PlotTbl: Plot Data Stored in a Table

Jeff Miller (miller at psy.otago.ac.nz)

May 31, 2020

Contents

1 Overview

PlotTbl is a general-purpose function for creating a figure from data stored in a table. It was designed to be used with tables in which some variables hold the X/Y values to be plotted and other variables hold numeric codes that distinguish different conditions. With such data, PlotTbl makes it easy to plot X/Y relationships for each condition separately, with different conditions distinguished by different line types

(e.g., solid/dashed), marker types, colors, subplots, etc. Many options are provided for control over titles, X and Y axis labels, legends, etc.

Screenshots in the README.md file on GitHub show a set of example figures produced with a data table. All but one of the figures were produced by a single PlotTbl function call.

Users needing even more fine-grained control over individual subplots can call the function SubplotTbl, which plots only one subplot within a subplot-type figure.

2 Requirements

You need a version of MATLAB that supports the "table" data type (R2013b or newer). You also need my MATLAB package ExtractNameVal available at https://github.com/milleratotago/ExtractNameVal

3 License

4 Syntax

PlotTbl is called with one of these four basic parameter configurations:

1. PlotTbl(Tbl,Name,Value);
2. PlotTbl(Tbl,sX,sY,Name,Value);
3. PlotTbl(Tbl,sX,Name,Value);
4. PlotTbl(Tbl,sY,Name,Value);

Tbl is the data table containing the data to be plotted.

sX and sY are strings indicating the names of the X and Y variables to be plotted. In some cases the X and/or Y variable names are specified by name-value pairs (see various option names ending in XVars and YVars below), and in these cases the sX and/or sY parameters *must* be omitted.

Name,Value represents a series of name-value pair arguments.

5 Examples

Suppose that you have a table AHWDat containing the average heights and weights of male and female children of three different nationalities in six different age groups (see Demo.m). Here is the format of the table (the heights and weights are not intended to be realistic):

Age	Gender	Nationality	AvgHeight	AvgWeight
2	1	1	114	66
2	1	2	116	64
2	1	3	111	68
2	2	1	125	73
2	2	2	124	75
2	2	3	122	75
4	1	1	117	72
4	1	2	118	72
...	...	...	...	...

AHWDat.Gender is 1/2 to code male/female, and AHWDat.Nationality is 1/2/3 to code three nationalities (e.g., Germany, France, Spain), and AHWDat.Age holds the ages (2, 4, 6, 8, 10, 12). The variables AHWDat.AvgWeight, and AHWDat.AvgHeight hold the average weights and heights of the children of each gender, nationality, and age group.

Example: *PlotTbl(AHWDat,'Age','AvgWeight','SubplotRowsCodeVar','Gender','SubplotColsCodeVar','Nationality')*

This command produces a figure with six subplots (2 rows x 3 columns). The two rows of subplots correspond to genders 1 and 2, and the three columns of subplots correspond to the three nationalities. Each plot has Age on the X axis and AvgWeight on the Y axis.

Example: *PlotTbl(AHWDat,'Age','SubplotRowsYVars',{'AvgWeight','AvgHeight'},...*
'SubplotColsCodeVar','Nationality','LineTypeCodeVar','Gender')

This command produces a figure with 6 subplots of AvgWeight versus Age in the top row and AvgHeight versus Age in the bottom row. The three nationalities are the three columns of subplots. There are two lines per subplot—one for each Gender—with the lines distinguished by line type (i.e., solid, dashed).

Example: *PlotTbl(AHWDat,'Age','SubplotRowsYVars',{'AvgWeight','AvgHeight'},...*
'SubplotColsCodeVar','Nationality','LineTypeCodeVar','Gender','LineTypeLegend',{'Male','Female'})

This command produces the same plot as the previous one except that the two Gender values are specified as "Male" and "Female" in the legend.

For more examples, see Demo.m.

6 Name-value Pairs for Design Element Control

It is convenient to consider the name-value pairs as falling into three groups: design element control, axis control, and legend control. The largest and most important group provides design element control.

There are seven sets of name-value pair options to control the organization of the figure in terms of its design elements, namely:

- line types, such as solid versus dotted (LineType)
- marker types, such as squares versus circles (MarkerType)
- color (Color)
- line width (LineWidth)
- marker size (MarkerSize)
- subplot rows (SubplotRows)
- subplot columns (SubplotCols)

All seven are controlled with basically the same set of options.

6.1 Name-value pairs for line type control

Name-value pair: *LineTypeCodeVar — variable name* (or equivalently)
Name-value pair: *LineType — variable name*

This name-value pair indicates that a different line type should be used for each different value of the indicated table variable. The variable name is specified as a character vector, say sVar.

Example: *PlotTbl(...,'LineTypeCodeVar','Gender')*

In this example different line types will be used to plot the data for males versus females (i.e., Gender==1 versus Gender==2).

Name-value pair: *LineTypeXVars — cell array of variable names* (or equivalently)
Name-value pair: *LineTypeX — cell array of variable names*

This name-value pair indicates that different line types should be used to distinguish different variables in the table, with those variables being used as X's in the X/Y plots.

