Chapter 21: Printings and graphs

Once a regression has been performed, it can be useful to recall corresponding results. How to do it is explained in part 1 of this chapter. Part 2 then explains succinctly how to make graphs with Scilab. Part 3 presents the tools developed in GROCER to draw graphs, in particular to deal with timeseries. Part 4 gives some hints on how to use the "handle" representation of Scilab new graphic mode. Part 5 presents the functions and their specifications.

1. Printing results

Except for a very few functions (bkw), every printing of econometric results involves one or another subroutine devoted to such a printing. So, a user can easily print these results later, provided they have been saved into a corresponding tlist. Although there a several printings subroutines, the user can use a "one size fits all" function, called prtres. If however you want to modify the corresponding subroutines, you have to know that they are the followings:

- prtuniv: ols, olst, olsc, olsar1, nls, iv, logit, probit, tobit, Theil-Goldberger, hwhite, lad, ridge, robust, pfixed, prandom, pbetween, ppooled
- prtsys: sur, twosls, threesls
- prtvar: var, bvar, ecm, becm
- prtvarma: varma
- prtunitr: adf, ers, kpss, phil_perr, schmphi,
- prtchi: archtest, arlm, chowtest, doornhans, jbnorm, predfail, predfailin, reset, white
- prttvp: tvp
- prtauto: auto
- prtjohan: johansen
- prtdiebmar: diebmar
- prtbrybos: brybos
- prtms: ms_reg, ms_mean, ms_var
- prtbma_g: bma_g
- prthausmann: phausmann
- pltspectal: spectral, specvarma
- prtdisag: chowlin, fernandez, litterman

As an example, take again Hendry and Ericsson estimate through the following commands:

```
-->load('SCI/macros/grocer/db/bdhenderic.dat') ;
```
// load the database…

```
-->bounds('1964q3','1989q2')
```
// set the bounds

```
-->rols=ols('del(lm1-lp)','del(lp)','del(lagts(1,lm1-lp-ly))','rnet',
'lagts(1,lm1-lp-ly)','cte','noprint');
```
// perform the ols estimation, without printing the results, but with saving them in the results tlist

// rols

```
-->a=4
 a  =

    4.
```

// do another operation…

```
-->prtres(rols);

ols estimation results for dependent variable: del(lm1-lp)
estimation period: 1964q3-1989q2
number of observations: 100
number of variables: 5
R² =  .7616185      ajusted R² = .7515814
Overall F test: F(4,95) = 75.880204       p-value = 0
standard error of the regression:  .0131293
sum of squared residuals:  .0163761
DW(0) =2.1774376
Belsley, Kuh, Welsch Condition index: 114

variable                coeff      t-statistic p value
del(lp)                  -.6870384 -5.4783422  3.509E-07
del(lagts(1,lm1-lp-ly))  -.1746071 -3.0101342   .0033444
rnet                     -.6296264 -10.46405   0
lagts(1,lm1-lp-ly)       -.0928556 -10.873398  0
cte                       .0234367  5.818553    7.987E-08


                          *
                      *     *
```

// you should have alternatively typed: prtuniv(rols)

Note that all these functions have an optional second argument (the third as regards function prtauto, see below), which is the logical name of the output file. When this argument is not given (as it is the case in the example above and in the estimation functions), then the output file is supposed to be the screen (%io(2) in Scilab terminology). You can however direct your results toward another file, provided it has been opened before.

For instance, if you want to save the above results on file he.txt, then you could type (see Scilab reference manual for more details on function file):

```
-->foutput=file('open','he.txt','unknown');

-->prtres(rols,foutput);

-->file('close',foutput);
```

All these printing functions use a subroutine called printmat (based on lines of code written by INRIA), whose purpose is to print a string matrix on screen with all columns vertically aligned. If you want to improve the presentation of GROCER econometric results, printmat is the function to modify.

All printing functions are built upon the same model as prtuniv, except one: prtauto. The output file is indeed only the third argument of prtauto and the second one is either an empty string or a string beginning with 'prt=', followed by the list of options separated by commas (this is exactly the same way printings options are entered in function automatic, see chapter 13 for more details on this). Note also that you can alternatively use the subroutines prtauto_univ if you want to print the results of one specific regression performed by automatic (initial, stage 0, stage 1 union or final model) and prtauto_multi if you want to print the -possibly- numerous models generated by stages one or two. In that case, you should use prtauto_univ in the following way: prtauto_univ(result,namemod,test,output) where namemod is one of the possible models ('initial model', 'stage 0 model', 'stage 1 union model', 'stage 2 union model' or 'final model') and test is a boolean. Similarly, you can use prtauto_multi in the following way: prtauto_univ(result,namemod,test,output) where namemod is either 'stage 1 models' or 'stage 2 models', test has the same meaning as in prtauto_univ, and path is a boolean equal to %t if you want to display the path (that is the order of elimination of the variables) and % if not.

2. Scilab graphs.

Scilab contains two graphic modes: on old one, inherited from past versions and a new one, developed for the 2.7 version. You can switch from a mode to another one with the following instructions: set("old_style","on") and set("old_style","off"). (°) Grocer versions from the 1.2 one work under the new graphic mode. For users used to work with the old one, this is –almost– transparent: the only difference is that now, the y scale is not adjusted to the series, but begins and ends with rounded numbers (which is the standard way to draw the y scale of a graph); the user who wants to have the y scale drawn the old way has now an option to enter to pltseries, namely 'just_scale=%T' (this feature can be useful if you want to see the relation between 2 series). (°°)

2.1 Graphic Windows

Scilab graphs are drawn on graphic windows. By default, Scilab draws a graph on a window numbered "0". You can however choose any integer to draw your graphs by using Scilab function xset:

```
-->scf(1)
```

opens the window "numbered" one.

Every call to a graphic routine is executed on the currently opened graphic window, without clearing what has been previously executed: this is necessary to draw complex graphs, but it implies that, if you want to draw successively several graphs, you have to give the instruction clf(i) (clf(1) for window number 1 for instance) to clear graphic window number i and erase all previous graphs.

xset has several other options, that allows the user to choose some parameters. These options are very numerous, so the reader is left to Scilab help for a whole description of these options. The most useful ones (according to me) are the following:

- xset('line style',i) which set the line style (used for the series and the axes) to i (see Scilab

function pltseries0 for an example);

- xset('thickness,i) which sets the thickness (in pixels) of the lines;
- xset('font size',i) to set the font size (in pixels) to i;
- xset('use color',0) or xset('use color',1) to draw black and white or color (default) graphs;
- xset('default') reload the default values.

2.2 2D graphs

Scilab allows you to draw 2D and 3D graphs. The interested reader should have a look at Scilab manual for precision about 3D graphs.

A 2D graph involves necessarily a real vector x, representing the values taken by the x axis, and a matrix y, with the same number of rows as x, representing the values of the series to draw, each series being represented by a column of the y matrix.

A simple graphic call is the then following

```
-->plot2d([1:12], [sqrt([1:12])' log([1:12])'])
```

This does not mean that you cannot represent series, such as timeseries, where the x axis is not a real vector, but rather that you have to use the intermediation of a vector. For instance, the same graph involving instead the twelve months can be done the following way:

```
-->plot2d([1:12],[sqrt([1:12])' log([1:12])'],axesflag=0);   drawaxis(x=1:12,y=0,
dir='d',val=['jan' 'feb' 'mar' 'apr' 'may' 'jun' 'jul' 'aug' 'sep' 'oct' 'nov'
'dec']) ; drawaxis(x=1,y=[0:0.4:3.6],dir='l')
```

The operating mode involves entering in plot2d the option axesflag=0, which means that no axis is drawn, and then using the function drawaxis, that draws an x or y axis. Note that you have to call the function also for the y axis, even if the default one drawn by plot2d was suitable to you. Drawaxis imposes to enter the axis type, which is done through option dir: dir=’d’ means an horizontal axis with tics going down from the axis; dir=’l’ means a vertical axis with tics going left from the axis; the other options are dir=’u’ for an means an horizontal axis with tics going up from the axis and dir=’l’ means a vertical axis with tics going right from the axis.

plot2d has many options, that can be used to customize your gaphs. A complete call to plot2d would involve the following command:

plot2d(x,y,style,'abc',leg,rect,nax)

where:

- x is the vector or values on the x axis;
- y is the matrix of values on the y axis;
- style is a vector of size equal to the number of series to draw and each co-ordinate of x represents the style (i.e the color for a color graph or the type of line for black and white graphs); if style is set to a negative number then series are represented by marks and not by lines;

- a, b and c are three integers; a is equal to 1 if the user wants a legend for the series, 0 if not; b is equal to 0 if the user wants to use the previous axis, 1 if she wants to enter the axes (and the axes parameters are then given by the values of argument rect, see below), 2 for an automatic scale, 3 and 4 for isometric scales; c=0 if nothing is drawn around the plot, c=1 if the user wants to draw axes, the y axis being displayed on the left; c=2 if the user wants the plot to be surrounded by a box without tics, c=3 if the user wants to draw axes, the y axis being displayed on the right; c=4 if she wants that the axes are drawn centered in the middle of the frame box, c=5 if she wants the axes to be drawn so as to cross at point (0,0) (if point (0,0) does not lie inside the frame, axes will not appear on the graph);
- leg is the legend, the name of each series must be separated by a @ (for instance 'toto@tutu@titi');
- rect is a (1x4) vector where the the first 2 co-ordinates are the minimum values for the x and the y axis respectively and the last 2 are the maximum values for the x and the y axis respectively; this in particular useful if you want to have a graph that is bigger than the amplitude of the series.
- nax is a (1x4) vector that defines the number of tics: the first 2 co-ordinates determines the number of subgraduations and graduations on the x axis and the last 2 ones the number of subgraduations and graduations on the y axis.

