GUI to Interactively Explore Data in a Table

On this page…

About the tablestat Example
View and Run the tablestat GUI
Designing the GUI
Extending Tablestat

About the tablestat Example

This example shows how to program callbacks for interactive data exploration, including:

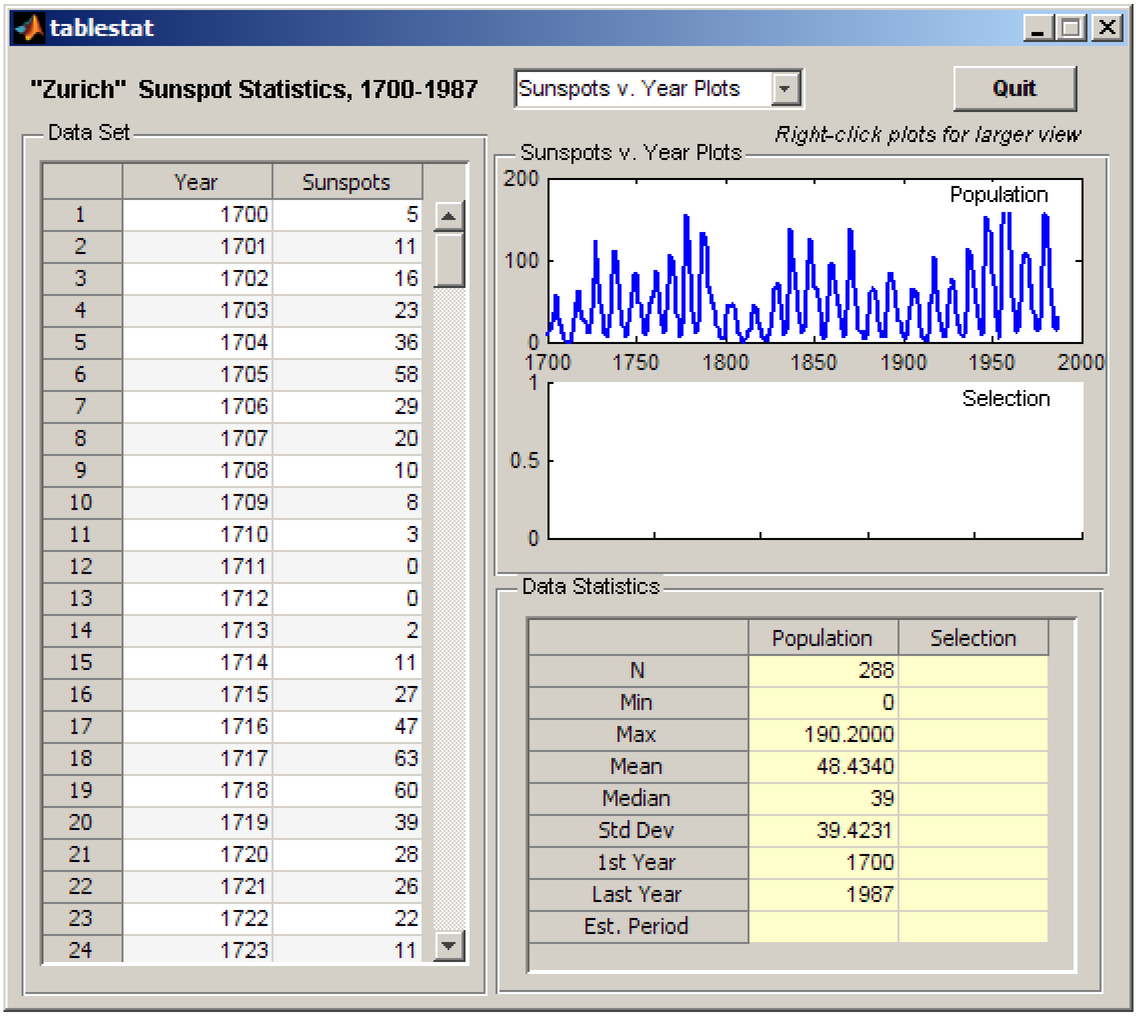
- An Opening Function to initialize a table and a plot.
- A uitable's Cell Selection Callback to do plot selected data in real time as the user selects data observations.
- A Pop-up menu's callback to generate line graphs that display different views of data.
- A context menu attached to an axes.

Use the GUI to plot different kinds of graphs into different axes for an entire data set or selections of it, and to see how Fourier transforms can identify periodicity in time series data. The GUI contains:

- A table of sunspot observations having two columns of data (dates and observations).
- A second table, statistically summarizing the data and a user-selected subset of it.
- Two axes that plot time series or Fourier analyses for the data and a user-selected subset of it, each having a context menu that outputs its contents to a new figure.
- A pop-up menu to change the type of data graph being displayed.
- A cell-selection callback that updates a column of statistics and a plot as the user highlights observations.
- Plots that portray periodicity in the entire data set and in selections of it.
- Context menus for the axes that let the user display their contents in a separate figure window.

Use this GUI—or one you adapt from it—to analyze and visualize time-series data containing periodic events.

Besides the tables and axes, the GUI features three panels, a push button to quit the application, static text, and functions for analyzing and plotting data. It opens as shown in the following figure.



	Year	Sunspots
1	1700	5
2	1701	11
3	1702	16
4	1703	23
5	1704	36
6	1705	58
7	1706	29
8	1707	20
9	1708	10
10	1709	8
11	1710	3
12	1711	0
13	1712	0
14	1713	2
15	1714	11
16	1715	27
17	1716	47
18	1717	63
19	1718	60
20	1719	39
21	1720	28
22	1721	26
23	1722	22
24	1723	11

	Population	Selection
N	288	
Min	0	
Max	190.2000	
Mean	48.4340	
Median	39	
Std Dev	39.4231	
1st Year	1700	
Last Year	1987	
Est. Period		

Note The `tablestat` example is based on the MATLAB `sunspots` demo and data set. Click here to view that demo (which is not GUI-based) in the MATLAB Help Browser.

Back to Top

View and Run the tablestat GUI

If you are reading this document in the MATLAB Help browser, you can access the example FIG-file and code file by clicking the following links. If you are reading this on the Web or in PDF form, go to the corresponding section in the MATLAB Help Browser to use the links.