Example: *PlotTbl(...,'LineTypeXVars',{'AvgHeight' 'AvgWeight'})*

In this example different line types will be used to distinguish lines whose X values are AvgHeight versus lines whose X values are AvgWeight.

Name-value pair: *LineTypeYVars — cell array of variable names* (or equivalently)
Name-value pair: *LineTypeY — cell array of variable names*

This name-value pair indicates that different line types should be used to distinguish different variables in the table, with those variables being used as Y's in the X/Y plots.

Example: *PlotTbl(...,'LineTypeYVars',{'AvgHeight' 'AvgWeight'})*

In this example different line types will be used to distinguish lines whose Y values are AvgHeight versus lines whose Y values are AvgWeight.

Name-value pair: *LineTypeXYVars — cell array of pairs of variable names* (or equivalently)
Name-value pair: *LineTypeXY — cell array of pairs of variable names*

This name-value pair indicates that different line types should be used to distinguish different combinations of variables in the table. Two variables are listed; the first is used as X's in the X/Y plots, and the second is used as Y's.

Example: *PlotTbl(...,'LineTypeXYVars',{'AvgHeight' 'AvgWeight' 'AvgWeight' 'AvgHeight'})*

In this example different line types will be used to distinguish plots of AvgHeight as X and AvgWeight as Y, as compared with plots of AvgWeight as X and AvgHeight as Y.

Note: You should never specify *more than one* of the four options LineTypeCodeVar, LineTypeXVars, LineTypeYVars, and LineTypeXYVars. That is, line types can only be distinguished in one of these ways—not several. If you want the plots to be distinguished in more than one way, use other design elements.

Name-value pair: *LineTypeSpecs — cell array of line type specifications*

This name-value pair indicates that you want to replace PlotTbl's default set of line type specifications with a new set. Use MATLAB's standard line type specifications in whatever order you prefer. The line types will be used in order by values of the CodeVar or the order of the variables listed for LineTypeXVars or LineTypeYVars.

Example: *PlotTbl(...,'LineTypeSpecs',{':' '-.'})*

In this example the first line will be dotted and the second line will be dash/dot.

Name-value pair: *LineTypeOrder — 'stable' or 'sorted'*

This name-value pair is only applicable when used in conjunction with the 'LineTypeCodeVar' pair. It indicates whether you want PlotTbl to assign the different values of the indicated code variable to the

different line types in the order in which they appear in the table (i.e., 'stable', which is the default) or whether you want them assigned in sorted numerical order.

Example: *PlotTbl(...,'LineTypeOrder','sorted')*

In this example the values of the indicated code variable will be assigned to line types in numerical order, regardless of the order in which they appear in the table.

Name-value pair: *LineTypeLegend — cell array of legend labels*

This name-value pair indicates that you want to replace PlotTbl's default labels of the line types with the new labels indicated in your cell array. The order of the legend labels in the cell array should correspond to the order in which the different values of the indicated variable are assigned to the different line types, as determined by the LineTypeOrder name-value pair.

Example: *PlotTbl(...,'LineTypeLegend',{'Male' 'Female'})*

In this example the legends for the first and second line types will be "Male" and "Female", respectively, rather than the defaults of "Gender=1" and "Gender=2". Obviously, this option would make sense if used in combination with the name-value pair "'LineTypeCodeVar','Gender"'.

Note 1: It is sometimes frustrating that MATLAB only provides four line types. To work around this limitation, PlotTbl defines some additional line types that have the same style as the built-in line types (i.e., solid, dashed, etc) but vary in line thickness. Specifically, these added line types are specified by strings just like the additional ones but with the character '2', '3', or '4' added on at the end, with thickness increasing according to the size of this number. These added line types are used by default in PlotTbl, so if you need (say) six line types you will get regular MATLAB ones for the first four and then you will get thicker versions for the next two.

Note 2: Name-value pairs analogous to LineTypeCodeVar, LineTypeXVars, LineTypeYVars, LineTypeSpecs, and LineTypeLegend, are available for controlling marker types, colors, line widths, marker sizes, and subplots as described in the following subsections.

6.2 Name-value pairs for marker type control

Name-value pair: *MarkerTypeCodeVar — variable name* (or equivalently)
Name-value pair: *MarkerType — variable name*

This name-value pair indicates that a different marker type should be used for each different value of the indicated table variable. The variable name is specified as a character vector, say sVar.

Example: *PlotTbl(...,'MarkerTypeCodeVar','Gender')*

In this example different marker types will be used to plot the data for males versus females (i.e., Gender==1 versus Gender==2).

Name-value pair: *MarkerTypeXVars — cell array of variable names* (or equivalently)
Name-value pair: *MarkerTypeX — cell array of variable names*

This name-value pair indicates that different marker types should be used to distinguish different variables in the table, with those variables being used as X's in the X/Y plots.

Example: *PlotTbl(...,'MarkerTypeXVars',{'AvgHeight' 'AvgWeight'})*

In this example different marker types will be used to distinguish lines whose X values are AvgHeight versus lines whose X values are AvgWeight.