Note that there are functions that can perform some of these options after a call to plot2d: we have already seen the use of drawaxis, but there is also a function legends that draws legends (and which allows to set both the style line and the style color, see scilab function pltseries0 for an example).

There is also a function xtitle that add a title to the graph. The syntax is very simple: xtitle(title) where title is a string. You can also add titles to the x and y axes: type xtitle(title,xtitle,ytitle). The font of the title can be set before by using the command xset('font title',i)

You can draw several graphs on the same graphic window. This is done by using the command xsetech that sets a part of the window where the graph will be drawn. xsetech(a,b,c,d) defines the part of the graphic widow that will be used, where a and c (respectively b and d) are integers between 0 and 1 and define the upper part of the graph and its height (respectively the left part of the graph and its length); for instance, xsetech(0,0,0.5,0.5) draws a graph in the upper and left quarter of the graphic window. In grocer function plt_irf you can find the following use of xsetech: xsetech([0 (j-1)/N 1 1/N]), that makes use of 1/N height of the graphic window.

(°) 2 useful instructions deserve lastly attention: drawlater() and drawnow(). Drawlater() commands to wait before drawing the graphs on the window (successive –time consuming- calls to the graphic window are therefore avoided), and drawnow() commands the drawing of the graph. (°°)

3. Grocer graphs.

Grocer proposes a 2D graphic function called pltseries, which is an extended version of plot2d, that can in particular deal with timeseries. The extensions of plot2d are the following:

- as with plot2d, you can have an x axis which is a vector, but, unlike plot2d, you can also draw timeseries, with an x axis equal to the date span of the timeseries;
- you can draw series or timeseries referring to a left and a right axis; in that case you have to enter as on option in pltseries 'yaxis=vec' where vec is a vector of 1 and 2, whose co-ordinate i refers to the axis pertaining to series i (1 for the left axis and 2 for the right one);
- you can mix curves and bars on the same graph;
- each series is represented by a curve with different color *and* different line style; in plot2d, either you have different colors, but line styles are then the same for all series, or you use the black and white option (by setting xset("use color",0)) and line styles are different, but all curves are black; it is not convenient if you want that your graphs can be printed by a black as well as a white color writer, especially when copied in other software (such as Word); this is why pltseries allows your series to be drawn with different colors and styles; you can use the defaults colors and styles or choose them by entering the options 'color=vec' (respectively 'style=vec') where vec is a vector of integers corresponding to the chosen colors; (°) you can also choose the thickness of these lines with the option 'thickn=vec';(°°)
- you can choose the y co-ordinate of the x axis, by entering 'yaxex=x' where c is the chosen co-ordinate; you can also choose the x co-ordinate of the y axis or axes, by entering 'x0(1)=xx' (and 'x0(2)=xx' for the second y axis);
- (°) you can choose to adjust the size of the scale to the amplitude of the series with the option 'just_scale=%f';
- when using timeseries, you can choose the time span over which to draw the timeseries by the option 'bounds=[''bounds1'';''bounds2'']' where bounds1 and bounds2 are two acceptable bounds. (°°)

A call to <u>pltseries</u> takes the following form: pltseries(serie1,…,serien,option1,…, optionk), where:

- serie1 until serien are either vectors or timeseries, between quotes or not (or any object that can be entered in <u>explox</u>, such as matrix or lists of ts or vectors).
- option1 until optionk is one of the following options:

  - 'title=x' if the user wants to give its own title (default: the name of the ts or vectors if they have been given between quotes, the string 'ts' if not); see chapter 16 (p.2) for an application;
  - 'bounds=["b1" ; "b2" ;...; "bn"]' if the user wants to give its own bounds (default: the whole series); this option allows to draw ts even if some of them do not cover the whole time span specified by the bounds; if the option is not given, then the series are plotted only on the greatest time span over which they are altogether not NA; see chapter 16 (p.11) for an application;
  - 'y0=xx' if the user wants to put the x axis at value xx (default: y minimum value)
  - 'yaxis=xx' where xx is a (1xp) matrix of 1 and 2, if the user wants 2 axes, respectively at the left and the right of the graph; the j the series is represented on the left axis if xx(j)==1 and on the right one if xx(j)==2 (default: only a left axis); see chapter 2 (p.10) for an application;
  - 'x=xx' where xx is the (1xnobs) string vector to put on the x axis

- ‘styleg =xx’ with xx an integer row vector of size p representing the style of each piece of the legend; see chapter 16 (p.11) for an application;
- ‘color =xx’ with xx integer row vector of size p representing the line color of each series
- ‘style =xx’ with xx integer row vector of size p representing the line style of each series
- (°) ‘thickn=xx’ with xx integer row vector of size p representing the line thickness of each series (°°)
- ‘leg =xx’ with xx title of the legend;
- ‘bars=v’ where v is a vector whose co-ordinate i indicates whether series i must be drawn as a bar (if co-ordinate i is equal to1) or a curve (if co-ordinate i is different from1); see chapter 15 (p.2) for an application;
- 'x0(1)=xx' with xx integer representing the x location of the first y axis (default: put at x=1);
- 'x0(2)=xx' with xx integer representing the x location of the second y axis (default: put at x=nobs);
- 'window=x' if the user wants to specify the # x where the graph is plotted (default: the window 1)

Pltseries uses subroutines that can be useful for other types of graphs. First is the function pltseries0: this function is less flexible that pltseries, but makes the same kind of graphs as pltseries. This function takes as inputs a y matrix; the y value where to to draw the x axis (or [] if this value is to be set to the minimum of y values); the title, the x scale and the number of the graphic window. There are also optional arguments, such as 'styleg=', to set the legend style, that is the location where the legend is put, or 'yaxis=...' if the user wants to draw 2 axes... (see the options in part 3 of this chapter).

This function can then be used to draw graphs involving complex x axes. For instance, the same graph as the one drawn in part 2.2 can be drawn more simply by calling:

```
-->pltseries0([sqrt([1:12])' log([1:12])'],0,'',['jan' 'feb' 'mar' 'apr' 'may' 'jun' 'jul' 'aug' 'sep' 'oct' 'nov' 'dec'],1)
```

Pltseries0 itself uses several subroutines. Drawx draws an x axis that is readable, that is where the characters are not mixed one with another on the screen. The arguments to drawx are: the axis legend, that must be a string vector, whose size is equal to the size of the x vector; ref_nbinter, which is equal to the maximum number of chars that can be drawn; font_axis, the font of these chars, y0 which is the value of the x axis where the y axis is drawn and x0 which is the value where the x axis begins (optional, default is set to 1).

Drawy is drawx counterpart for the y axis. It takes as input the y matrix; the axis font; the y0 value if the user wants to draw the x axis at a predefined value or [] if the user wants that drawy returns the minimum y value where the x axis can then be drawn.

The general parameters that govern these drawings and makes them readable are governed by function param_g: default parameters, as defined in scilab database provided with grocer, are set for a xxx inches screen. They must be adapted for other screen sizes by this function param_g.