If you intend to modify the layout or code of this GUI example, first save a copy of its code file and FIG-file to your current folder (you need write access to your current folder to do this). Follow these steps to copy the example files to your current folder and then to open them:

1. Click here to copy the files to your current folder
2. Type `guide tablestat` or click here to open the FIG-file in GUIDE
3. Type `edit tablestat` or click here to open the code file in the Editor

You can view the properties of any component by double-clicking it in the Layout Editor to open the Property Inspector for it. You can modify either the figure, the code, or both, and then save the GUI in your current folder using **File** > **Save as** from GUIDE. This saves both files, allowing you to rename them, if you choose.

To just inspect the GUI in GUIDE and run it, follow these steps instead:

1. Click here to add the example files to the MATLAB path (only for the current session).
2. Click here to run the `tablestat` GUI.

3. Click here to display the GUI in the GUIDE Layout Editor (read only).
4. Click here to display the GUI code file in the MATLAB Editor (read only).

> **Note** Do not save GUI files to the `examples` folder where you found them or you will overwrite the original files. If you want to save GUI files, use **File** > **Save as** from GUIDE, which saves both the GUI FIG-file and the GUI code file.

Summary of Tablestat Functions

The following table describes all the functions in `tablestat.m`, indicates what they do, and whether GUIDE created declarations for them or not. As the third column indicates, most of the callbacks generated by GUIDE have been customized. Click any function name to view its code in the MATLAB editor.

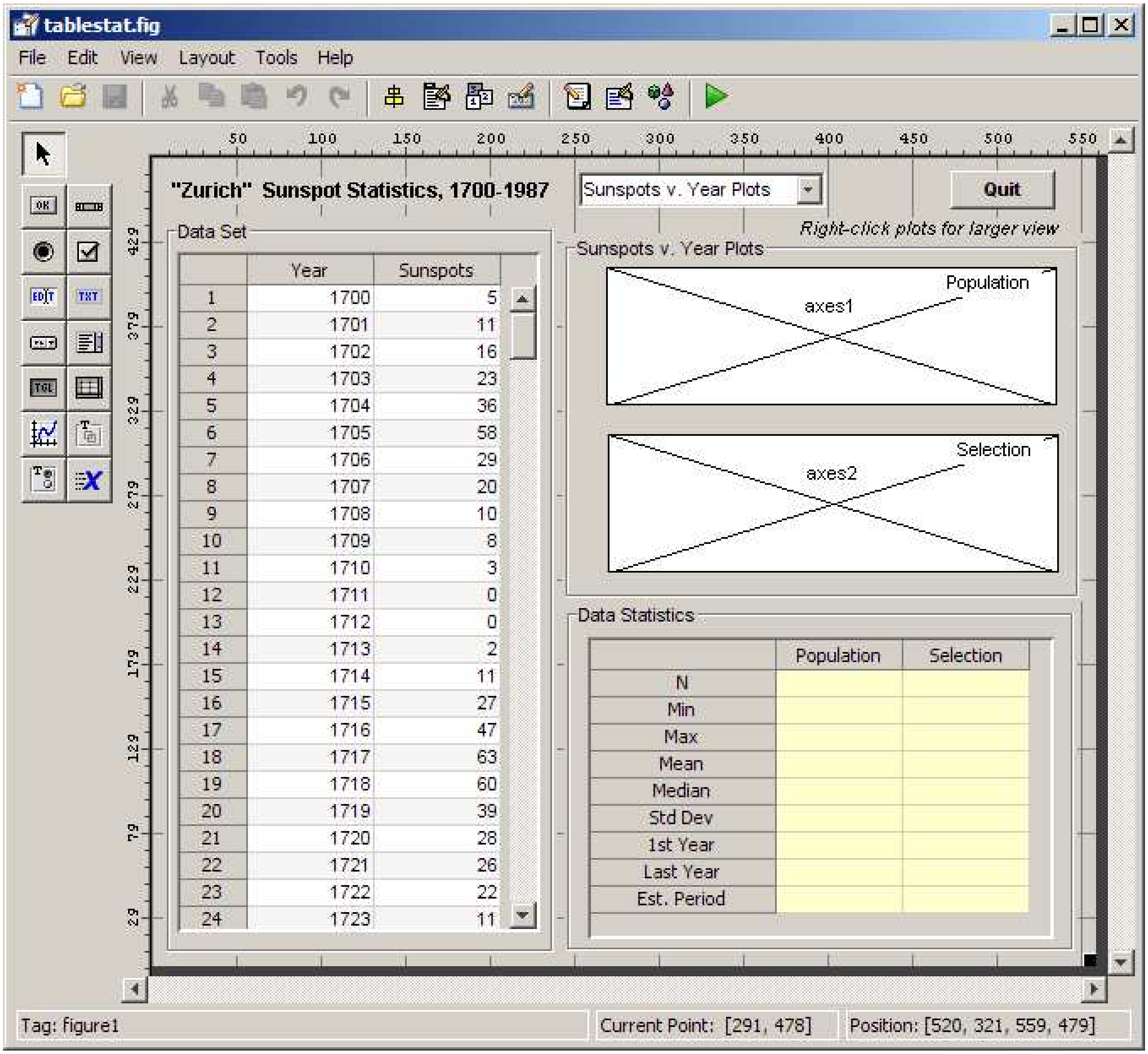
Function Name	Function Behavior	GUIDE-Generated?
`tablestat`	Main function	Yes; not customized
`tablestat_OpeningFcn`	Adds member to handles, generates population statistics and plots	Yes
`tablestat_OutputFcn`	Returns values when `tablestat` exits (not used)	Yes; not customized
`data_table_CellSelectionCallback`	Transforms table indices into unique row numbers, generates selection statistics an plot	Yes
`plot_type_Callback`	Refreshes displays when user selects new plot type	Yes
`plot_type_CreateFcn`	Manages appearance of pop-up menu during its creation	Yes; not customized
`plot_ax1_Callback`	Creates new figure with copy of axes1 plot in it	Yes
`plot_ax2_Callback`	Creates new figure with copy of axes2 plot in it	Yes
`refreshDisplays`	Controls updating of data statistics table and plots	No
`setStats`	Computes statistics for population or selection	No
`plotPeriod`	Generates plots (either time series or periodogram)	No
`quit_Callback`	Closes the figure	Yes

Back to Top

Designing the GUI

- Initializing the Data Table
- Computing the Data Statistics
- Specifying the Type of Data Plot
- Responding to Data Selections
- Updating the Statistics Table and the Graphs
- Displaying Graphs in New Figure Windows

In the GUIDE Layout Editor, the `tablestat` GUI looks like this.



