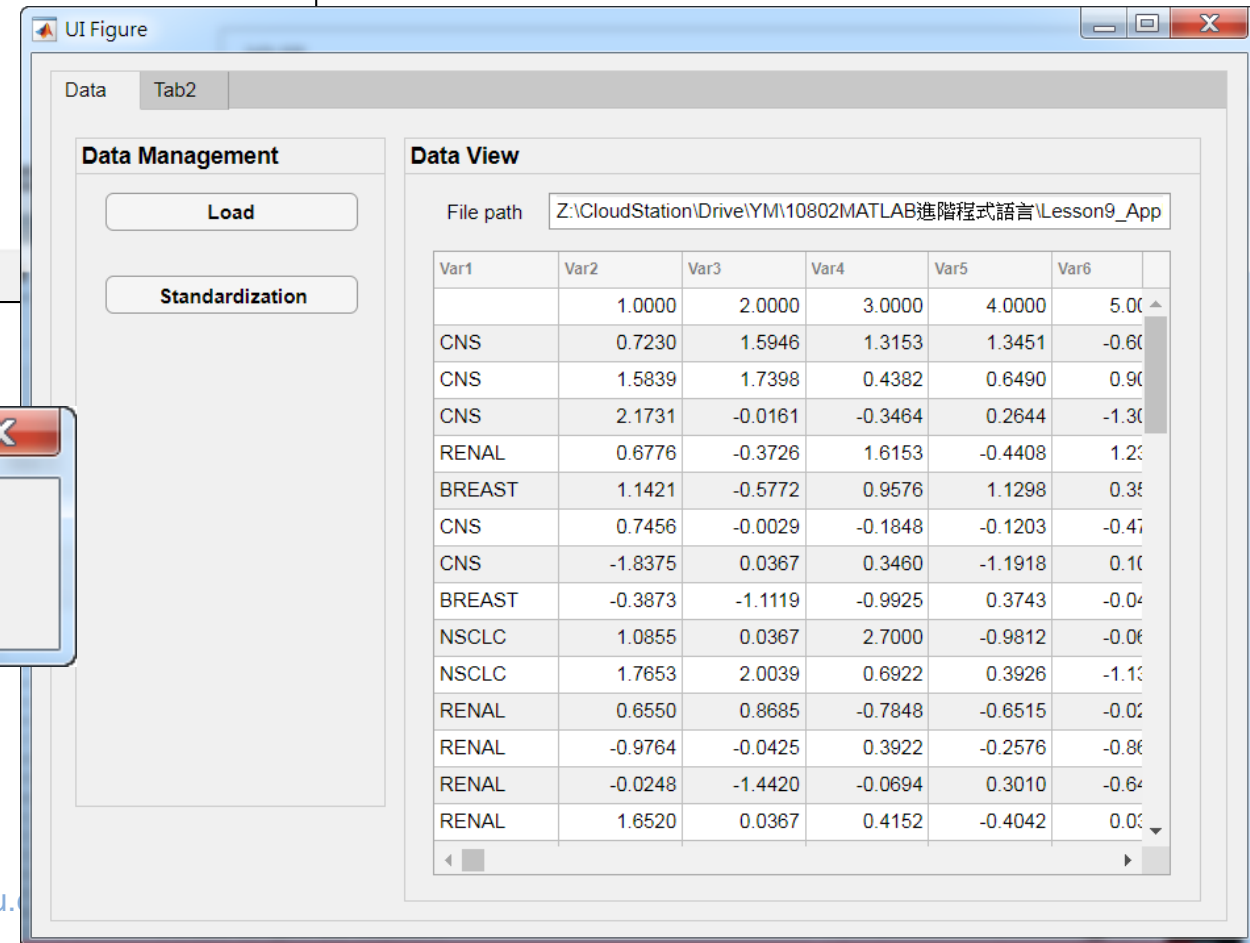
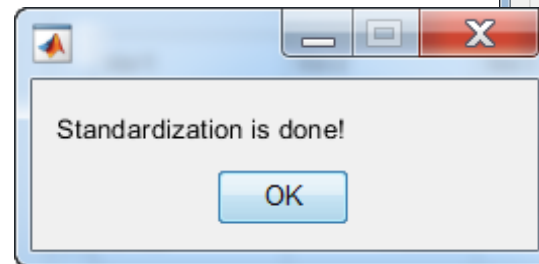
The screenshot displays the MATLAB App Designer interface. On the left, the 'Data Management' pane contains a 'Load' button and a 'Standardization' button. A context menu is open over the 'Standardization' button, showing options: Cut (Ctrl+X), Copy (Ctrl+C), Paste (Ctrl+V), Duplicate (Ctrl+D), Delete (Delete), Align, Same Size, Grouping, Callbacks (selected), and Help on Selection. The 'Callbacks' sub-menu is open, showing 'Add ButtonPushedFcn callback' and 'Select existing callback...'. On the right, the 'Data View' pane shows a table with columns 'Column 1', 'Column 2', 'Column 3', and 'Column 4'. Below the table, a code editor window is open, showing the following code:

```
% Button pushed function: StandardizationButton
function StandardizationButtonPushed(app, event)

end
```