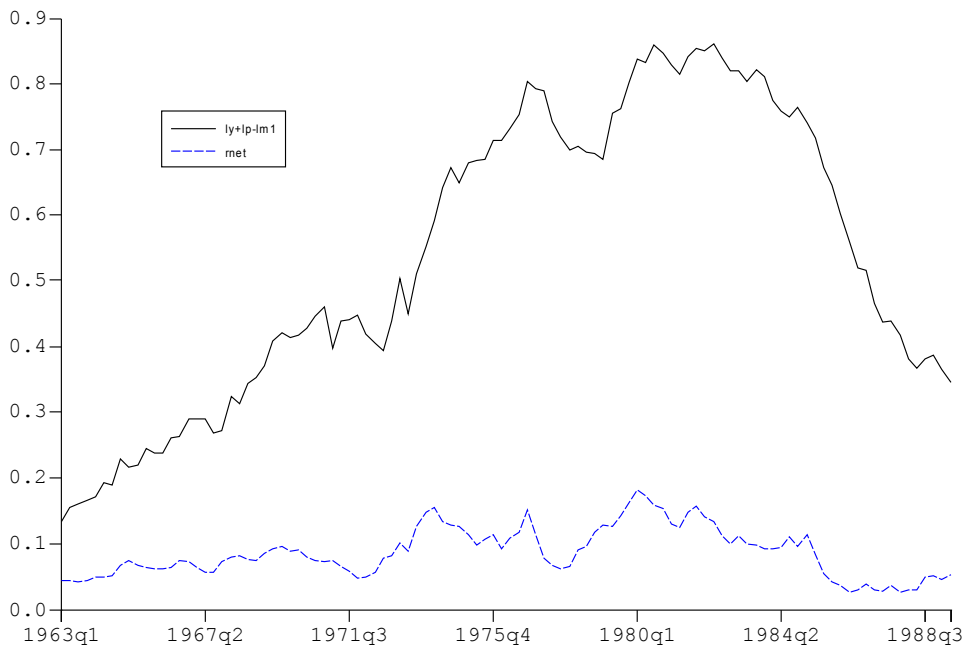
Lastly, there exists functions that graph results from some econometric programs. These are: plt_dbf and plt_dff that takes as input a results tlist from dfbeta and graphs the dfbetas, the dffits, studentized residuals and max hats; pltbrybos, that takes a results tlist from brybos and graphs the series, its peaks and trhoughs and the "recession" periods; pltcusum, that takes a results tlist from cusumb or cusumf and graphs the results of a cusum test; pltirf, that takes as input a results tlist from irf and graphs the impulse function from a var, along with their confidence bands; plttvp, that takes as input a results tlist from tvp and graphs the time varying coefficients; pltuniv, that takes as input a results tlist from any univariate function draws the and graphs either the residuals, or the observed and fitted values, or both; pltacf that takes as input a results tlist from acf or pacf and graphs the corresponding (partial) autocorrelations.

(°) As an example, load once again Hendry and Ericsson database:

```
-->load('SCI/macros/grocer/db/bdhenderic.dat')
```

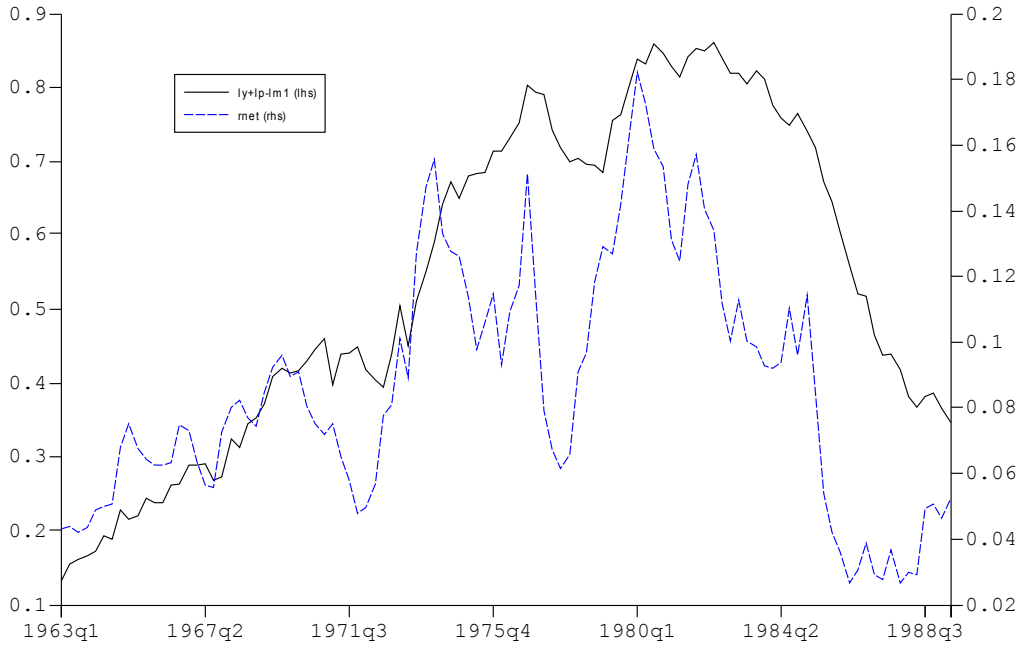
And graph the log of the inverse of the money velocity and of the interest rate:

```
-->pltseries('delts(lm1)','delts(ly+lp)')
```



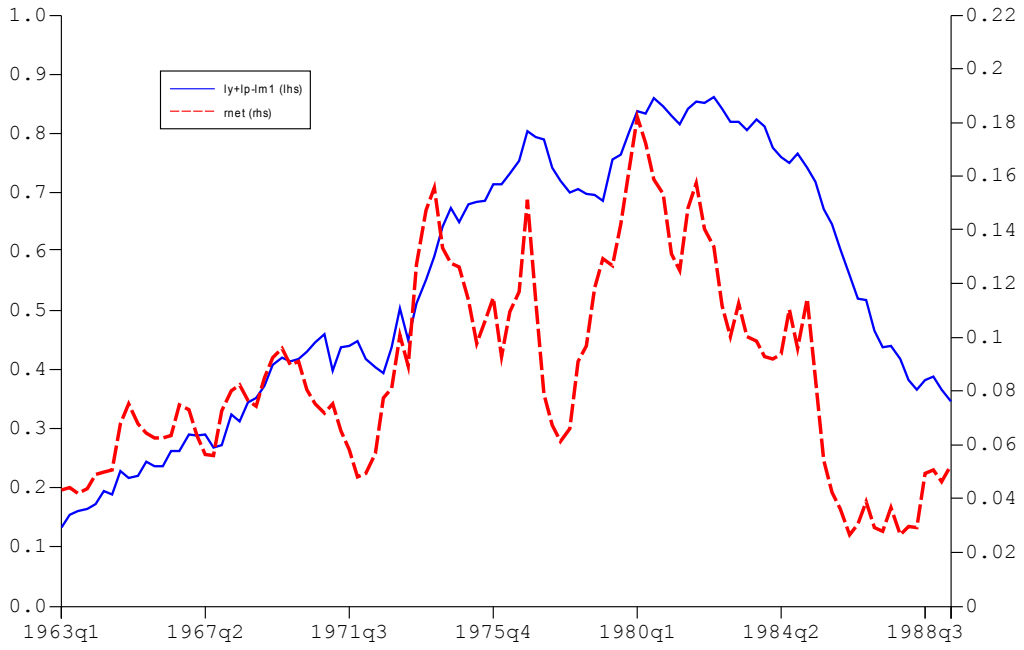
To see better the relationship between the 2 series, it is better to use 2 different axes:

```
-->pltseries('ly+lp-lm1','rnet','yaxis=[1 2]')
```