	Year	Sunspots
1	1700	5
2	1701	11
3	1702	16
4	1703	23
5	1704	36
6	1705	58
7	1706	29
8	1707	20
9	1708	10
10	1709	8
11	1710	3
12	1711	0
13	1712	0
14	1713	2
15	1714	11
16	1715	27
17	1716	47
18	1717	63
19	1718	60
20	1719	39
21	1720	28
22	1721	26
23	1722	22
24	1723	11

	Population	Selection
N		
Min		
Max		
Mean		
Median		
Std Dev		
1st Year		
Last Year		
Est. Period		

Tag: figure1 Current Point: [291, 478] Position: [520, 321, 559, 479]

Perform the following steps in GUIDE and in the Property Inspector to generate the layout, thereby creating the following objects:

1. Using the Panel tool , drag out the three uipanels in the positions that are shown above. Keep the defaults for their `Tag` properties (which are `uipanel1`, `uipanel2`, and `uipanel3`). Create, in order:
   - A long panel on the left, renaming its `Title` to `Data Set` in the Property Inspector.
   - A panel on the lower right, half the height of the first panel, renaming its `Title` to `Data Statistics` in the Property Inspector.
   - A panel above the **Data Statistics** panel, renaming its `Title` to `Sunspots v. Year Plots` in the Property Inspector. This panel changes its name when the type of plot that is displayed changes.
2. Using the Table tool , drag out a uitable inside the **Data Set** panel, setting these properties in the Property Inspector to nondefault values:
   - `ColumnName`, set to `Year` and `Sunspot`.
   - `Data`, which you can set as described in the following section Initializing the Data Table.
   - `Tag`, set to `data_table`.
   - `TooltipString`, set to `Drag to select a range of 11 or more observations`.
   - `CellSelectionCallback`, which GUIDE automatically sets to `data_table_CellSelectionCallback` and declares in the code file when you click the

pencil-and-paper icon.

3. Drag out a second uitable, inside the **Data Statistics** panel, setting these properties in the Property Inspector:
   - `BackgroundColor` to yellow (using the color picker).
   - `ColumnName` to `Population` and `Selection`.
   - `Tag` to `data_stats`.
   - `TooltipString` to `statistics for table and selection`.
   - `RowName` to nine strings: `N`, `Min`, `Max`, `Mean`, `Median`, `Std Dev`, `1st Year`, `Last Year`, and `Est. Period`.

     You can conveniently set these labels with the Table Property Editor as follows:

     a. Double-click the **Data Statistics** table to open it in the Property Inspector.

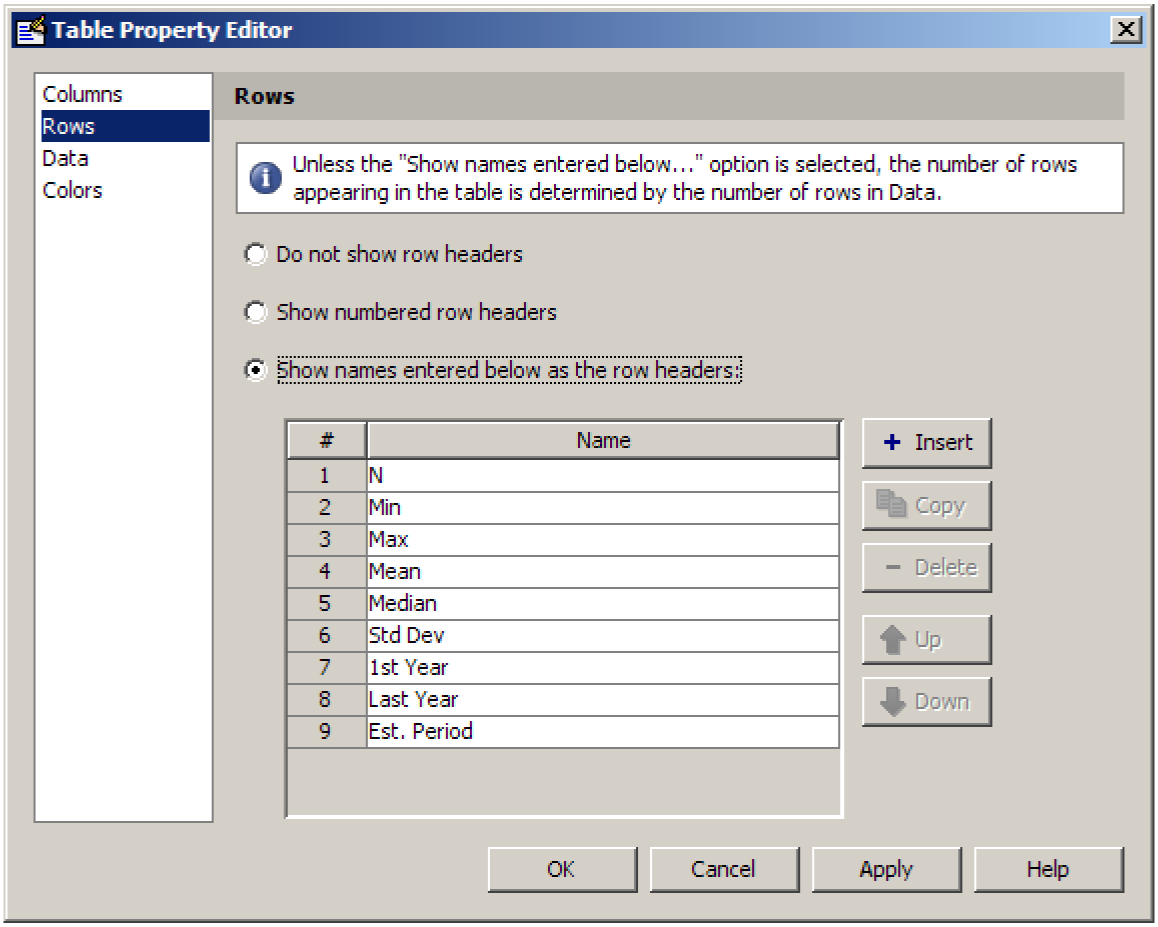
     b. In the Property Inspector, click the Table Property Editor icon to the right of the `RowName` property to open the Table Property Editor.

     c. In the Table Property Editor, select **Rows** from the list in the left-hand column.

     d. Select the bottom radio button, **Show names entered below as row headers**.

     e. Type the nine strings listed above in order on separate lines in the data entry pane and click **OK**.