# Standardize the data

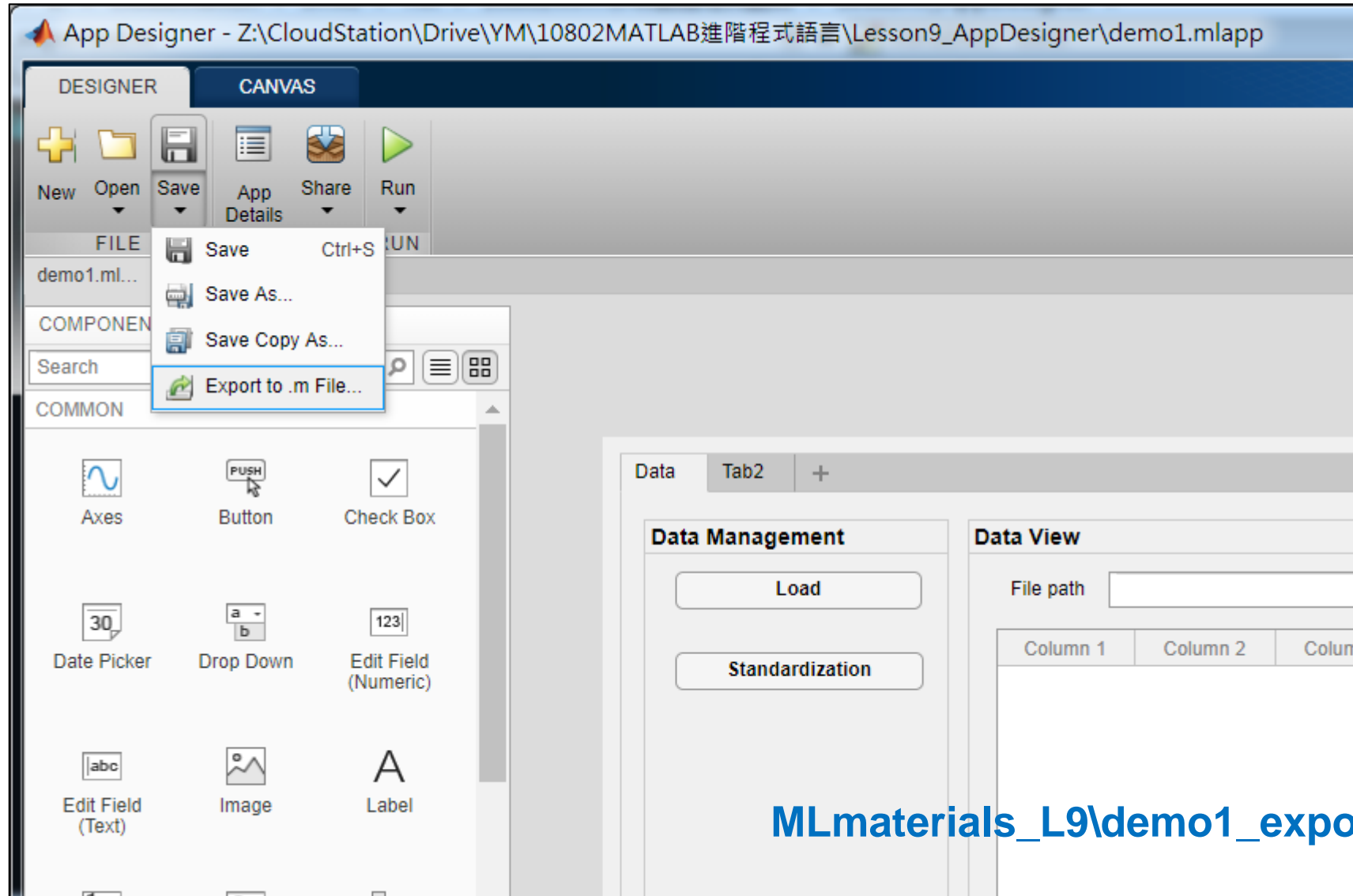
```
35 % Button pushed function: StandardizationButton
36 function StandardizationButtonPushed(app, event)
37     data = app.UITable.Data;
38     numdata = table2array(data(2:end,2:end));
39     Z = zscore(numdata);
40     data(2:end,2:end) = array2table(Z);
41     app.UITable.Data = data;
42
43     msgbox('Standardization is done!')
44 end
```