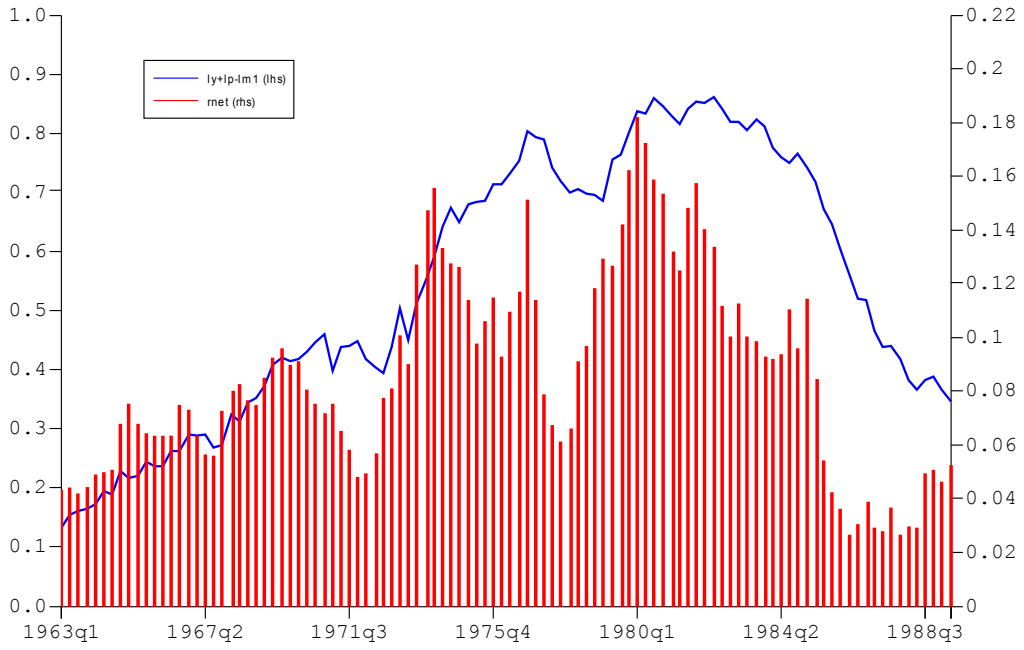
Now, change the thickness and the colors of the curves:

```
-->pltseries('ly+lp-lm1','rnet','yaxis=[1 2]','thickn=[2 3]','color=[2 5]')
```



Now, draw the second curve with bars:

```
-->pltseries('ly+lp-lm1','rnet','yaxis=[1        2]','thickn=[2        3]','color=[2
5]','bars=[0 1]')
```



Note that the second axis has been adjusted to begin at 0. Note also that the bars are plain: in Scilab, bars cannot be discontinuous and have a thickness greater than 1.

4. A few remarks about Scilab new graphic mode.

The new graphic mode allows more flexibility to the user (and this flexibility has been used for grocer graphs of the version 1.2), but this remains rather tedious. So I will give in this part only a few hints at how it works (Scilab step by step, available at http://h0.web.u-psud.fr/orscilab/index.html, can very usefully complement this part, but still remains incomplete).

Scilab works with handles, which gathers the graph properties. To see how it can work, redraw the first graph of part 3:

```
-->pltseries('delts(lm1)','delts(ly+lp)')
```

The handle of this graph can be recovered with the instruction:

```
-->h1=gcf()
```

Which results in:

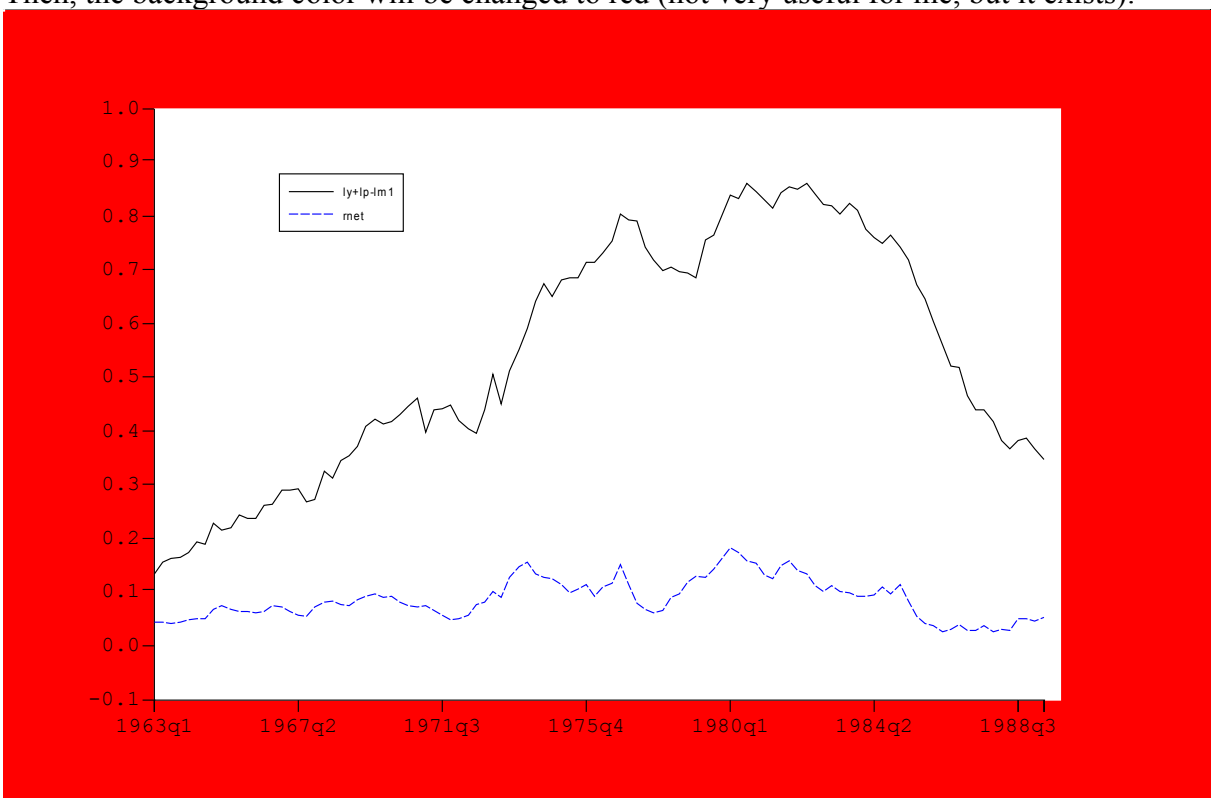
```
 h1  =

Handle of type "Figure" with properties:
========================================
```

```
children: ["Axes";"Axes"]
figure_style = "new"
figure_position = [-4,-4]
figure_size = [1024,668]
axes_size = [1024,668]
auto_resize = "on"
figure_name = "Scilab Graphic (%d)"
figure_id = 1
color_map= matrix 32x3
pixmap = "off"
pixel_drawing_mode = "copy"
immediate_drawing = "on"
background =  -2
visible = "on"
rotation_style = "unary"
user_data = []
```

Now that the handle has been recovered, you can change its properties. For instance, if you enter:

```
-->h1.background=5
```

Then, the background color will be changed to red (not very useful for me, but it exists):



I now go back to the standard background:

```
-->h1.background=-2
```

Most of the graph properties can however be found in the handle “children”

The command:

```
-->h2=h1.children(1)
```

will allow one to recover the first of these children:

```
h2  =

Handle of type "Axes" with properties:
======================================
parent: Figure
children: "Compound"

visible = "on"
axes_visible = ["off","off","off"]
axes_reverse = ["off","off","off"]
grid = [-1,-1]
x_location = "bottom"
y_location = "left"
title: "Label"
x_label: "Label"
y_label: "Label"
z_label: "Label"
auto_ticks = ["on","on","on"]
x_ticks.locations = matrix 11x1
y_ticks.locations = matrix 11x1
z_ticks.locations = [-1;0;1]
x_ticks.labels = matrix 11x1
y_ticks.labels = matrix 11x1
z_ticks.labels = ["-1";"0";"1"]
box = "off"
sub_ticks = [1,1]
font_style = 6
font_size = 1
font_color = -1

isoview = "off"
cube_scaling = "off"
view = "2d"
rotation_angles = [0,270]
log_flags = "nnn"
tight_limits = "off"
data_bounds = [0,0;0.001,0.001]
zoom_box = []
margins = [0,0,0,0]
axes_bounds = [0,0,0.001,0.001]

auto_clear = "off"
auto_scale = "on"

hiddencolor = 4
line_mode = "on"
line_style = 0
thickness = 1
mark_mode = "off"
mark_style = 0
mark_size_unit = "tabulated"
```

```
mark_size = 0
mark_foreground = -1
mark_background = -2
foreground = -1
background = -2
clip_state = "off"
clip_box = []
user_data = []
```

And the following:

```
-->h3=h1.children(2)
```

Allows to recover the second of these children:

```
 h3  =

Handle of type "Axes" with properties:
======================================
parent: Figure
children: ["Compound";"Compound";"Axis";"Axis";"Axis";"Axis";"Axis"]

visible = "on"
axes_visible = ["off","off","off"]
axes_reverse = ["off","off","off"]
grid = [-1,-1]
x_location = "bottom"
y_location = "left"
title: "Label"
x_label: "Label"
y_label: "Label"
z_label: "Label"
auto_ticks = ["on","on","on"]
x_ticks.locations = [0;20;40;60;80;100;120]
y_ticks.locations = [-0.2;0;0.2;0.4;0.6;0.8;1]
z_ticks.locations = [-1;0;1]
x_ticks.labels = ["0";"20";"40";"60";"80";"100";"120"]
y_ticks.labels = ["-0.2";"0.0";"0.2";"0.4";"0.6";"0.8";"1.0"]
z_ticks.labels = ["-1";"0";"1"]
box = "off"
sub_ticks = [3,3]
font_style = 6
font_size = 3
font_color = -1

isoview = "off"
cube_scaling = "off"
view = "2d"
rotation_angles = [0,270]
log_flags = "nnn"
tight_limits = "on"
data_bounds = [1,-0.1;108,1]
zoom_box = []
margins = [0.125,0.125,0.125,0.125]
axes_bounds = [0,0,1,1]
```

```
auto_clear = "off"
auto_scale = "on"

hiddencolor = 4
line_mode = "on"
line_style = 1
thickness = 1
mark_mode = "off"
mark_style = 0
mark_size_unit = "tabulated"
mark_size = 0
mark_foreground = -1
mark_background = -2
foreground = -1
background = -2
clip_state = "clipgrf"
clip_box = []
user_data = []
```

Note that this handle has again 7 children. The first one is a “compound” which relates the different elements relative to the last series. The second one is a “compound” which relates the different elements relative to the first series (note that they are in reverse order).