Name-value pair: *MarkerTypeYVars — cell array of variable names* (or equivalently)
Name-value pair: *MarkerTypeY — cell array of variable names*

This name-value pair indicates that different marker types should be used to distinguish different variables in the table, with those variables being used as Y's in the X/Y plots.

Example: *PlotTbl(...,'MarkerTypeYVars',{'AvgHeight' 'AvgWeight'})*

In this example different marker types will be used to distinguish lines whose Y values are AvgHeight versus lines whose Y values are AvgWeight.

Name-value pair: *MarkerTypeXYVars — cell array of pairs of variable names* (or equivalently)
Name-value pair: *MarkerTypeXY — cell array of pairs of variable names*

This name-value pair indicates that different marker types should be used to distinguish different combinations of variables in the table. Two variables are listed; the first is used as X's in the X/Y plots, and the second is used as Y's.

Example: *PlotTbl(...,'MarkerTypeXYVars',{'AvgHeight' 'AvgWeight' 'AvgWeight' 'AvgHeight'})*

In this example different marker types will be used to distinguish plots of AvgHeight as X and AvgWeight as Y, as compared with plots of AvgWeight as X and AvgHeight as Y.

Note: You should never specify *more than one* of the four options MarkerTypeCodeVar, MarkerTypeXVars, MarkerTypeYVars, and MarkerTypeXYVars. That is, marker types can only be distinguished in one of these ways—not several. If you want the plots to be distinguished in more than one way, use other design elements.

Name-value pair: *MarkerTypeSpecs — string list of marker type specifications*

This name-value pair indicates that you want to replace PlotTbl's default set of marker type specifications with a new set. Use MATLAB's standard marker type specifications in whatever order you prefer. The marker types will be used in order by values of the CodeVar or the order of the variables listed for MarkerTypeXVars or MarkerTypeYVars.

Example: *PlotTbl(...,'MarkerTypeSpecs','so*')*

In this example the first marker will be the square, the second will be the circle, and the third will be the asterisk.

Name-value pair: *MarkerTypeOrder — 'stable' or 'sorted'*

This name-value pair is only applicable when used in conjunction with the 'MarkerTypeCodeVar' pair. It indicates whether you want PlotTbl to assign the different values of the indicated code variable to the different marker types in the order in which they appear in the table (i.e., 'stable', which is the default) or whether you want them assigned in sorted numerical order.

Example: *PlotTbl(...,'MarkerTypeOrder','sorted')*

In this example the values of the indicated code variable will be assigned to marker types in numerical order, regardless of the order in which they appear in the table.

Name-value pair: *MarkerTypeLegend — cell array of legend labels*

This name-value pair indicates that you want to replace PlotTbl's default labels of the marker types with the new labels indicated in your cell array. The order of the legend labels in the cell array should correspond to the order in which the different values of the indicated variable are assigned to the different marker types, as determined by the MarkerTypeOrder name-value pair.

Example: *PlotTbl(...,'MarkerTypeLegend',{'Male' 'Female'})*

In this example the legends for the first and second marker types will be "Male" and "Female", respectively, rather than the defaults of "Gender=1" and "Gender=2". Obviously, this option would make sense if used in combination with the name-value pair "'MarkerTypeCodeVar','Gender"'.

6.3 Name-value pairs for color control

Name-value pair: *ColorCodeVar — variable name* (or equivalently)
Name-value pair: *Color — variable name*

This name-value pair indicates that a different color should be used for each different value of the indicated table variable. The variable name is specified as a character vector, say sVar.

Example: *PlotTbl(...,'ColorCodeVar','Gender')*

In this example different colors will be used to plot the data for males versus females (i.e., Gender==1 versus Gender==2).

Name-value pair: *ColorXVars — cell array of variable names* (or equivalently)
Name-value pair: *ColorX — cell array of variable names*

This name-value pair indicates that different colors should be used to distinguish different variables in the table, with those variables being used as X's in the X/Y plots.

Example: *PlotTbl(...,'ColorXVars',{'AvgHeight' 'AvgWeight'})*

In this example different colors will be used to distinguish lines whose X values are AvgHeight versus lines whose X values are AvgWeight.

Name-value pair: *ColorYVars — cell array of variable names* (or equivalently)
Name-value pair: *ColorY — cell array of variable names*

This name-value pair indicates that different colors should be used to distinguish different variables in the table, with those variables being used as Y's in the X/Y plots.

Example: *PlotTbl(...,'ColorYVars',{'AvgHeight' 'AvgWeight'})*

In this example different colors will be used to distinguish lines whose Y values are AvgHeight versus lines whose Y values are AvgWeight.

Name-value pair: *ColorXYVars — cell array of pairs of variable names* (or equivalently)
Name-value pair: *ColorXY — cell array of pairs of variable names*

This name-value pair indicates that different colors should be used to distinguish different combinations of variables in the table. Two variables are listed; the first is used as X's in the X/Y plots, and the second is used as Y's.

Example: *PlotTbl(...,'ColorXYVars',{'AvgHeight' 'AvgWeight' 'AvgWeight' 'AvgHeight'})*

In this example different colors will be used to distinguish plots of AvgHeight as X and AvgWeight as Y, as compared with plots of AvgWeight as X and AvgHeight as Y.