MLmaterials\_L9\demo1.mlapp

<http://cflu.lab.nycu.edu.tw/>

# Generate M-code (\*.m)



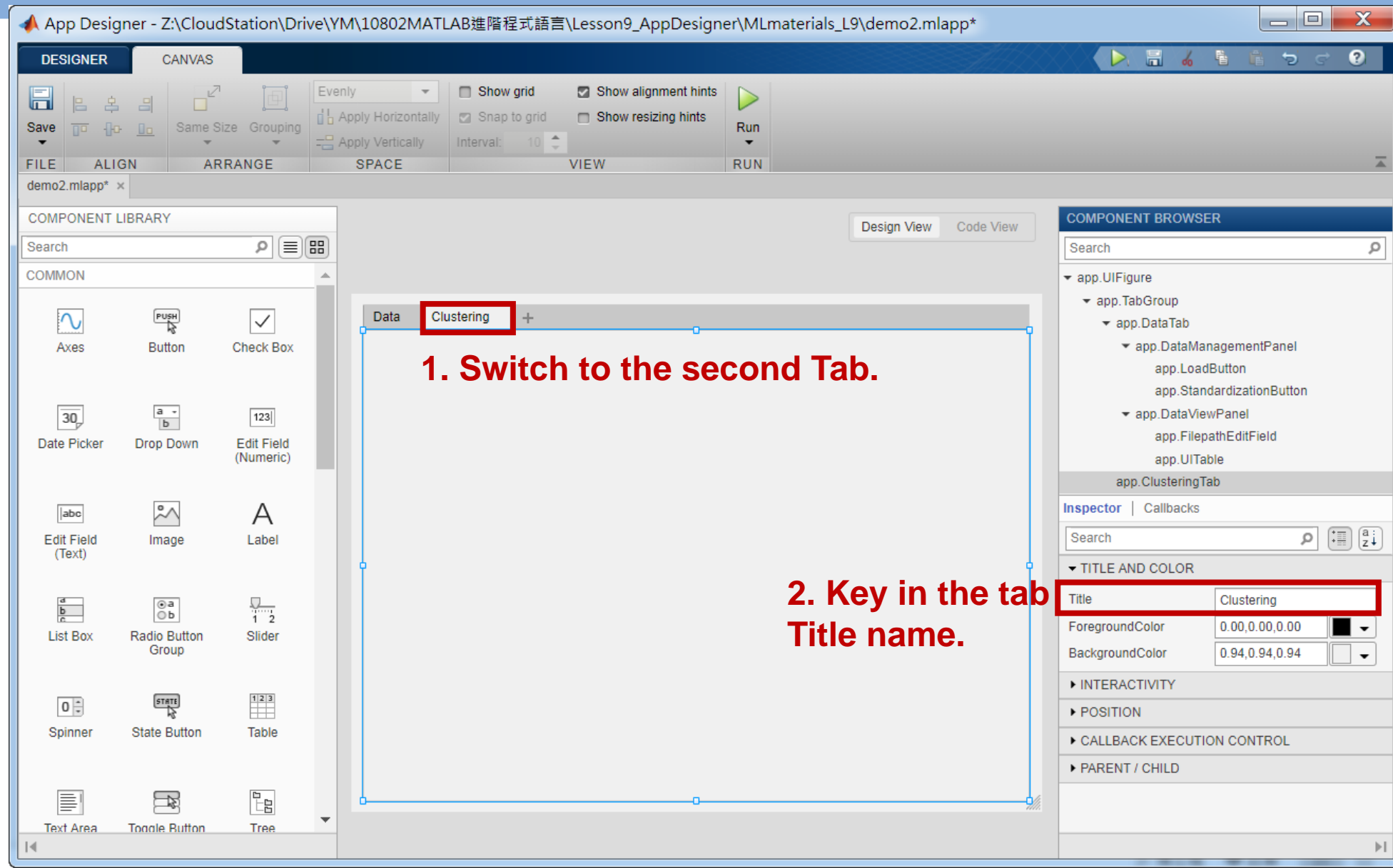
MLmaterials\_L9\demo1\_exported.m



# Advanced Usages of App Designer

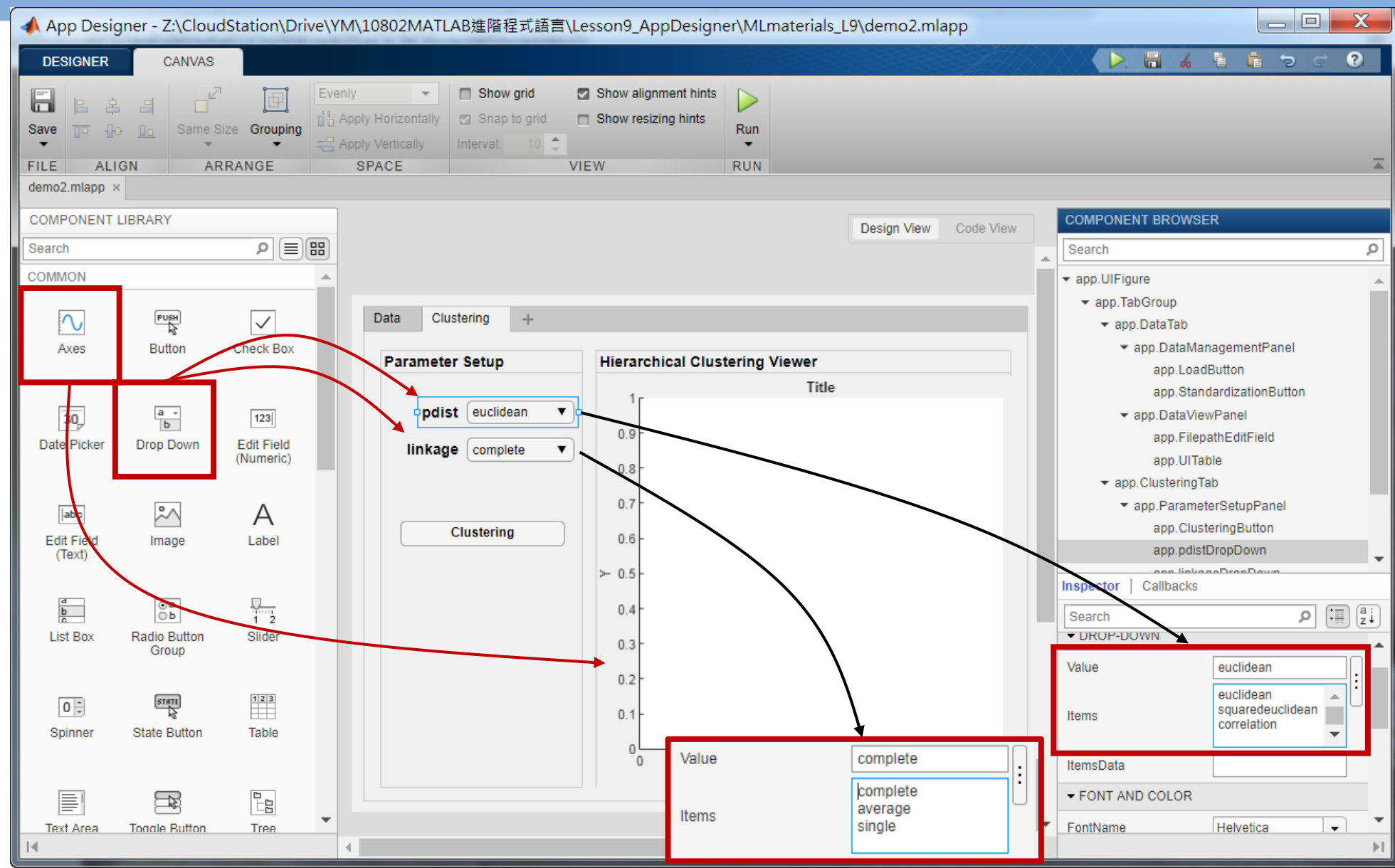
Property and package of App

# UI Tab Usage





# Layout of the second Tab



# Setup a check point for standardization

1. **Load**

2. **Clustering**

**Error Dialog**

! Please standardize data first!

OK

Var1	Var2	Var3
	1.0000	2.0
CNS	0.3000	1.1
CNS	0.6800	1.2
CNS	0.9400	-0.0
RENAL	0.2800	-0.3
BREAST	0.4850	-0.4
CNS	0.3100	-0.0
CNS	-0.8300	
BREAST	-0.1900	-0.8
NSCLC	0.4600	
NSCLC	0.7600	1.4
RENAL	0.2700	0.6
RENAL	-0.4500	-0.0

# Add in a Property under app

The screenshot displays the MATLAB App Designer interface. The top toolbar includes icons for Save, Callback, Function, Property, App Input Arguments, Go To, Find, Comment, Indent, and a checkbox for 'Enable app coding ale'. The 'Property' menu is open, showing two options: 'Private Property' (highlighted with a red box) and 'Public Property'. The 'CODE BROWSER' panel is visible on the left, showing the 'Properties' tab. A tooltip explains how to add a property: 'Add a property to create a variable to store and share data between callbacks and functions. Specify the property name with the prefix app. to access the property value: app.Property = someData;'. The background code editor shows a list of MATLAB components: linkageDropDownLabel, linkageDropDown, dendrogramEditFieldLabel, dendrogramEditField, pdistDropDown, and HierarchicalClusteringViewerPa.