So :

```
-->h4=h3.children(1)
```

provides a another compound with a new child(ren), which is a “polyline” and which represents therefore a curve. It can be again recovered:

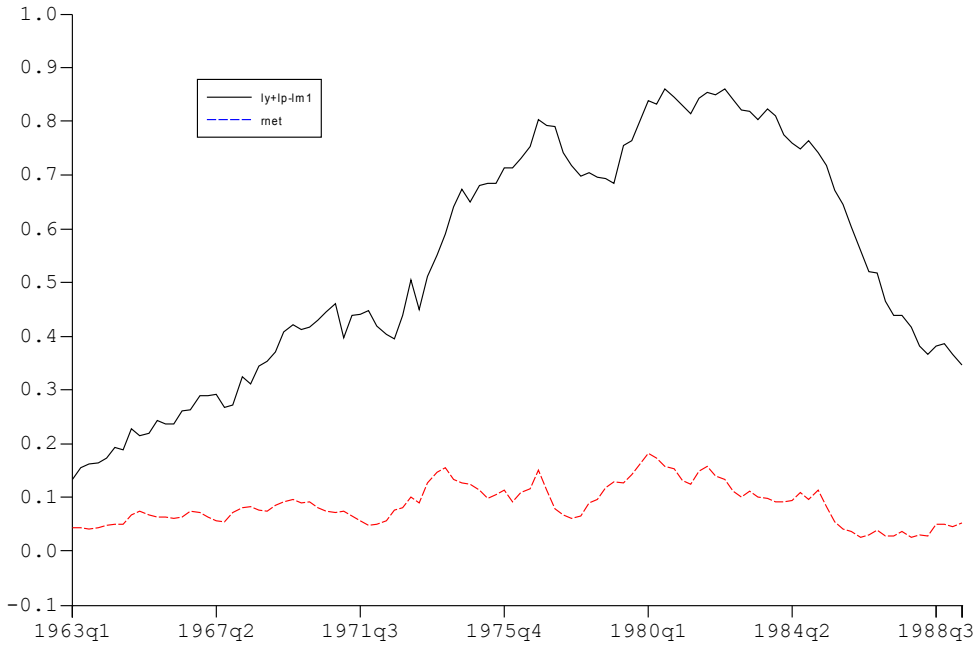
```
 h4  =

Handle of type "Compound" with properties:
==========================================
parent: Axes
children: "Polyline"
visible = "on"
user_data = []

-->h5=h4.children(1)
 h5  =

Handle of type "Polyline" with properties:
==========================================
parent: Compound
children: []
visible = "on"
data = matrix 106x2
closed = "off"
line_mode = "on"
fill_mode = "off"
line_style = 2
thickness = 1
arrow_size_factor = 1
```

```
polyline_style = 1
foreground = 5
background = -2
interp_color_vector = []
interp_color_mode = "off"
mark_mode = "off"
mark_style = 0
mark_size_unit = "tabulated"
mark_size = 0
mark_foreground = -1
mark_background = -2
x_shift = []
y_shift = []
z_shift = []
bar_width = 0
clip_state = "clipgrf"
clip_box = []
user_data = []
```

You can now change the curve color with the instruction:

```
-->h5.foreground=5
```

Which changes the curve color to red:



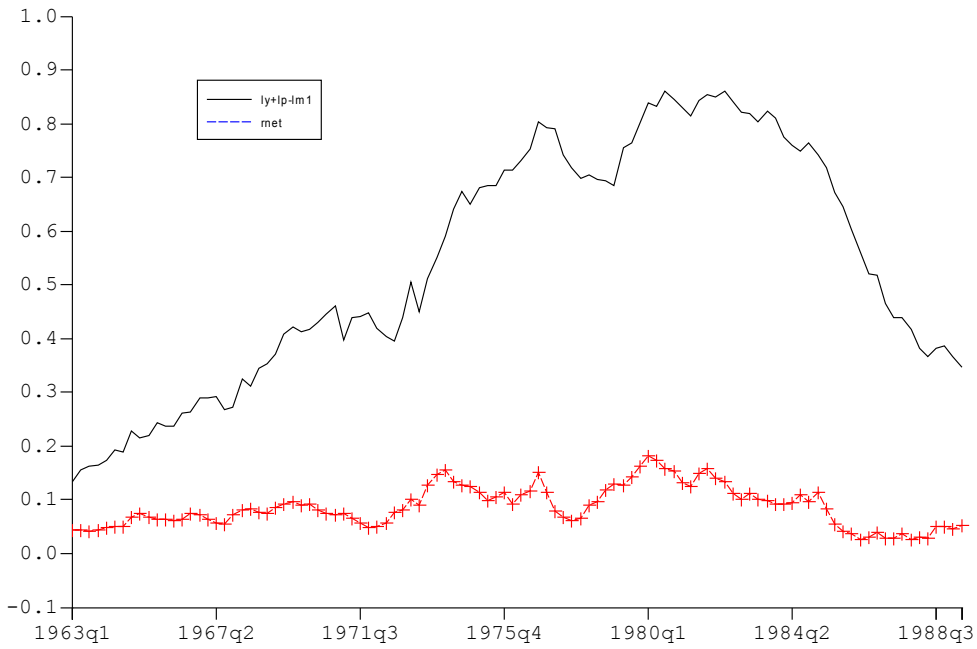
Note that you can have obtained the same result much more simply, with the command:

```
-->pltseries('ly+lp-lm1','rnet','color=[1 5]')
```

which should have also changed the color of the legend!

You can at this stage also however add red marks to the curve with the instructions:
```
-->h5.mark_style=1;h5.mark_foreground=5
```

With the following result:



Now, to change the legend, you should have used the following instructions:
```
-->h6=h2.children
```

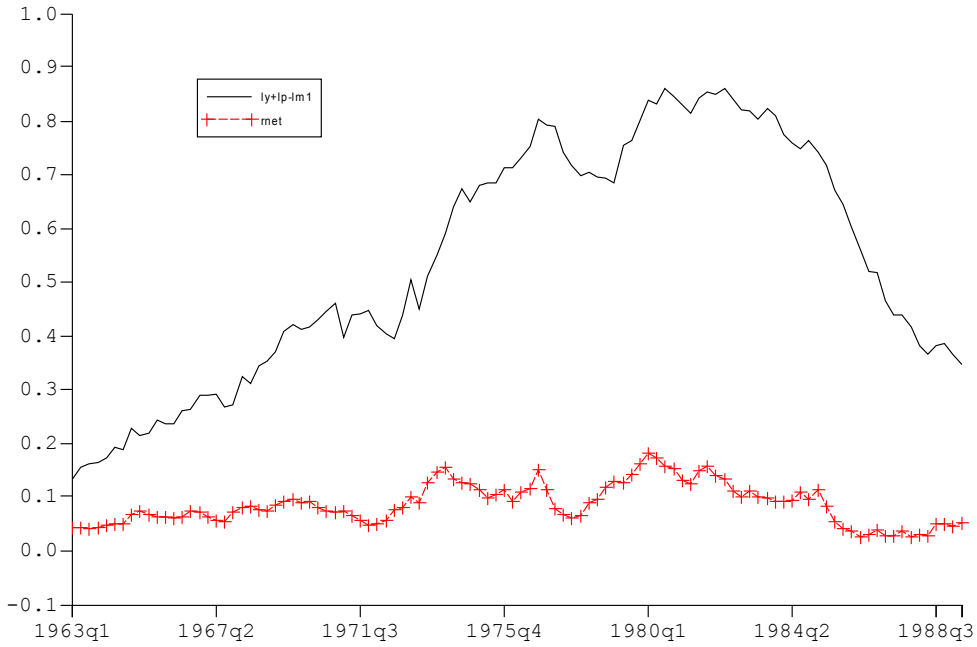
with the result:

```
h6  =

Handle of type "Compound" with properties:
==========================================
parent: Axes
children: ["Text";"Polyline";"Text";"Polyline";"Rectangle"]
visible = "on"
user_data = []
```

and again the first “polyline” represents the second series in the legend:

```
-->h7=h6.children(2)
```

Its properties can now be modified in accordance with the series:
```
-->h7.foreground=5; h7.mark_style=1;h7.mark_foreground=5
```



GROCER contains a function font_title, that allows the user to change the font of the title without having to deal with the corresponding handle. The call is the following:

```
--> font_title(n)
```

where n is the chosen font.