     The Table Property Editor looks like this before you close it.



Unless the "Show names entered below..." option is selected, the number of rows appearing in the table is determined by the number of rows in Data.

○ Do not show row headers

○ Show numbered row headers

● Show names entered below as the row headers:

#	Name
1	N
2	Min
3	Max
4	Mean
5	Median
6	Std Dev
7	1st Year
8	Last Year
9	Est. Period

   The **Data Statistics** table does not use any callbacks.

4. Use the Axes tool to drag out an axes within the top half of the **Sunspots v. Year Plots** panel, leaving its name as `axes1`.
5. Drag out a second axes, leaving its name as `axes2` inside the **Sunspots v. Year Plots** panel, directly below the first axes.

   Leave enough space below each axes to display the *x*-axis labels.
6. Identify the axes with labels. Using the Text tool, drag out a small rectangle in the upper right corner of the upper axes (`axes1`). Double-click it, and in the Property Inspector, change its `String` property to `Population` and its `Tag` property to `poplabel`.
7. Place a second label in the lower axes (`axes2`), renaming this text object `Selection` and setting its `Tag` property

to sellabel.

8. Create a title for the GUI. Using the Text tool, drag out a static text object at the top left of the GUI, above the data table. Double-click it, and in the Property Inspector, change its String property to "Zurich" Sunspot Statistics, 1700-1987 and its FontWeight property to bold.
9. Add a prompt above the axes; place a text label just above the **Sunspots v. Year Plots** panel, near its right edge. Change its Tag property to newfig, its String property to Right-click plots for larger view and its FontAngle property to Italic.
10. Make a pop-up menu to specify the type of graph to plot. Using the Pop-up Menu tool, drag out a pop-up menu just above the **Sunspots v. Year** panel, aligning it to the panel's left edge. In the Property Inspector, set these properties:
    - String to

      ```
      Sunspots v. Year Plots
      FFT Periodogram Plots
      ```

    - Tag to plot_type
    - Tooltip to Choose type of data plot
    - Click the Callback property's icon. This creates a declaration called plot_type_Callback, to which you add code later on.
11. Select the Push Button tool OK, and drag out a push button in the upper right of the figure. In the Property Inspector, rename it to **Quit** and set up its callback as follows:
    - Double-click it and in the Property Inspector, set its Tag property to quit and its String property to Quit.
    - Click the Callback property to create a callback for the button in the code file tablestat.m. GUIDE sets the Callback of the **Quit** item to quit_Callback.
    - In the code file, for the quit_Callback function. enter:

      ```
      close(ancestor(hObject,'figure'))
      ```

    .
12. Save the GUI in GUIDE, naming it tablestat.fig. This action also saves the code file as tablestat.m.

Initializing the Data Table

You can use the Opening Function to load data into the table. In this example, however, you use GUIDE to put data into the **Data Set** table, so that the data becomes part of the figure after you save it. Initializing the table data causes the table to have the same number of rows and columns as the variable that it contains:

1. In the Command Window, access the sunspot demo data set. Enter:

   ```
   load sunspot.dat
   ```

   The variable sunspot, a 288-by-2 double array, is displayed in the MATLAB workspace.
2. Double-click the **Data Set** table to open the Property Inspector for the data table.
3. In the Property Inspector, click the Table Editor icon to the right of the Data property to open the Table Property Editor.
4. In the Table Property Editor, select **Table** from the list in the left-hand column.
5. Select the bottom radio button, **Change data value to the selected workspace variable below**.
6. From the list of workspace variables in the box below the radio button, select sunspot and click **OK**.

GUIDE inserts the sunspot data in the table.

> **Note** If you are designing a GUI like this but need to allow your users to load their own numeric data in place of the sunspot data, you need a way to interrogate the MATLAB workspace and present a list of variables to the user. The GUIDE example Accessing Workspace Variables from a List Box describes how to provide this kind of functionality with GUIDE. You can extend its functionality to list only variables of class double, of a certain dimensionality, etc.

Computing the Data Statistics

The Opening Function retrieves the preloaded data from the data table and calls the `setStats` subfunction to compute population statistics, and then returns them. The `data_table_CellSelectionCallback` performs the same action when the user selects more than 10 rows of the data table. The only difference between these two calls is what input data is provided and what column of the **Data Statistics** table is computed. Here is the `setStats` function:

```
function stats = setStats(table, stats, col, peak)
% Computes basic statistics for data table.
%   table  The data to summarize (a population or selection)
%   stats  Array of statistics to update
%   col    Which column of the array to update
%   peak   Value for the peak period, computed externally

stats{1,col} =   size(table,1);      % Number of rows
stats{2,col} =    min(table(:,2));
stats{3,col} =    max(table(:,2));
stats{4,col} =   mean(table(:,2));
stats{5,col} = median(table(:,2));
stats{6,col} =    std(table(:,2));
stats{7,col} =        table(1,1);    % First row
stats{8,col} =        table(end,1);  % Last row
if ~isempty(peak)
    stats{9,col} = peak;             % Peak period from FFT
end
```

> **Note** When assigning data to a uitable, use a cell array, as shown in the code for `setStats`. You can assign data that you retrieve from a uitable to a numeric array, however, only if it is entirely numeric. Storing uitable data in cell arrays enables tables to hold numbers, strings of characters, or combinations of them.

The `stats` matrix is a 9-by-2 cell array in which each row is a separate statistic computed from the `table` argument. The last statistic is not computed by `setStats`; it comes from the `plotPeriod` function when it computes and plots the FFT periodogram and is passed to `setStats` as the `peak` parameter.

Specifying the Type of Data Plot

At any time, the user of `tablestat` can choose either of two types of plots to display with the `plot_type` pop-up menu:

- Sunspots v. Year Plots — Time-series line graphs displaying sunspot occurrences year by year (default).
- Periodogram Plots — Graphs displaying the FFT-derived power spectrum of sunspot occurrences by length of cycle in years.