Note: You should never specify *more than one* of the four options ColorCodeVar, ColorXVars, ColorYVars, and ColorXYVars. That is, colors can only be distinguished in one of these ways—not several. If you want the plots to be distinguished in more than one way, use other design elements.

Name-value pair: *ColorSpecs — string list of color specifications*

This name-value pair indicates that you want to replace PlotTbl's default set of color specifications with a new set. Use MATLAB's standard color specifications in whatever order you prefer. The colors will be used in order by values of the CodeVar or the order of the variables listed for ColorXVars or ColorYVars.

Example: *PlotTbl(...,'ColorSpecs','rgbk')*

In this example the first line will be red, the second green, the third blue, and the fourth black.

Name-value pair: *ColorOrder — 'stable' or 'sorted'*

This name-value pair is only applicable when used in conjunction with the 'ColorCodeVar' pair. It indicates whether you want PlotTbl to assign the different values of the indicated code variable to the different colors in the order in which they appear in the table (i.e., 'stable', which is the default) or whether you want them assigned in sorted numerical order.

Example: *PlotTbl(...,'ColorOrder','sorted')*

In this example the values of the indicated code variable will be assigned to colors in numerical order, regardless of the order in which they appear in the table.

Name-value pair: *ColorLegend — cell array of legend labels*

This name-value pair indicates that you want to replace PlotTbl's default labels of the colors with the new labels indicated in your cell array. The order of the legend labels in the cell array should correspond to the

order in which the different values of the indicated variable are assigned to the different colors, as determined by the ColorOrder name-value pair.

Example: *PlotTbl(...,'ColorLegend',{'Male' 'Female'})*

In this example the legends for the first and second colors will be "Male" and "Female", respectively, rather than the defaults of "Gender=1" and "Gender=2". Obviously, this option would make sense if used in combination with the name-value pair "'ColorCodeVar','Gender"'.

Specifying your own RGB values. Alternatively, you may specify colors like this:

Name-value pair: *ColorSpecs — cell array of RGB color specifications*

This name-value pair indicates that you want to replace PlotTbl's default set of color specifications with a new set. Use MATLAB's standard RGB color specifications in whatever order you prefer. Your colors will be used in order by values of the CodeVar or the order of the variables listed for ColorXVars or ColorYVars.

Example: *PlotTbl(...,'ColorSpecs',{[0.4 0.4 0.4], [0.6 0.6 0.6], [0.8 0.8 0.8]})*

In this example the first line will be the darkest, the second medium, and the third lightest.

6.4 Name-value pairs for line width control

Name-value pair: *LineWidthCodeVar — variable name* (or equivalently)
Name-value pair: *LineWidth — variable name*

This name-value pair indicates that a different line width should be used for each different value of the indicated table variable. The variable name is specified as a character vector, say sVar.

Example: *PlotTbl(...,'LineWidthCodeVar','Gender')*

In this example different line widths will be used to plot the data for males versus females (i.e., Gender==1 versus Gender==2).

Name-value pair: *LineWidthXVars — cell array of variable names* (or equivalently)
Name-value pair: *LineWidthX — cell array of variable names*

This name-value pair indicates that different line widths should be used to distinguish different variables in the table, with those variables being used as X's in the X/Y plots.

Example: *PlotTbl(...,'LineWidthXVars',{'AvgHeight' 'AvgWeight'})*

In this example different line widths will be used to distinguish lines whose X values are AvgHeight versus lines whose X values are AvgWeight.

Name-value pair: *LineWidthYVars — cell array of variable names* (or equivalently)
Name-value pair: *LineWidthY — cell array of variable names*

This name-value pair indicates that different line widths should be used to distinguish different variables in the table, with those variables being used as Y's in the X/Y plots.

Example: *PlotTbl(...,'LineWidthYVars',{'AvgHeight' 'AvgWeight'})*

In this example different line widths will be used to distinguish lines whose Y values are AvgHeight versus lines whose Y values are AvgWeight.

Name-value pair: *LineWidthXYVars — cell array of pairs of variable names* (or equivalently)
Name-value pair: *LineWidthXY — cell array of pairs of variable names*

This name-value pair indicates that different line widths should be used to distinguish different combinations of variables in the table. Two variables are listed; the first is used as X's in the X/Y plots, and the second is used as Y's.

Example: *PlotTbl(...,'LineWidthXYVars',{'AvgHeight' 'AvgWeight' 'AvgWeight' 'AvgHeight'})*

In this example different line widths will be used to distinguish plots of AvgHeight as X and AvgWeight as Y, as compared with plots of AvgWeight as X and AvgHeight as Y.

Note: You should never specify *more than one* of the four options LineWidthCodeVar, LineWidthXVars, LineWidthYVars, and LineWidthXYVars. That is, line widths can only be distinguished in one of these ways—not several. If you want the plots to be distinguished in more than one way, use other design elements.