To conclude this part, my advice should be that, unless you have very specific needs as regards graphs, you should use standard Scilab (plot2d, plot2d3, xrects,…) and grocer (pltseries, pltseries0,…) tools. (°°)

5. The functions and their specifications.

dealyna_____splits y series containing Nas

CALLING SEQUENCE
[listx,listy]=dealyna(y)

PARAMETERS

INPUT:
y = a row or column vector

OUTPUT:
* listx = list of coordinates corresponding to non NA vectors
* listy = list of vector of values homogenous with respect to NA's

DESCRIPTION
Splits a vector y into a series of vectors, each containing only non NA successive values; saves them into a list and save into another list the vectors of corresponding observations.

Examples:
1) m=[1 %nan %nan %nan 3 4 3 %nan 6];[listx,listy]=dealyna(m)
2) [listx,listy]=dealyna(yext)

Example 1 gives:
listy =

listy(1)

1.

listy(2)

! 3. 4. 3. !

listy(3)

6.
listx =

listx(1)

1.

listx(2)

! 5. 6. 7. !

listx(3)

9.
Example 2 taken from function pltseries0().

drawx_____draw a readable x axis

CALLING SEQUENCE
drawx(xscale0,dilat,ref_nbinter,font_axis,y0,x0,ninter,ntics)

PARAMETERS

INPUT:
* xscale0 = a (1xnobs) string vector, representing the scale
* dilat = a scalar, representing the # of series pertaining to a given date (used for bar graphics)
* ref_nbinter = a number indicating the maximum numbers of chars that can be displayed
* font_axis = size of characters on the axis
* y0=value of the y axis where to draw the x axis

OUTPUT:
* nothing: just draw the axis

DESCRIPTION
Draws an x axis which remains readable whatever number of values it contains. The parameter ref_nbinter must be adapted to the size of the screen (function param_graph). Must be used after a 2D graphic instruction (such as plot2d).

Examples:
1) drawx(xscale0,ref_nbinter,font_axis,y0)
2) drawx(string([1:100]),20,3,0)

Example 1 is taken from function pltseries0. Example 2 draws an x axis with range going from 1 to 100, a maximum number of chars on the axis equal to 20, a font size of 3 and at the axis is located at y=0

drawy_____draw a readable y axis

CALLING SEQUENCE
y0=drawy(y,font_axis,y0,x0,dircar)

PARAMETERS

INPUT:
* y = the y matrix of the graph
* font_axis = size of characters on the axis
* y0 = value of the y axis where to draw the x axis or [] if the user wants to put the x axis at the minimum y value
* x0 = value of the y axis where to draw the x axis (default: 1)
* dircar= the tics direction (default: 'l', see drawaxis for details)

OUTPUT:
nothing: all results are plotted

DESCRIPTION
Draws an y axis which remains readable whatever number of values it contains. Must be used after a 2D graphic instruction (such as plot2d).

Examples:
1) y0=drawy(y(:,y_1),font_axis,y0)
2) y0=drawy([sqrt([1:12])' log([1:12])'],3,[],0,'l')

Example 1 is taken from function pltseries0. Example 2 draws an y axis associated with the values of the matrix [sqrt([1:12] log([1:12]).The font size is 3 and the axis is located at x=0, the direction of the tics is the left one. Lastly, drawy calculates the minimum y value to feed frunction drawx.

font_title_____changes the font of the title

CALLING SEQUENCE
font_title(f,wind)

PARAMETERS

INPUT:
* f= size of the font
* wind = number of an open window

OUTPUT:
nothing: result is plotted

DESCRIPTION
Changes the font of the title

Example:
font_title(4,0)

Set to 4 the font of the title of the graph in window number 0.

param_g_____determines the font parameters

CALLING SEQUENCE
param_g(ref_nbinter,maxcar,thresh,font_axis)

PARAMETERS

INPUT:
* ref_nbinter = # of chars that can be drawn on the x axis
* maxcar = maximum font for the title
* thresh = # of chars in the title that causes a decline of 1 unit in the title font
* font_axis= font of the axes

OUTPUT:
nothing: the parameters are saved in scilab database 'SCI/macros/grocer/db/param_g.dat'

DESCRIPTION
Set the values useful to set titles and axes that are readable. Default values suitable for one type of screen cannot be so for others: if this is the case, the user has to enter, by trial and error, the values suitable to him.

Examples:
1) param_g(40,9,8,3)
1) param_g(60,12,11,4)

Example 1 set the default values. Example 2 set values suitable for bigger screens.

plt_dfb_____Plots dfbetas

CALLING SEQUENCE
plt_dfb(res)

PARAMETERS

INPUT:
* res = the results typed list of function dfbeta

OUTPUT:
nothing: all results are plotted

DESCRIPTION
plots the dfbetas coefficients of an ols regression. Dfbetas are plotted 4 by for 4 on the same graphic window.

plt_dff_____plots dffits, studentized residuals ...

CALLING SEQUENCE

plt_dff(res)

PARAMETERS

INPUT:
* res = the results typed list of function dfbeta

OUTPUT:
nothing: all results are plotted

DESCRIPTION
Plots the dffits, studentized residuals and hat-matrix diagonal. Opens a new graphic window.

plt_irf1_____plots impulse functions

CALLING SEQUENCE
plt_irf1(res)

PARAMETERS

INPUT:
res = the results typed list of an IRF

OUTPUT:
nothing: all results are printed on the graphic window

DESCRIPTION

Plots the results of the impulse functions from a var regression.

pltacf_____plots partial autocorrelation fonction

CALLING SEQUENCE

pltacf(res)

PARAMETERS

INPUT:
res= a tlist result from function pacf

OUTPUT:
nothing: all results are printed on the graphic window

DESCRIPTION
Plots partial autocorrelation fonction.

Example
load('SCI/macros/grocer/db/varma_d.dat');elec_cons = transdif(seriesa,0,1,1,12);
respacf=pacf(elec_cons,'m=30','noplt');pltpacf(respacf)

pltbrybos_____plots the results of the Bry-Boschan function

CALLING SEQUENCE

pltbrybos(res)

PARAMETERS

INPUT:
res= a tlist result from function brybsos

OUTPUT:
nothing: all results are printed on the graphic window

DESCRIPTION
Plots the results of the Bry-Boschan function.

Example
load('SCI/macros/grocer/db/pigiron.dat'); rbb1 = brybos('log(pigiron)','proc="bb"','noprint');
pltbrybos(rbb1)

Plots the results of the Bry-Boschan procedure applied to the pigiron data.

pltcusum_____plots the results of the cusum test

CALLING SEQUENCE

pltcusum(res,siz,output)

PARAMETERS

INPUT:
* res = the results typed list of function cusumb or cusumf
* siz = the chosen size for the test (0.01, 0.05 or 0.1)
* out = the symbolic name of the file where the results are printed (default: %io(2))

OUTPUT:
nothing: all results are printed on the graphic window

DESCRIPTION
Plots the results of the cusum test. 3 sizes are available, governed by the parameter siz.

Example
pltcusum(rescusum,0.05)

pltseries0_____2d plot

CALLING SEQUENCE

pltseries0(y,y0,title,xscale0,wind,arg1,...,argn)

PARAMETERS

INPUT:
* y = the (nxp) values real matrix of the graphed series
* y0 = the value where to draw the x axis or [] if the user wants the axis to be put a the y minimum value
* title = title of the graph
* xscale0 = a n(x1) string vector representing the x scale
* wind = the number of the window where to draw the graph (-1 if the user wants to draw it on the currently opened window)
* argi = optional arguments:
- styleg = integer row vector of size p representing the location of the legend
  (default: 5, that is the legend is placed interactively with the mouse, see legends in the help menu)
- style = integer row vector of size p representing the line style of each series
- color = integer row vector of size p representing the line color of each series
- leg = title of the legend
- yaxis = integer row vector of size p representing the axis for each series (1=left; 2=right)
- 'bars = xx' with xx integer row vector of size p representing the nature of the representation of the series (1=bars; anything else = curves)
- 'x0(1)=xx' with xx integer representing the x location of the first y axis (default: put at x=1)
- 'x0(2)=xx' with xx integer representing the x location of the second y axis (default: put at x=nobs)

OUTPUT:
nothing: all results are printed on the graphic window.