> **Note** For information on Fourier transforms, see Fourier Transforms and The FFT in One Dimension in the MATLAB Mathematics documentation.

When the plot type changes, one or both axes refresh. They always show the same kind of plot, but the bottom axes is initially empty and does not display a graph until the user selects at least 11 rows of the data table.

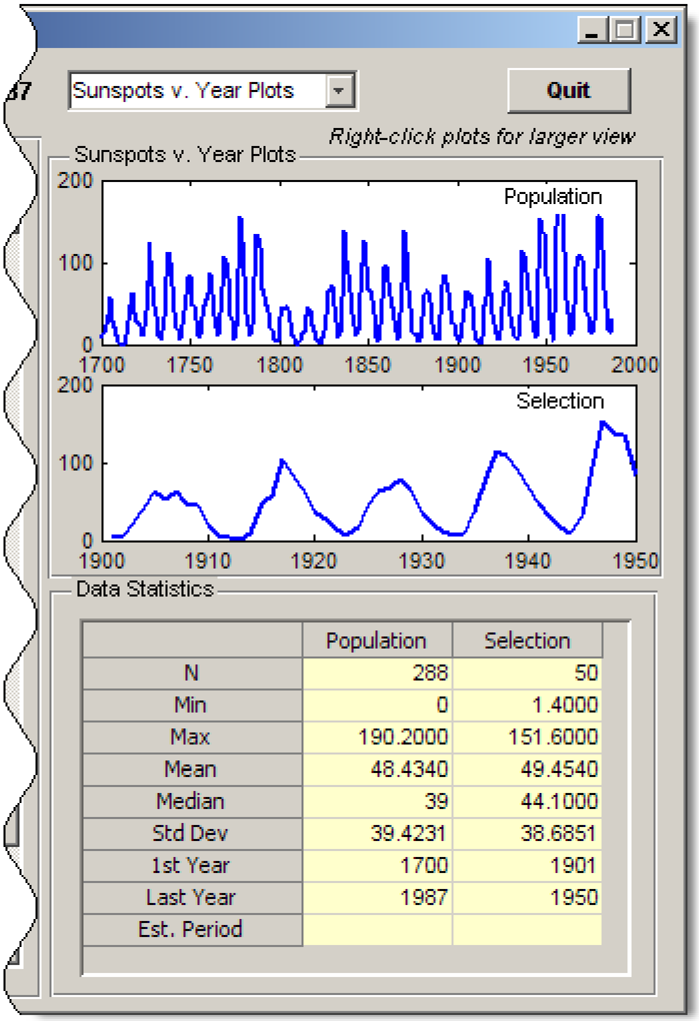
The callback of the `plot_type` control is `plot_type_Callback`. GUIDE generates it, and you must add code to it that updates plots appropriately. In the example, the callback consists of this code:

```
function plot_type_Callback(hObject, eventdata, handles)
% hObject    handle to plot_type (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% ---- Customized as follows ----
% Determine state of the pop-up and assign the appropriate string
% to the plot panel label
index = get(hObject,'Value');    % What plot type is requested?
strlist = get(hObject,'String');        % Get the choice's name
set(handles.uipanel3,'Title',strlist(index))  % Rename uipanel3

% Plot one axes at a time, changing data; first the population
table = get(handles.data_table,'Data'); % Obtain the data table
refreshDisplays(table, handles, 1)
```

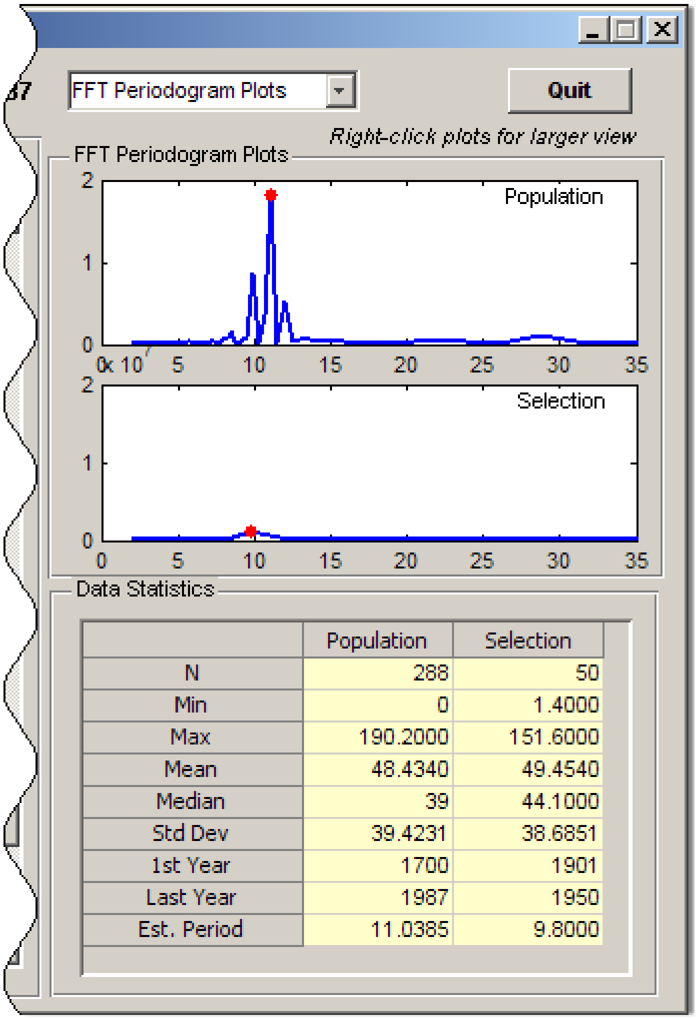
```
% Now compute stats for and plot the selection, if needed.
% Retrieve the stored event data for the last selection
selection = handles.currSelection;
if length(selection) > 10  % If more than 10 rows selected
    refreshDisplays(table(selection,:), handles, 2)
else
    % Do nothing; insufficient observations for statistics
end
```

The function updates the Data Statistics table and the plots. To perform the updates, it calls the `refreshDisplays` function twice, which is a custom function added to the GUI code file. In between the two calls, the `refreshDisplays` function retrieves row indices for the current selection from the `currSelection` member of the `handles` structure, where they were cached by the `data_table_CellSelectionCallback`.