Name-value pair: *LineWidthSpecs — vector of line widths*

This name-value pair indicates that you want to replace PlotTbl's default set of line width specifications with a new set. Use MATLAB's standard line width specifications in whatever order you prefer. The line widths will be used in order by values of the CodeVar or the order of the variables listed for LineWidthXVars or LineWidthYVars.

Example: *PlotTbl(...,'LineWidthSpecs',[3 5 8 12])*

In this example the first line will have width 3, the next width 5, etc.

Name-value pair: *LineWidthOrder — 'stable' or 'sorted'*

This name-value pair is only applicable when used in conjunction with the 'LineWidthCodeVar' pair. It indicates whether you want PlotTbl to assign the different values of the indicated code variable to the different line widths in the order in which they appear in the table (i.e., 'stable', which is the default) or whether you want them assigned in sorted numerical order.

Example: *PlotTbl(...,'LineWidthOrder','sorted')*

In this example the values of the indicated code variable will be assigned to line widths in numerical order, regardless of the order in which they appear in the table.

Name-value pair: *LineWidthLegend — cell array of legend labels*

This name-value pair indicates that you want to replace PlotTbl's default labels of the line widths with the new labels indicated in your cell array. The order of the legend labels in the cell array should correspond to the order in which the different values of the indicated variable are assigned to the different line widths, as determined by the LineWidthOrder name-value pair.

Example: *PlotTbl(...,'LineWidthLegend',{'Male' 'Female'})*

In this example the legends for the first and second line widths will be "Male" and "Female", respectively, rather than the defaults of "Gender=1" and "Gender=2". Obviously, this option would make sense if used in combination with the name-value pair "'LineWidthCodeVar','Gender"'.

6.5 Name-value pairs for marker size control

Name-value pair: *MarkerSizeCodeVar — variable name* (or equivalently)
Name-value pair: *MarkerSize — variable name*

This name-value pair indicates that a different marker size should be used for each different value of the indicated table variable. The variable name is specified as a character vector, say sVar.

Example: *PlotTbl(...,'MarkerSizeCodeVar','Gender')*

In this example different marker sizes will be used to plot the data for males versus females (i.e., Gender==1 versus Gender==2).

Name-value pair: *MarkerSizeXVars — cell array of variable names* (or equivalently)
Name-value pair: *MarkerSizeX — cell array of variable names*

This name-value pair indicates that different marker sizes should be used to distinguish different variables in the table, with those variables being used as X's in the X/Y plots.

Example: *PlotTbl(...,'MarkerSizeXVars',{'AvgHeight' 'AvgWeight'})*

In this example different marker sizes will be used to distinguish lines whose X values are AvgHeight versus lines whose X values are AvgWeight.

Name-value pair: *MarkerSizeYVars — cell array of variable names* (or equivalently)
Name-value pair: *MarkerSizeY — cell array of variable names*

This name-value pair indicates that different marker sizes should be used to distinguish different variables in the table, with those variables being used as Y's in the X/Y plots.

Example: *PlotTbl(...,'MarkerSizeYVars',{'AvgHeight' 'AvgWeight'})*

In this example different marker sizes will be used to distinguish lines whose Y values are AvgHeight versus lines whose Y values are AvgWeight.

Name-value pair: *MarkerSizeXYVars — cell array of pairs of variable names* (or equivalently)
Name-value pair: *MarkerSizeXY — cell array of pairs of variable names*

This name-value pair indicates that different marker sizes should be used to distinguish different combinations of variables in the table. Two variables are listed; the first is used as X's in the X/Y plots, and the second is used as Y's.

Example: *PlotTbl(...,'MarkerSizeXYVars',{'AvgHeight' 'AvgWeight' 'AvgWeight' 'AvgHeight'})*

In this example different marker sizes will be used to distinguish plots of AvgHeight as X and AvgWeight as Y, as compared with plots of AvgWeight as X and AvgHeight as Y.

Note: You should never specify *more than one* of the four options MarkerSizeCodeVar, MarkerSizeXVars, MarkerSizeYVars, and MarkerSizeXYVars. That is, marker sizes can only be distinguished in one of these ways—not several. If you want the plots to be distinguished in more than one way, use other design elements.

Name-value pair: *MarkerSizeSpecs — vector of marker sizes*

This name-value pair indicates that you want to replace PlotTbl's default set of marker size specifications with a new set. Use MATLAB's standard marker size specifications in whatever order you prefer. The marker sizes will be used in order by values of the CodeVar or the order of the variables listed for MarkerSizeXVars or MarkerSizeYVars.

Example: *PlotTbl(...,'MarkerSizeSpecs',[7 10])*

In this example the first marker will have size 7 and the second will have size 10.

Name-value pair: *MarkerSizeOrder — 'stable' or 'sorted'*

This name-value pair is only applicable when used in conjunction with the 'MarkerSizeCodeVar' pair. It indicates whether you want PlotTbl to assign the different values of the indicated code variable to the different marker sizes in the order in which they appear in the table (i.e., 'stable', which is the default) or whether you want them assigned in sorted numerical order.

Example: *PlotTbl(...,'MarkerSizeOrder','sorted')*

In this example the values of the indicated code variable will be assigned to marker sizes in numerical order, regardless of the order in which they appear in the table.