**DESIGNER** **EDITOR**

Save Callback Function **Property** App Input Arguments Go To Find Comment Indent ☒ Enable app coding ale

FILE INS VIEW

demo2.mlapp x

▼ **CODE BROWSER**

Callbacks | Functions | **Properties**

Search

+ Add a property to create a variable to store and share data between callbacks and functions. Specify the property name with the prefix app. to access the property value:

```
app.Property = someData;
```

19 - linkageDropDownLabel matl  
20 - linkageDropDown matl  
21 - dendrogramEditFieldLabel matl  
22 - dendrogramEditField matl  
23 - pdistDropDown matl  
24 - HierarchicalClusteringViewerPa

# Setup a check point for standardization

```
28
29 properties (Access = private)
30     check_zscore % check value of standardization
31 end
32
37 % Button pushed function: LoadButton
38 function LoadButtonPushed(app, event)
39     [filename,filepath] = uigetfile('*.','Please select a file to import. ');
40     if filename
41         app.FilepathEditField.Value = [filepath, filename];
42         data = readtable([filepath,filename]);
43         app.UITable.Data = data;
44         app.UITable.ColumnName = data.Properties.VariableNames;
45         app.check_zscore = 0;
46
52 % Button pushed function: StandardizationButton
53 function StandardizationButtonPushed(app, event)
54     data = app.UITable.Data;
55     numdata = table2array(data(2:end,2:end));
56     Z = zscore(numdata);
57     data(2:end,2:end) = array2table(Z);
58     app.UITable.Data = data;
59
60     app.check_zscore = 1;
61
62     msgbox('Standardization is done!')
63 end
```

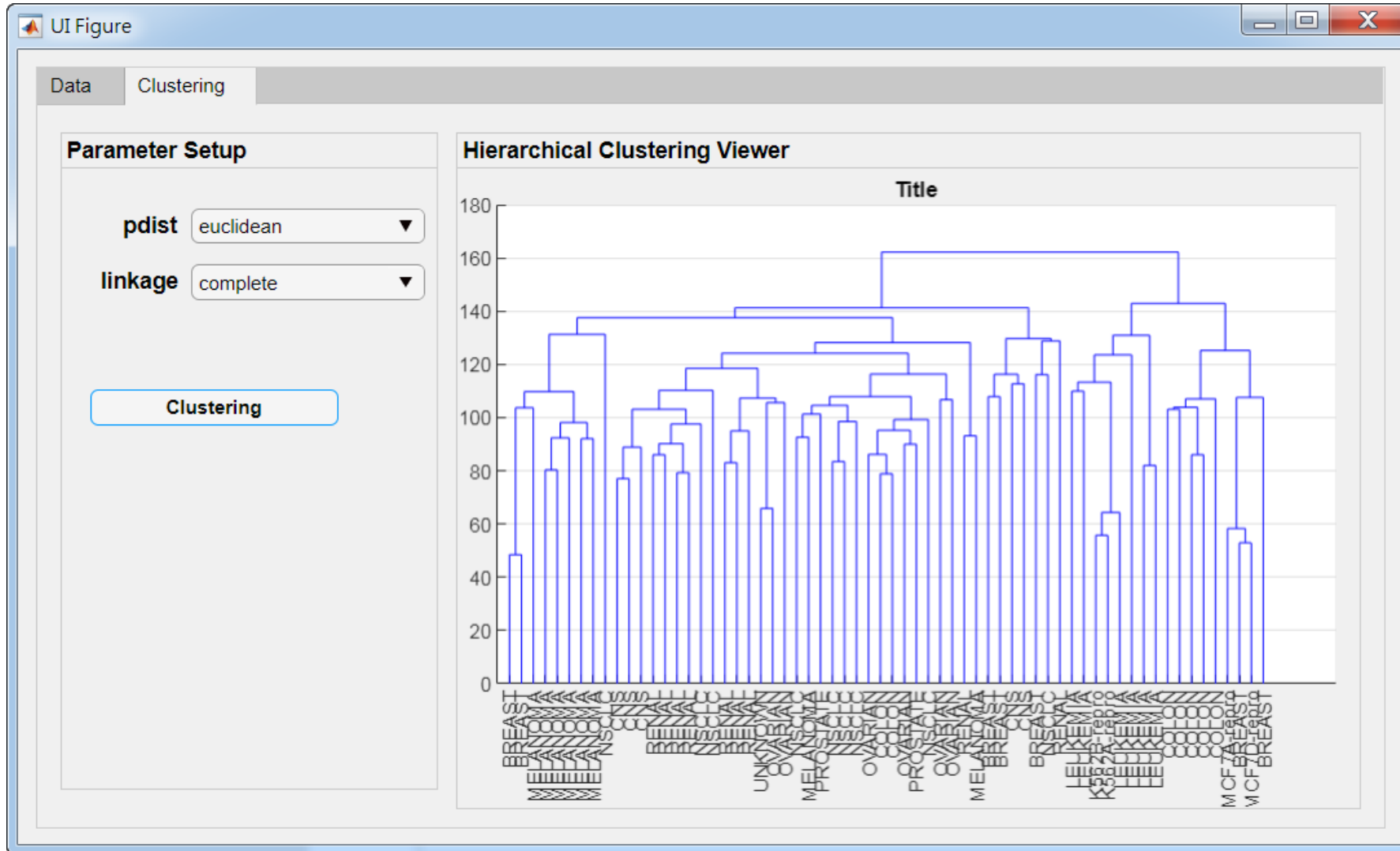
# Setup a check point for standardization