You can see the effect of toggling the plot type in the two illustrations that follow. The one on the left shows the Sunspots v. Year plots, and the one on the right shows the FFT Periodograms Plots. The selection in both cases is the years 1901–1950.



	Population	Selection
N	288	50
Min	0	1.4000
Max	190.2000	151.6000
Mean	48.4340	49.4540
Median	39	44.1000
Std Dev	39.4231	38.6851
1st Year	1700	1901
Last Year	1987	1950
Est. Period		



	Population	Selection
N	288	50
Min	0	1.4000
Max	190.2000	151.6000
Mean	48.4340	49.4540
Median	39	44.1000
Std Dev	39.4231	38.6851
1st Year	1700	1901
Last Year	1987	1950
Est. Period	11.0385	9.8000

Responding to Data Selections

The **Data Set** table has two columns: **Year** and **Sunspots**. The data tables's Cell Selection Callback analyzes data from its second column, regardless of which columns the user highlights. The `setStats` function (not generated by GUIDE) computes summary statistics observations from the second column for insertion into the **Data Statistics** table on the right. The `plotPeriod` function (not generated by GUIDE) plots either the raw data or a Fourier analysis of it.

The `data_table_CellSelectionCallback` function manages the application's response to users selecting ranges of data. Ranges can be contiguous rows or separate groups of rows; holding down the **Ctrl** key lets users add discontiguous rows to a selection. Because the Cell Selection Callback is triggered as long as the user holds the left mouse button down within the table, the selection statistics and lower plot are refreshed until selection is completed.

Selection data is generated during mouseDown events (mouse drags in the data table). The uitable passes this stream of cell indices (but not cell values) via the `eventdata` structure to the `data_table_CellSelectionCallback` callback. The callback's code reads the indices from the `Indices` member of the `eventdata`.

When the callback runs (for each new value of `eventdata`), it turns the event data into a set of rows:

```
selection = eventdata.Indices(:,1);
selection = unique(selection);
```

The event data contains a sequence of `[row, column]` indices for each table cell currently selected, one cell per line. The preceding code trims the list of indices to a list of selected rows, removing column indices. Then it calls the `unique` MATLAB function to eliminate any duplicate row entries, which arise whenever the user selects both columns. For example, suppose `eventdata.Indices` contains:

```
  1    1
  2    1
  3    1
  3    2
  4    2
```

This indicates that the user selected the first three rows in column one (Year) and rows three and four in column two (Sunspots) by holding down the **Ctrl** key when selecting numbers in the second column. The preceding code transforms the indices into this vector:

```
1
2
3
4
```

This vector enumerates all the selected rows. If the selection includes less than 11 rows (as it does here) the callback returns, because computing statistics for a sample that small is not useful.

When the selection contains 11 or more rows, the data table is obtained, the selection is cached in the `handles` structure, and the `refreshDisplays` function is called to update the selection statistics and plot, passing the portion of the table that the user selected:

```
table = get(hObject,'Data');
handles.currSelection = selection;
guidata(hObject,handles)
refreshDisplays(table(selection,:), handles, 2)
```

Caching the list of rows in the selection is necessary because the user can force selection data to be replotted by changing plot types. As the `plot_type_Callback` has no access to the data table's event data, it requires a copy of the most recent selection.

Updating the Statistics Table and the Graphs

The code must update the Data Statistics table and the graphs above it when:

- The GUI is initialized, in its `tablestat_OpeningFcn`.
- The user selects cells in the data table, its `data_table_CellSelectionCallback`.
- The user selects a different plot type, in the `plot_type_Callback`.

In each case, the `refreshDisplays` function is called to handle the updates. It in turn calls two other custom functions:

- `setStats` — Computes summary statistics for the selection and returns them.
- `plotPeriod` — Plots the type of graph currently requested in the appropriate axes.

The `refreshDisplays` function identifies the current plot type and specifies the axes to plot graphs into. After calling `plotPeriod` and `setStats`, it updates the **Data Statistics** table with the recomputed statistics. Here is the code for `refreshDisplays`:

```
function refreshDisplays(table, handles, item)
if isequal(item,1)
    ax = handles.axes1;
elseif isequal(item,2)
    ax = handles.axes2;
end
peak = plotPeriod(ax, table,...
              get(handles.plot_type,'Value'));
stats = get(handles.data_stats, 'Data');
stats = setStats(table, stats, item, peak);
set(handles.data_stats, 'Data', stats);
```

If you are reading this document in the MATLAB Help Browser, click the names of the functions underlined above to see their complete code (including comments) in the MATLAB Editor.

Displaying Graphs in New Figure Windows

- Creating Two Context Menus
- Attaching the Context Menus to Axes
- Coding the Context Menu Callbacks
- Using the Plot in New Window Feature

The `tablestat` GUI contains code to display either of its graphs in a larger size in a new figure window when the user right-clicks either axes and selects the pop-up menu item, **Open plot in new window**. The static text string (tagged `newfig`) above the plot panel, **Right-click plots for larger view**, informs the user that this feature is available.

The axes respond by:

1. Creating a new figure window.
2. Copying their contents to a new axes parented to the new figure.
3. Resizing the new axes to use 90% of the figure's width.
4. Constructing a title string and displaying it in the new figure.
5. Saving the figure and axes handles in the `handles` structure for possible later use or destruction.

> **Note** Handles are saved for both plots, but each time a new figure is created for either of them, the new handles replace the old ones, if any, making previous figures inaccessible from the GUI.

Creating Two Context Menus. To create the two context menus, from the GUIDE **Tools** menu, select the **Menu Editor**. After you create the two context menus, attach one to the each axes, `axes1` and `axes2`. In the Menu Editor, for each menu:

1. Click the **Context Menus** tab to select the type of menu you are creating.
2. Click the **New Context Menu** icon .

   This creates a context menu in the Menu Editor workspace called `untitled`. It has no menu items and is not attached to any GUI object yet.
3. Select the new menu and in the **Tag** edit field in the **Menu Properties** panel, type `plot_axes1`.
4. Click the **New Menu Item** icon .

   A menu item is displayed underneath the `plot_axes1` item in the Menu Editor workspace.
5. In the **Menu Properties** panel, type `Open plot in new window` for **Label** and `plot_ax1` for **Tag**. Do not set anything else for this item.
6. Repeat the last four steps to create a second context menu:
   - Make the **Tag** for the menu `plot_axes2`.
   - Create a menu item under it and make its **Label** `Open plot in new window` and assign it a **Tag** of `plot_ax2`.
7. Click **OK** to save your menus and exit the Menu Editor.