DESCRIPTION

Plots series allowing complex x scale (for instance, coming from a ts).

Examples
1) pltseries0(grocer_y,y0,tit,xscale0,wind,'leg='+strleg,yax,varargin(:))
2) pltseries0([log(1:4)' sqrt(1:4)'],[],'example',['winter' 'spring' 'summer' 'autumn'],1)

Example 1 is taken from function pltseries. Example 2 draws a graph with an x axis representing the 4 seasons. Y matrix is [log(1:4)' sqrt(1:4)'], x axis is put at the minimum y value. Title is 'example' and graphic window is numbered 1.

pltseries_____2D plot

CALLING SEQUENCE

pltseries(arg1,...,argn)

PARAMETERS

INPUT:
argi =:
- a time series
- a real (nxp) vector
- a string equal to the name of a time series or a (nxp) real vector between quotes
- a matrix or a list of such elements
- 'title=x' if the user wants to give its own title (default: the name of the ts if it has been given between quotes, the string 'ts' if not)
- 'bounds=["b1" ; "b2" ;...; "bn"]' if the user wants to give its own bounds (default: the whole series)
- 'y0=xx' if the user wants to put the x axis at value xx (default: y minimum value)
- 'yaxis=xx' where xx is a (1xp) matrix of 1 and 2, if the user wants 2 axes, respectively at the left and the right of the graph; the j the series is represented on the left axis if xx(j)==1 and on the right one if xx(j)==2 (default: only a left axis)
- 'bars = xx' with xx integer row vector of size p representing the nature of the representation of the series (1=bars; anything else = curves)
- 'x0(1)=xx' with xx integer representing the x location of the first y axis (default: put at x=1)
- 'x0(2)=xx' with xx integer representing the x location of the second y axis (default: put at x=nobs)
- 'x=xx' where xx is the (1xnobs) string vector to put on the x axis
- styleg = integer row vector of size p representing the style of each piece of the legend
- color = integer row vector of size p representing the line color of each series
- style = integer row vector of size p representing the line style of each series
- leg = title of the legend
- 'window=x' if the user wants the graph to be plotted on window # x (default: the window 1)

OUTPUT:
nothing: all results are printed on the graphic window.

DESCRIPTION

Plots series... This function allows the user to make some complex graphs by gathering in one function some capabilities embodied in Scilab graphic functions drawaxis, legend and xset. The user can graph vectors and matrices, as with plot2d, but also timeseries. She can choose to use 2 y axes instead of one by setting the option 'yaxis=...'. She can choose the position of the x axis with the option 'yaxex=...'. She can control the color and style of the lines and the location of the legend '.

Examples
1) load('SCI/macros/grocer/db/bdhenderic.dat'); pltseries('delts(lm1)','delts(lp)')
2) pltseries('ly+lp-lm1','rnet','title=money velocity and interest rate', 'yaxis=[1 2]','leg=[money velocity (lhs);interest rate (rhs)','color=[2 6]','styleg=3', 'window=3','bounds=["1968q1";"1985q4"]')

Example 1 shows the simplest use of pltseries: only the series (here ts) are entered.
Example 2 shows a complex use: the user chooses her legend, title, colors, style and ts bounds. The legend is drawn in the lower left-hand corner. The first series is represented along the lhs x scale and the second one along the rhs y axis.

plspectral_____plots results of spectral analysis

CALLING SEQUENCE

pltspectral(res,pspec,pcospe,pdcorr,pphase,pcoher,pcohes,pboot)

PARAMETERS

INPUT:
* res= results tlist from the function spectral
* pspec = 1 if the user wants to print the spectrum of all variables
* pcospe = 1 if the user wants to print the cospectrum of all variables
* pdcorr = 1 if the user wants to print the dynamic correlation of the variables
* pphase = 1 if the user wants to print the phase spectrum
* pcoher = 1 if the user wants to print the dynamic coherency of the variables
* pcohes = 1 if the user wants to print the cospectrum of all variables
* pcospe = 1 if the user wants to print the cohesion of the variables
* pboot = 1 if the user wants to plot the confidence bands

OUTPUT:
nothing: all results are plotted on the graphic window

DESCRIPTION
Plots results of spectral analysis.

Example
rspec = spectral('dlger','dlfr','dlit','dlsp','trunc=6','spec=1','cospe=1','noprint') ;
pltspectral(rspec,1,1,1,0,0,0,0)

Plots the spectrum, the cospectrum and the dynamic correlations only from results tlist rspec.

plttvp_____Plots tvp results

CALLING SEQUENCE

plttvp(res)

PARAMETERS

INPUT:
res = the results typed list of a tvp regression

OUTPUT:
nothing (printed on a graphic window).

DESCRIPTION

Plots the coefficients of a time-varying regression.

pltuniv_____plots univariate results

CALLING SEQUENCE
pltuniv(res,options)

PARAMETERS

INPUT:
* res = the results typed list of a univariate regression
* options = 'res', 'fitted', or 'all' according to the results that the user wants to graph (default: all)

OUTPUT:
nothing (printed on a graphic window)

DESCRIPTION
Plots the results of least-squares regression.

Examples:
1) pltres(rhe)
2) pltres(rhe,'fitted','res')

These 2 examples do the same thing: graph residuals on a graph and observed and fitted values on another graph.

prtauto_____prints automatic() results

CALLING SEQUENCE
prtauto(results,lnp,output)

PARAMETERS

* results = a tlist provided by automatic
* lnp = 'prt=opt1,...,opt' where opti is one of the following option:

- 'all': all results are printed
- 'initial': initial model
- 'st0_mod': stage 0 model printed
- 'st1_mod': stage 1 models printed
- 'st1_union': stage 1 union model printed
- 'st2_mod': stage 2 models printed
- 'st2_union': stage 2 union models printed
- 'final': final model printed
- 'st1_path': paths of stage 1 models printed
- 'st2_path': paths of stage 2 model printed
- 'path': stages 1 and 2 paths printed
- 'test_final': specification tests for final model printed
- 'test_inter': specification tests for all intermediate models printed
- 'test' : specification tests for all models printed

* output = the file where to print the results

OUTPUT:
nothing, just print the results

DESCRIPTION
Prints the results of an automatic regression.

Example
prtresults(rauto,list('initial','final','test_final'),%io(2))

Prints on screen selected results of rauto: inital and final models, with specification tests for the final model.

prtauto_multi_____prints some automatic() results

CALLING SEQUENCE
prtauto_multi(results,namemod,test,path,output)

PARAMETERS

INPUT:
* results = a tlist results from automatic()
* namemod = the name of the sub-results to print in the results tlist
* test = %t if the user wants to print the specification tests results ; %f if not
* path = %t if the user wants to print the name of the successively eliminated variables that lead to the printed models
* output = the file where the results are printed

OUTPUT:
nothing, just print the results

DESCRIPTION
Prints the sub-results named namemod in a tlist provided by automatic when there can be several regressions in the corresponding field (i.e stage 1 or 2 models).

Example
1) prtauto_multi(results,'stage 1 models',test_inter,st1_path,output)
2) prtauto_multi(r,'stage 2 models',%t,%f)

Example 1 is taken from function prtauto: test_inter, st1_path and output depends on what options the user has entered in function partauto.
Example 2 displays on screen (no fifth argument) stage 2 models results including specification tests, but without giving the path corresponding to the models.

prtauto_univ_____prints some automatic() results

CALLING SEQUENCE
prtauto_univ(results,namemod,test,output)

PARAMETERS

INPUT:
* results = a tlist results from automatic()
* namemod = the name of the sub-results to print in the results tlist
* test = %t if the user wants to print the specification tests results ; %f if not
* output = the file where the results are printed

OUTPUT:
nothing, just print the results

DESCRIPTION
Prints the sub-results named namemod in a tlist provided by automatic when there there is only one regression in the corresponding field (i.e initial, stage0, union or final model).