Name-value pair: *MarkerSizeLegend — cell array of legend labels*

This name-value pair indicates that you want to replace PlotTbl's default labels of the marker sizes with the new labels indicated in your cell array. The order of the legend labels in the cell array should correspond to the order in which the different values of the indicated variable are assigned to the different marker sizes, as determined by the MarkerSizeOrder name-value pair.

Example: *PlotTbl(...,'MarkerSizeLegend',{'Male' 'Female'})*

In this example the legends for the first and second marker sizes will be "Male" and "Female", respectively, rather than the defaults of "Gender=1" and "Gender=2". Obviously, this option would make sense if used in combination with the name-value pair "'MarkerSizeCodeVar','Gender"'.

6.6 Name-value pairs for subplot row and column control

Name-value pair: *SubplotRowsCodeVar — variable name*
Name-value pair: *SubplotColsCodeVar — variable name*

Each of these two name-value pairs indicates that a different subplot row or subplot column should be used for each different value of the indicated variable. The variable name is specified as a character vector, say sVar.

Example: *PlotTbl(...,'SubplotRowsCodeVar','Gender')*

This name-value pair indicates that the data for males and females should be plotted in two different rows of subplots. The figure might have just one column, or there might be several columns distinguished by the values of some other variable or by the variable plotted on the X or Y axis.

Name-value pair: *SubplotRowsXVars — cell array of variable names*
Name-value pair: *SubplotColsXVars — cell array of variable names*

Each of these two name-value pairs indicates that a different subplot row or column should be used for X/Y plots with each of the indicated variable names on the X axis.

Name-value pair: *SubplotRowsYVars — cell array of variable names*
Name-value pair: *SubplotColsYVars — cell array of variable names*

Each of these two name-value pairs indicates that a different subplot row or column should be used for X/Y plots with each of the indicated variable names on the Y axis.

Note: You should never specify *more than one* of the four options SubplotRowsCodeVar, SubplotRowsXVars, SubplotRowsYVars, and SubplotRowsXYVars, and never more than one of the four corresponding SubplotCols options. But you can specify both SubplotRows and SubplotCols in any combination that you want—that is, these "design elements" can be controlled independently.

Name-value pair: *SubplotRowsLegend — cell array of legend labels*
Name-value pair: *SubplotColsLegend — cell array of legend labels*

Each of these two name-value pairs indicates that you want to replace the default labels of these subplots with the new labels indicated in your cell array. (Strictly speaking, these are not legends, but rather titles at the tops of the subplots.)

Example: *PlotTbl(...,'SubplotRowsLegend',{'Male' 'Female'})*

In this example the first subplot will be labelled "Male" and the second will be labelled "Female", rather than the defaults of "Gender=1" and "Gender=2". This would make sense in combination with the option
Example: *PlotTbl(...,'SubplotRowsCodeVar','Gender')*

7 Name-value Pairs for Control of Axis Labels

Name-value pair: *XLabel — vector of subplot numbers*
Name-value pair: *YLabel — vector of subplot numbers*

Each of these two name-value pairs indicates that you want an X or Y axis label to appear on the subplot corresponding to one of the numbers in the vector. By default, all axes are labelled; this option is used when you want to turn off some of the labels. The subplots are numbered using the standard numbering scheme for MATLAB's subplot command.

Example: *PlotTbl(...,'XLabel',[7 8 9],'YLabel',[1 4 7])*

In this example, with a 3x3 arrangement of subplots, the X axes will be labelled for the plots in the bottom row (i.e., subplot numbers 7-9) and the Y axes will be labelled for the subplots in the left-most column (i.e., subplot numbers 1, 4, and 7).

Name-value pair: *XLabelStr — cell array of X labels for the subplots*
Name-value pair: *YLabelStr — cell array of Y labels for the subplots*

Each of these two name-value pairs allows you to specify the text of the X and Y axis labels for the subplots, which can be used to override the defaults (i.e., the X and Y variable names). This capability is needed when the desired axis label is not a legal MATLAB variable name. The length of the cell array is the number of subplots (i.e., you must specify one X or Y axis label for each subplot).

Example: *PlotTbl(...,'XLabelStr',{'\alpha' '\alpha' '\alpha' '\alpha'})*

In this example the X axes of all four subplots will be labelled α.

8 Name-value Pairs for Control of Legends

Name-value pair: *Legend — vector of subplot numbers*

This name-value pair provides a list of numbers of the subplots for which you want the legends to be displayed. By default, a legend is displayed only on the first subplot.

Example: *PlotTbl(...,'Legend',[1 3])*

In this example legends will be displayed on the first and third subplots.

Name-value pair: *LegendLoc — string*

This name-value pair can be used to override MATLAB's default choice of where to put the legend(s) on the plot(s). Use MATLAB's standard location specifications (e.g., 'Northeast', 'Best') to indicate where you want the legend(s).

Name-value pair: *LegendPos — vector of four position numbers*

Alternatively, this name-value pair can be used to override MATLAB's default choice of where to put the legend(s) on the plot(s). The option indicates that the legend(s) should be placed in the custom position indicated by the four numerical values in the vector (i.e., left bottom width height). Use MATLAB's standard position specification, determined by its units (e.g., see https://www.mathworks.com/matlabcentral/answers/80980-what-does-the-four-vector-mean-in-matlab-legend-position).