For more information about using the Menu Editor, see Creating Menus.

Attaching the Context Menus to Axes. Add the context menus you just created to the axes:

1. In the GUIDE Layout Editor, double-click axes1 (the top axes in the upper right corner) to open it in the Property Inspector.
2. Click the right-hand column next to UIContextMenu to see a drop-down list.
3. From the list, select plot_axes1.

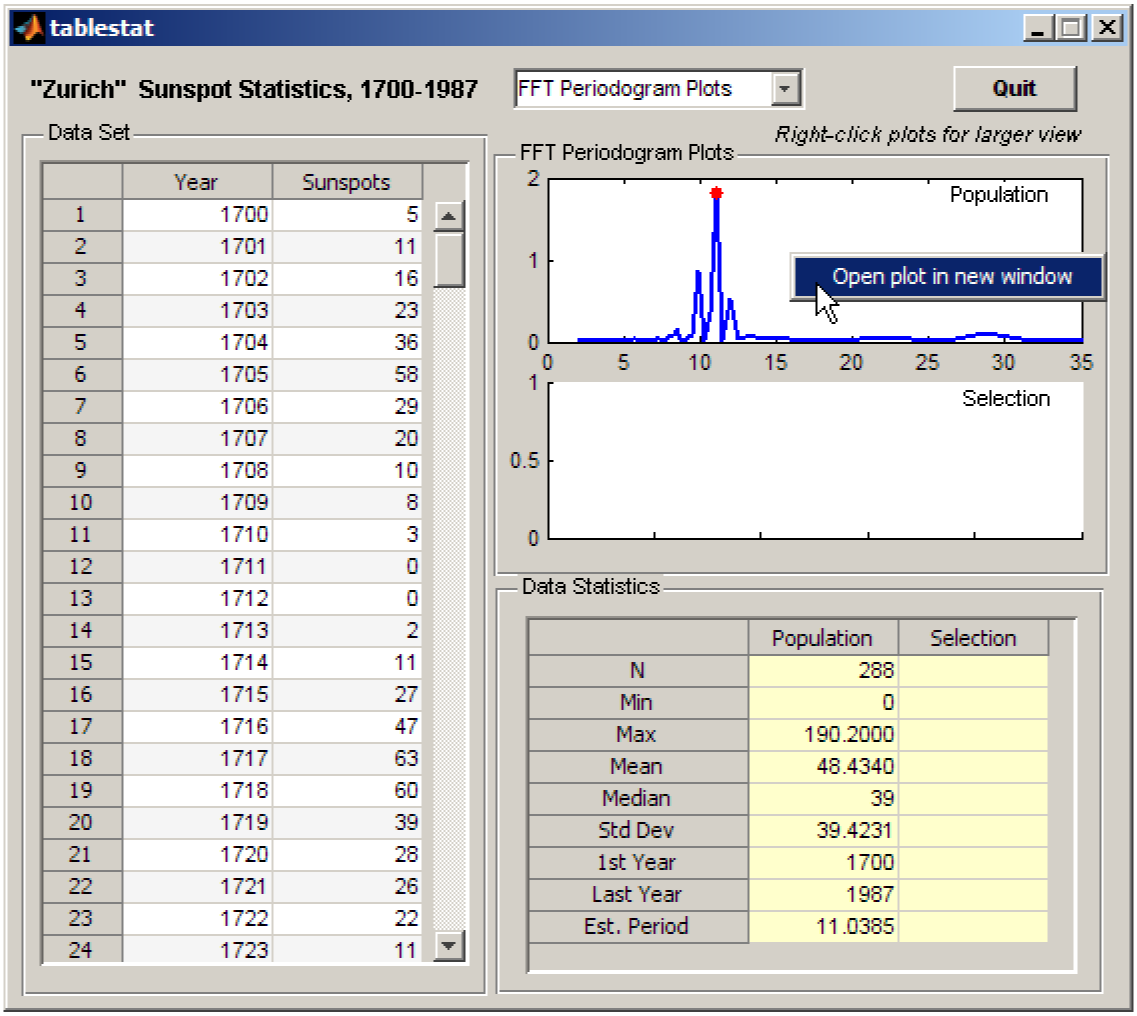
Perform the same steps for axes2, but select plot_axes2 as its UIContextMenu.

Coding the Context Menu Callbacks. The two context menu items perform the same actions, but create different objects. Each has its own callback. Here is the plot_ax1_Callback callback for axes1:

```
function plot_ax1_Callback(hObject, eventdata, handles)
% hObject    handle to plot_ax1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
% Displays contents of axes1 at larger size in a new figure

% Create a figure to receive this axes' data
axes1fig = figure;
% Copy the axes and size it to the figure
axes1copy = copyobj(handles.axes1,axes1fig);
set(axes1copy,'Units','Normalized',...
              'Position',[.05,.20,.90,.60])
% Assemble a title for this new figure
str = [get(handles.uipanel3,'Title') ' for ' ...
       get(handles.poplabel,'String')];
title(str,'Fontweight','bold')
% Save handles to new fig and axes in case
% we want to do anything else to them
handles.axes1fig = axes1fig;
handles.axes1copy = axes1copy;
guidata(hObject,handles);
```

The other callback, plot_ax2_Callback, is identical to plot_ax1_Callback, except that all instances of 1 in the code are replaced by 2, and poplabel is replaced with sellabel. The poplabel and sellabel objects are the **Population** and **Selection** labels on axes1 and axes2, respectively. These strings are appended to the current Title for uipanel3 to create a title for the plot in the new figure axes1fig or axes2fig.

Using the Plot in New Window Feature. Whenever the user right-clicks one of the axes and selects Open plot in new window, a new figure is generated containing the graph in the axes. The callbacks do not check whether a graph exists in the axes (axes2 is empty until the user selects cells in the **Data Set**) or whether a previously opened figure contains the same graph. A new figure is always created and the contents of axes1 or axes2 are copied into it. For example, here the user right-clicks a periodogram in axes1 and chooses Open plot in new window.