The screenshot displays a MATLAB App Designer interface for a clustering application. The 'Parameter Setup' panel on the left shows 'pdist' set to 'euclidean' and 'linkage' set to 'complete'. A 'Clustering' button is visible. A context menu is open over the button, showing options like 'Cut', 'Copy', 'Paste', 'Duplicate', 'Delete', 'Align', 'Same Size', 'Grouping', 'Callbacks', and 'Help on Selection'. The 'Callbacks' option is selected, showing a sub-menu with 'Add ButtonPushedFcn callback' and 'Select existing callback...'. The 'Hierarchical Clustering Viewer' panel on the right shows a dendrogram with a y-axis from 0.7 to 1.0 and an x-axis labeled 'X'. A code editor window is overlaid on the viewer, showing the following MATLAB function:

```
function ClusteringButtonPushed(app, event)
    if app.check_zscore == 0
        errordlg('Please standardize data first!')
        return
    end
end
```

An 'Error Dialog' window is also shown, with a red exclamation mark icon and the message 'Please standardize data first!'. The dialog has an 'OK' button.

# Perform Hierarchical Clustering

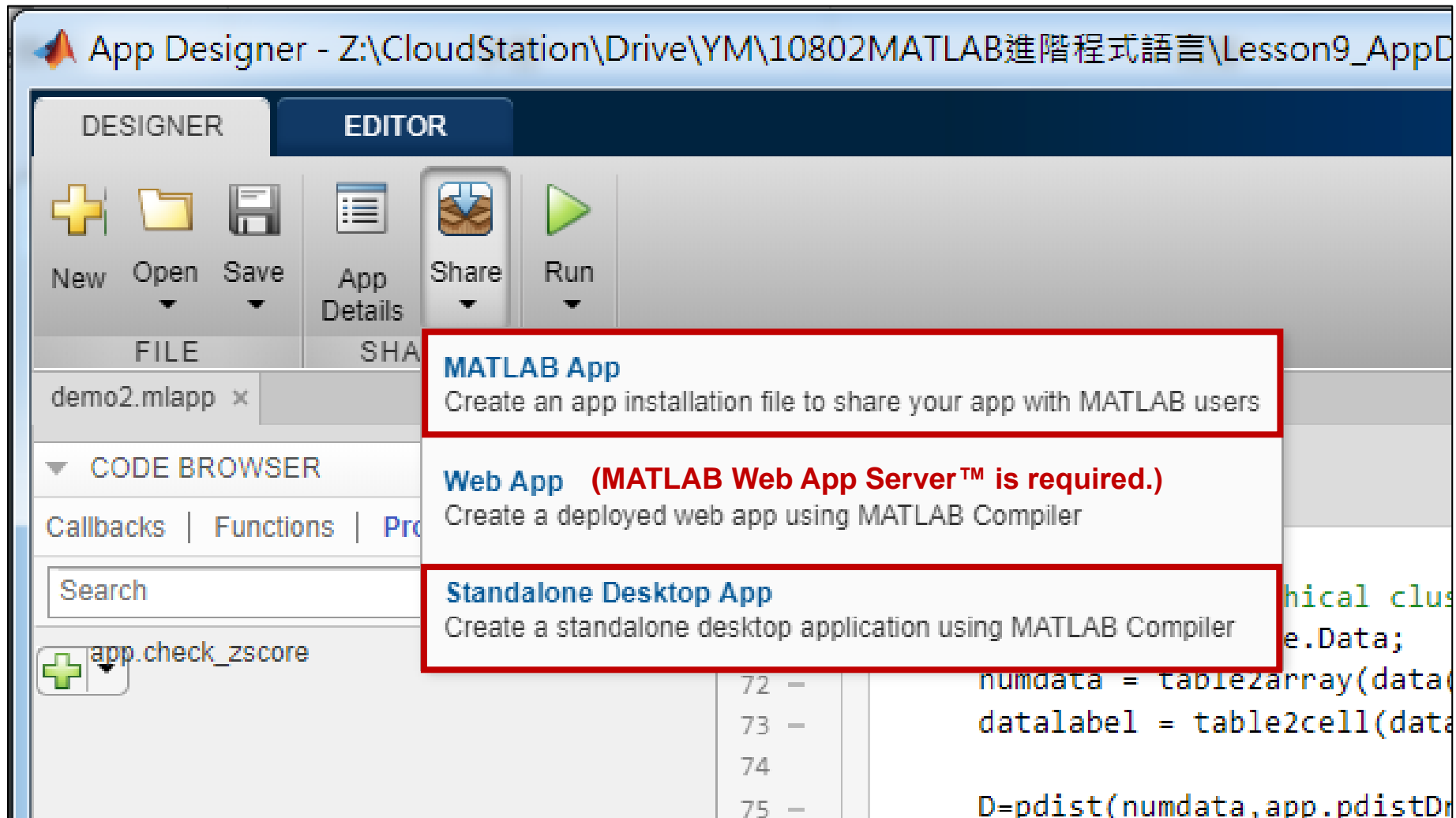




# Perform Hierarchical Clustering

```
70 % perform hierarchical clustering
71 - data = app.UITable.Data;
72 - numdata = table2array(data(2:end,2:end));
73 - datalabel = table2cell(data(2:end,1));
74
75 - D=pdist(numdata,app.pdistDropDown.Value);
76 - Ctree=linkage(D,app.linkageDropDown.Value);
77
78 - [H,T,outperm]=dendrogram(Ctree,0,'Labels',datalabel,'Orientation','top');
79
80 - plot(app.UIAxes,0)
81 - hold(app.UIAxes) % hold axes
82 - for i=1:length(H)
83 -     plot(app.UIAxes,H(i).XData,H(i).YData,'color','b')
84 - end
85 - hold(app.UIAxes) % release axes
86 - close(gcf) % close the figure generated by dendrogram
87
88 - app.UIAxes.XTick = 1:length(datalabel);
89 - app.UIAxes.XTickLabel = datalabel(outperm);
90 - app.UIAxes.XTickLabelRotation = 90;
91 - app.UIAxes.XLabel.String = '';
92 - app.UIAxes.YLabel.String = '';
93 - app.UIAxes.YGrid = 'on';
```

MLmaterials\_L9\demo2.mlapp





THE END

**Contact:**

盧家鋒 alvin4016@nycu.edu.tw