Note: To override the default legend positioning, you can specify LegendLoc or LegendPos, but not both.

Name-value pair: *LegendBox — 'boxon' or 'boxoff'*

This name-value pair indicates whether you want the box on the legend(s). I think boxes just add clutter, so the default is 'boxoff'.

9 Name-value Pairs for Adding Reference Lines

Name-value pair: *AddXRef — real number*

This name-value pair adds a vertical reference line at the indicated X value; for example, you might want to add a reference line at X=0 or at X=the average X value.

By default this reference line is black and dotted, but you can change these characteristics with the following:

Name-value pair: *XRefStyle — character string*

The character string of this pair is used as a MATLAB line descriptor such as 'k:' for a black dotted line (see appendices for lists of MATLAB's line types and colors).

Name-value pair: *AddYRef — real number*

Name-value pair: *YRefStyle — character string*

These are analogous to AddXRef and XRefStyle to add a horizontal line at the indicated Y value.

Name-only option: *AddDiagonal* —

This option has no argument, but merely indicates that a diagonal line should be drawn to mark X=Y. The following pair controls the style of this line, analogous to XRefStyle and YRefStyle.

Name-value pair: *DiagonalStyle — character string*

10 SubplotReshape

It is sometimes desirable to override PlotTbl's default configuration of the subplots within a figure. For example, suppose you want to display subplots showing 9 different Y variables. If you simply use SubplotRowsYVars or SubplotColsYVars, you will get a 9x1 or 1x9 configuration of plots, which does not look good. A 3x3 layout would look much better, and you can request that with the SubplotReshape name-value pair.

Name-value pair: *SubplotReshape — [nrows ncols]*

Example: *PlotTbl(...,'SubplotReshape',[3 3])*

In this example, PlotTbl will arrange the subplots in a configuration with 3 rows and 3 cols. (Note that a 3x3 layout could also be used even if there were only 8 different Y variables; in that case, there would just be an empty cell in the matrix of subplots.)

11 Customizing Further

MATLAB's plot command recognizes many "extra" options that are not explicitly handled by PlotTbl. You can use the PassThru command to set these extra options:

Name-value pair: *PassThru — cell array of parameters to pass to plot*

Example: *PlotTbl(...,'PassThru','MarkerIndices',[1 5 10])*

In this example, PlotTbl will display markers for the 1st, 5th, and 10th data points.

For further control over the plots, you can pass a function for PlotTbl to call after each subplot is plotted.

This function is called with the subplot row and column numbers as arguments, so you can do different things for different subplots if you want.

Name-value pair: *Customize — YourFunctionName*

For example, you could define a function to set the horizontal and vertical axes to the ranges of (0,1):

Example: *MyCustomFn = @(x,y)(axis([0 1 0 1])); % Define this custom function before calling PlotTbl*

Then you would tell PlotTbl to call it after each subplot like this:

Example: *PlotTbl(...,'Customize',MyCustomFn)*

Alternatively, the parameter following 'Customize' can be a cell array of function handles if you want to call several functions to customize each graph. For example, here are three functions to set the axis ranges and to specify the tic values and for the X axis:

Example: *MyCustomAxFn = @(x,y)(axis([0.5 2.5 60 100]));* Example: *MyCustomXTicFn = @(x,y)(xticks([1 2]));* Example: *MyCustomXTicLblFn = @(x,y)(xticklabels('far', 'near'));*

Then you would tell PlotTbl to call these three functions after each subplot by including them in a cell array, like this:

Example: *PlotTbl(...,'Customize',MyCustomAxFn,MyCustomXTicFn,MyCustomXTicLblFn,)*

Finally, even more fine-grained control can be achieved by calling the function SubplotTbl directly. SubplotTbl recognizes nearly all of the same formatting commands as PlotTbl except for SubplotRows and SubplotCols, which you have to manage yourself. See Demo.m for an example.

12 Tips

- PlotTbl plots the data from all of the rows in whatever table is passed to it. If you just want to plot some of the rows of a given table, just pass the desired row subset using MATLAB's standard row-selection methods.

  Example: *PlotTbl(AHWDat(AHWDat.Gender==2,:),'Age','AvgWeight','LineTypeCodeVar','Nationality');*

  In this example, data are just plotted for Gender 2.

- When you select out certain rows as suggested in the preceding tip, remember that this can change the order in which various CodeVar values appear in the table. For example, if you created a new table by eliminating rows with small values of height, the first gender to appear in the new table might be female, even though the males appeared first in the original table. This can create problems when you use the 'stable' Order and specify your own legends with a command like 'LineTypeLegend',{'Male' 'Female'}. If the females appear first in the new table (after selection), then the male and female legend labels will be reversed—not good!

  A useful trick to avoid this problem is to reset the values that you do not want plotted to NaN. All rows then remain in the same so the orders of CodeVars are not perturbed, but MATLAB does not plot the NaN values so they are omitted, as you want.