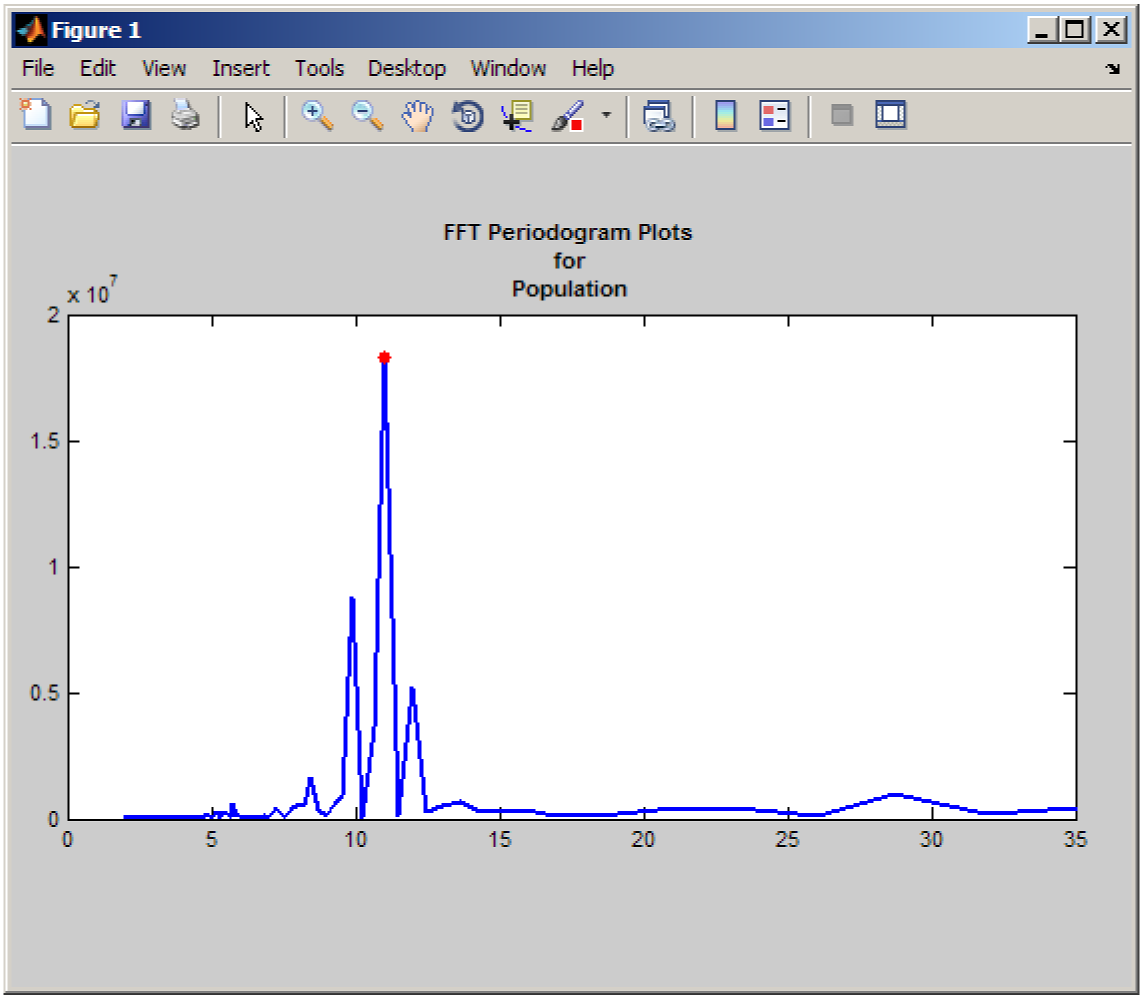
"Zurich" Sunspot Statistics, 1700-1987

Right-click plots for larger view

	Year	Sunspots
1	1700	5
2	1701	11
3	1702	16
4	1703	23
5	1704	36
6	1705	58
7	1706	29
8	1707	20
9	1708	10
10	1709	8
11	1710	3
12	1711	0
13	1712	0
14	1713	2
15	1714	11
16	1715	27
17	1716	47
18	1717	63
19	1718	60
20	1719	39
21	1720	28
22	1721	26
23	1722	22
24	1723	11

	Population	Selection
N	288	
Min	0	
Max	190.2000	
Mean	48.4340	
Median	39	
Std Dev	39.4231	
1st Year	1700	
Last Year	1987	
Est. Period	11.0385	

Upon Clicking **Open plot in new window**, a new figure is displayed with the following content.



It is the user's responsibility to remove the new window when it is no longer needed. The context menus can be programmed to do this. Because their callbacks call `guidata` to save the handle of the last figure created for each of the GUI's axes, another callback can delete or reuse either figure. For example, the `plot_ax1_Callback` and `plot_ax2_Callback` callbacks could check `guidata` for a valid axes handle stored in `handles.axes1copy` or `handles.axes2copy`, and reuse the axes instead of creating a new figure.

Back to Top

Extending Tablestat

You can extend the Tablestat example GUI in several ways to make it more capable:

- Enable the GUI to read in any data matrix in the MATLAB workspace or a data file. To do this:
  - Provide a file dialog box or an input dialog box and code capable of filtering out nonnumeric, nonmatrix data.
  - Provide default names for columns and a way for the user to rename them.
- Enable the user to select which data columns to analyze and plot:
  - A way for the user to indicate which columns to use as independent (*x*, normally) and dependent (*y*, normally) variables.
  - A uicontrol or menu to identify which columns to process, as Tablestat already uses cell selection to identify subsets of data.
- Program the GUI to open a plot in a new figure window when the user double-clicks one of its axes (instead of or in addition to using a context menu to do this). This involves:
  - Providing a `ButtonDownFcn` for each axes that obtains the current `SelectionType` property of the figure and determining if one or two clicks occurred.

    > **Tip** Use `get(gcbf, 'SelectionType')` in the callback and check for a value of `'open'`.

  - Setting the `NextPlot` property of `axes1` and `axes2` to `ReplaceChildren` to avoid deleting the handle of the `ButtonDownFcn` from the axes every time a graph is plotted into it (which always occurs when `NextPlot` is `Add`, the default).

- Generating a new figure and axes, and copying the contents of the clicked axes to it, as the context menu callbacks currently do.

▲ Back to Top

Was this topic helpful? Yes No

◀ GUI for Animating a 3-D View

List Box Directory Reader ▶