Example
1) prtauto_univ(results,'stage 1 union model',test_inter,output)
2) prtauto_univ(r,'final model',%t)

Example 1 is taken from function prtauto: test_inter and output depends on what options the user has entered in function partauto.
Example 2 displays on screen (no fourth argument) the final model results including specification tests.

prtbrybos_____prints the results of a Bry-Boschan dating rules

CALLING SEQUENCE
prtbrbos(res,out)

PARAMETERS

INPUT:
* res = the results typed list of a testing regression
* out = the symbolic name of the file where the results are printed (default: %io(2))

OUTPUT:
nothing: all results are printed on the specified file

DESCRIPTION
Prints the results of a Bry-Boschan dating rules.

Example
load('SCI/macros/grocer/db/pigiron.dat'); rbb1 = brybos('log(pigiron)','proc="bb"','noprint');
pltbrybos(rbb1)

Plots the results of the Bry-Boschan procedure applied to the pigiron data.

prtchi_____prints a Chi2 test

CALLING SEQUENCE
prtchi(res,out)

PARAMETERS

INPUT:
* res = the results typed list of a testing regression
* out = the symbolic name of the file where the results are printed (default: %io(2))

OUTPUT:
nothing: all results are printed on the specified file

DESCRIPTION
Prints the results of a Chi2 test.

Example
prtchi(rarlm)

Example 1 taken from function arlm: since there is no second argument, results are printed on screen.

prtdiebmar_____prints the results of the Diebold-Mariano statistics

CALLING SEQUENCE
prtchi(res,out)

PARAMETERS

INPUT:
* res = the results typed list of a testing regression
* out = the symbolic name of the file where the results are printed (default: %io(2))

OUTPUT:
nothing: all results are printed on the specified file

DESCRIPTION
Prints the results of the Diebold-Mariano statistics.

Example
prtdiebmar(res)

Example taken from function diebmar: since there is no second argument, results are printed on screen.

prtdisag_____prints the results of the Diebold-Mariano statistics

CALLING SEQUENCE
prtdisag(res,out)

PARAMETERS

INPUT:
* res = the results typed list of a disaggregation function
* out = the symbolic name of the file where the results are printed (default: %io(2))

OUTPUT:
nothing: all results are printed on the specified file

DESCRIPTION
Prints the results of disaggregation functions (Chow-Lin, Fernandez or Litterman).

Example
prtdiebmar(res)

Example taken from function chowlin: since there is no second argument, results are printed on screen.

prtfish_____prints a Fisher test

CALLING SEQUENCE
prtfish(res,out)

PARAMETERS

INPUT:
* res = the results typed list of a testing regression
* out = the symbolic name of the file where the results are printed (default: %io(2))

OUTPUT:
nothing: all results are printed on the specified file

DESCRIPTION
Prints the results of a Fisher test

Example
prtfish(rarlm)

Example taken from function arlm: since there is no second argument, results are printed on screen.

prthausmann_____prints a Hausmann test

CALLING SEQUENCE
prthaussman(res,out)

PARAMETERS

INPUT:
* res = the results typed list of a testing regression
* out = the symbolic name of the file where the results are printed (default: %io(2))

OUTPUT:
nothing: all results are printed on the specified file

DESCRIPTION
Prints the results of a Hausman, test

Example
prthausman(res,%io(2))

prtjohan_____prints Johansen cointegration results

CALLING SEQUENCE
prtjohan(res,out)

PARAMETERS

INPUT:
* res = the results typed list of a testing regression
* out = the symbolic name of the file where the results are printed (default: %io(2))

OUTPUT:
nothing: all results are printed on the specified file

DESCRIPTION
Prints the results of a Johansen cointegration test on the file out

Example
prtjohan(result,%io(2))

Example taken from function johansen; results are printed on screen.

prtjohvec_____prints Johansen cointegration vectors

CALLING SEQUENCE
prtjohvec(res,nbr,out)

PARAMETERS

INPUT:
* res = the results typed list of a johansen regression
* nbr = # cointegration vectors selected
* out = the symbolic name of the file where the results are printed (default: %io(2))

OUTPUT:
nothing: all results are printed on the specified file

DESCRIPTION
Prints on the file out the first nbr vectors from a Johansen cointegration test.

Example
prtjohvec(result,1,%io(2))

Prints on screen the first vectors from a Johansen cointegration test, whose results have been stored in tlist results.

prtoutliers_____prints outliers detection in the Bry-Boschan function

CALLING SEQUENCE
prtoutliers(nxo,xo,x,dates)

PARAMETERS

INPUT:
* nxo = dates of replacement
* xo = serie of replaced values
* x = initial serie
* dates = vector of dates

OUTPUT:
nothing: all results are printed on the specified file

DESCRIPTION
Prints outliers detection in the Bry-Boschan function.

Example
prtoutliers(nxo,xo,x,dates)

Taken from function turnbb.

prtres_____prints any regression results

CALLING SEQUENCE
prtres(res,out)

PARAMETERS

INPUT:
* res = the results typed list of a johansen regression
* out = the symbolic name of the file where the results are printed (default: %io(2))

OUTPUT:
nothing: all results are printed on the specified file

DESCRIPTION
Prints on the file out the results from any regression that have been saved in results tlist. The function simply determines the method used and calls a specialised printing function such as prtuniv, prtunitr,...

Example
prtres(result,%io(2))

Prints on screen the tlist result coming from some regression (ols, johansen, kpss,..).

prtsys_____prints system regression results

CALLING SEQUENCE
prtsys(res,out)

PARAMETERS

INPUT:
* res = the results typed list of a johansen regression
* out = the symbolic name of the file where the results are printed (default: %io(2))

OUTPUT:
nothing: all results are printed on the specified file

DESCRIPTION
Prints results of a sur, twosls or threesls estimation saved in tlist res.

Example
prtsys(result,%io(2))

Prints on screen the tlist result coming from a sur, twosls or threesls regression.

prttvp_____prints tvp estimation results

CALLING SEQUENCE
prttvp(res,out)

PARAMETERS

INPUT:
* res = the results typed list of a johansen regression
* out = the symbolic name of the file where the results are printed (default: %io(2))

OUTPUT:
nothing: all results are printed on the specified file

DESCRIPTION
Prints time varying parameters regression results.

Example
prttvp(result,%io(2))

Prints on screen the tlist result coming from a tvp regression.

prtunitr_____prints unit root estimation results

CALLING SEQUENCE
prtunitr(res,out)

PARAMETERS

INPUT:
* res = the results typed list of a johansen regression
* out = the symbolic name of the file where the results are printed (default: %io(2))

OUTPUT:
nothing: all results are printed on the specified file

DESCRIPTION
Prints the results from a unit root test (adf, kpss, phillips-perron or Schmidt-Phillips).

Example
prtunitr(result,%io(2))

Prints on screen the tlist result coming from a unit root regression.

prtuniv_____prints univariate estimation results

CALLING SEQUENCE
prtuniv(res,out)

PARAMETERS

INPUT:
* res = the results typed list of a johansen regression
* out = the symbolic name of the file where the results are printed (default: %io(2))

OUTPUT:
nothing: all results are printed on the specified file

DESCRIPTION
Prints the results from a univariate test (ols, olst, olsc, olsar1, nls, iv, logit, probit, tobit, Theil-Goldberger, hwhite, lad, ridge or robust).

Example
prtuniv(result,%io(2))

Prints on screen the tlist result coming from a univariate regression.

prtvar_____prints VAR estimation results

CALLING SEQUENCE
prtvar(res,out)

PARAMETERS

INPUT:
* res = the results typed list of a johansen regression
* out = the symbolic name of the file where the results are printed (default: %io(2))

OUTPUT:
nothing: all results are printed on the specified file

DESCRIPTION
Prints the results from a var model (var, bvar, ecm, becm).

Example
prtvar(result,%io(2))

Prints on screen the tlist result coming from a VAR regression.

prtvarf_____prints VAR forecast results

CALLING SEQUENCE
prtvar(res,out)

PARAMETERS

INPUT:
* res = the results typed list from a johansen regression
* out = the symbolic name of the file where the results are printed (default: %io(2))

OUTPUT:
nothing: all results are printed on the specified file

DESCRIPTION
Prints the results from a var forecast (from varf function).

Example
prtvarf(result,%io(2))

Prints on screen the tlist result coming from a VAR forecast.

yscale_____ determines y scale

CALLING SEQUENCE
sca=yscale(y)

PARAMETERS

INPUT:
* y = the y matrix of the graph

OUTPUT:
sca=the vector of the scale values

DESCRIPTION
Determines from the y values the values of the y scale.

Example
sca=yscale(y)

Example taken from function drawy.