- For any design element that does *not* vary across your plots, PlotTbl uses the first of the default "Specs" options for that design element. If you would prefer something else, you can change this just by setting the Specs for that design element, even though you are not using it to distinguish between lines. You can include the Specs option for any design element, whether that element appears anywhere else in the PlotTbl command or not.

  Example: *PlotTbl(...,'MarkerTypeSpecs',' ',...);*

  In this example, all lines will be drawn without markers (as indicated by the blank marker type specification). By default, PlotTbl's first MarkerTypeSpec is the square, so the lines would be drawn with square markers if this option were not specified.

- After plotting, it is possible to change the titles shown on the subplots to any titles that you want. See Demo.m for an example.

- PlotTbl repeats specifications if necessary. For example, MATLAB defines 4 line types (see Appendix). If you want to distinguish more than 4 categories via line types, PlotTbl will recycle through the same ones. This is not ideal, but perhaps better than bombing.

- MATLAB leaves a lot of space between subplots—often, too much for my taste. The function 'subplotResize' can be used to stretch the subplots, horizontally and vertically, into the space that MATLAB would normally leave between them.

- The function CellArModify often provides a convenient way to modify the properties of all subplots. See Demo.m for an example.

A Appendix: Complete List of Name-value Pairs

AddDiagonal — (takes no parameter)
    DiagonalStyle — string line descriptor; default is 'k:'
AddXRef — real number at which to add a vertical reference line
    XRefStyle — string line descriptor; default is 'k:'
AddYRef — real number at which to add a horizontal reference line

YRefStyle — string line descriptor; default is 'k:'
ColorCodeVar — variable name
ColorLegend — cell array of legend labels
ColorOrder — 'stable' or 'sorted'
ColorSpecs — string list of colors, or cell array of RGB triples
ColorXVars — cell array of variable names
ColorYVars — cell array of variable names
Legend — vector of subplot numbers where you want the legend
LegendBox — 'boxon' or 'boxoff'
LegendLoc — string such as 'Northeast'
LegendPos — [left bottom width height]
LineTypeCodeVar — variable name
LineTypeLegend — cell array of legend labels
LineTypeOrder — 'stable' or 'sorted'
LineTypeSpecs — cell array of line type specifications
LineTypeXVars — cell array of variable names
LineTypeYVars — cell array of variable names
LineWidthCodeVar — variable name
LineWidthLegend — cell array of legend labels
LineWidthOrder — 'stable' or 'sorted'
LineWidthSpecs — vector of line widths
LineWidthXVars — cell array of variable names
LineWidthYVars — cell array of variable names
MarkerSizeCodeVar — variable name
MarkerSizeLegend — cell array of legend labels
MarkerSizeOrder — 'stable' or 'sorted'
MarkerSizeSpecs — vector of marker sizes
MarkerSizeXVars — cell array of variable names
MarkerSizeYVars — cell array of variable names
MarkerTypeCodeVar — variable name
MarkerTypeLegend — cell array of legend labels
MarkerTypeOrder — 'stable' or 'sorted'
MarkerTypeSpecs — string list of marker type specifications
MarkerTypeXVars — cell array of variable names
MarkerTypeYVars — cell array of variable names
SubplotColsCodeVar — variable name
SubplotColsLegend — cell array of legend labels
SubplotColsOrder — 'stable' or 'sorted'
SubplotColsXVars — cell array of variable names
SubplotColsYVars — cell array of variable names
SubplotReshape — [nrows ncols]
SubplotRowsCodeVar — variable name
SubplotRowsLegend — cell array of legend labels
SubplotRowsOrder — 'stable' or 'sorted'
SubplotRowsXVars — cell array of variable names
SubplotRowsYVars — cell array of variable names
XLabel — vector of subplot numbers
XLabelStr — cell array of X labels for the subplots
YLabel — vector of subplot numbers
YLabelStr — cell array of Y labels for the subplots

Note: Name-value pairs of the form "???CodeVar" can be abbreviated as just "???", and those of the form "???XVars" or "???YVars" can be abbreviated as "???X" or "???Y".

B Appendix: Line Types in MATLAB

In PlotTbl's default order:

- '-': Solid line
- '--': Dashed line
- ':': Dotted line
- '-.': Dash-dot line

Use 'none' or ' ' to omit the line.

C Appendix: Marker Types in MATLAB

In PlotTbl's default order:

- 'square' or 's': Square
- 'o': Circle
- 'diamond' or 'd': Diamond
- '^': Upward-pointing triangle
- 'v': Downward-pointing triangle
- '+': Plus sign
- 'x': Cross
- '*': Asterisk
- '.': Point
- '>': Right-pointing triangle
- '<': Left-pointing triangle
- 'pentagram' or 'p': Five-pointed star (pentagram)
- 'hexagram' or 'h': Six-pointed star (hexagram)

Use 'none' or ' ' to omit the marker.

D Appendix: Colors in MATLAB

In PlotTbl's default order:

- k: Black
- r: Red
- g: Green
- b: Blue
- c: Cyan
- m: Magenta
- y: Yellow
- w: White