JLdraw User's Guide

Gilbert Ritschard
Gilbert.Ritschard@themes.unige.ch

April 18, 2001

This user's guide is meant to help people start with the nice JLdraw drawing macros written by Jean-Luc Doumont. See also his article "Drawing Effective (and Beautiful) Graphs with TeX", *TUGboat* 20(3), 1999. The present guide describes the usage of the more important commands. It is not exhaustive however.

JLdraw makes extensive use of `\special` commands that are driver dependent. Hence, be sure to have a version of JLdraw.tex that is compatible with your TEX to Postscript driver. JLdraw has originally been written for Textures and I have adapted it with the help of Jean-Luc Doumont for dvips. I have also written a short JLdraw.sty for use with LATEX 2_ε. JLdraw.tex can be obtained from Jean-Luc Doumont (JL@JLconsulting.be).

1 Usage

With LATEX 2_ε, load JLdraw with

```
\usepackage{JLdraw}
```

Start a drawing with the environment

```
\begin{JLdraw}[color]{width}{height}
...
\end{JLdraw}
```

Possible *color* specifications are: `\white`, `\black` or something like `\lgray` defined as `\newcommand\lgray{\gray{12}}`. See Section 2.5 for more details.

The `JLdraw.sty` file defines basic settings which are automatically inserted when `\begin{JLdraw}` is invoked. These settings can be customized by adding your own settings with

```
\mydrawinit{settings}
```

`JLdraw.sty` defines a length `\JLunit` that takes 1cm as default value. We suggest to use it as measurement unit for *width* and *height*. By changing the value of `\JLunit` before invoking `\begin{JLdraw}` it is then easy to rescale the graphic. For example, `\JLunit=1.5\JLdraw` would increase the size of the drawing area by 1.5.

With plain TEX, load JLdraw with

```
\input JLdraw.tex
\Prolog
```

A drawing is then started with

```
\draw[color]{width}{height}
...
\enddraw
```

2 Commands

2.1 Drawing Area Settings

`\drawframe`

draws a frame using the coordinates of the `\begin{JLdraw}`.

`\xrange{`xl`}{`xr`}{`den`}`

`\yrange{`yl`}{`yr`}{`den`}`

To specify that the x values range from xl/den to xr/den where xl, xr and the denominator den are integer values.

`\xaxis[`*options*`]{`x_0`}{`$incr$`}{`den`}{#`*majorTicks*`}{#`*minorTicks*`}`

`\yaxis[`*options*`]{`y_0`}{`$incr$`}{`den`}{#`*majorTicks*`}{#`*minorTicks*`}`

where x_0 is the coordinate of the first major tick mark, $incr$ is the increment between major tick marks, den a denominator that applies to the first two arguments, #*majorTicks* is the number of major tick marks in addition to the first one and #*minorTicks* the number of minor tick marks between two successive major tick marks.

Possible *options* are:

`\axistrue\axisfalse` (bottom or left axis)
`\mirrortrue\mirrorfalse` (top or right axis)
`\mirrorlabeltrue\mirrorlabelfalse`
`\Gridtrue\Gridfalse`
`\labeltrue\labelfalse`
`\linetrue\linefalse`

`\logtrue\logfalse` (log scale)

default values beeing `\axistrue\labeltrue\mirrortrue\linetrue`. The length of the tick marks may also be set in the *options* argument, for instance

```
\xaxis[\Ticklt=5pt\ticklt=.5\Ticklt]{0}{1}{10}{3}{4}
```

Warning: the axis labels may get strange values when *den* is not a power of 10.

The commands `\xaxis` and `\yaxis` define how the axes are drawn and their numerical labeling. Use `\an` for special or non numerical labeling.

`\setdraw{`*size*`}`

Sets the length of the major tick marks to *size*/2 and that of the minor thick marks to *size*/4. Sets also the distance between the y labels and the y axis to *size*.

Here are some additional lengths that be controlled:

`\Ticklt`	Length of large tick
`\ticklt`	Length of small tick
`\dyt`	Top relative offset
`\dyb`	Bottom relative offset
`\dxl`	Left relative offset
`\dxr`	Right relative offset

Length can be in grid units using `\pc` as measurment unit. For example: `\Ticklt=.25\pc`.

2.2 Labels and Other Objects

`\xlabel{`*text*`}`

puts a label below the x-axis.

`\marginlabel{`*text*`}`

puts a label next to the y-axis. The distance from the y-axis is controlled by the length `\mwd` (`\mwd=1\JLunit` for example).

`\toplabel{`*text*`}`

puts a title above the plot area.

`\an[`*dx/dy*`]{`*x*`}{`*y*`}{`*text*`}{`*hpos*`}{`*vpos*`}`

positions the *text* (that can be any object) at the point (x, y). The alignment is determined by *hpos* (l, c or r) and *vpos* (t, m or b) that define which side of the object will be positioned at (x, y). Small displacements from the point (x, y) can be controlled by *dx* and *dy*. The default values for these last two parameters is set with `\dx=`*length* and `\dy=`*length*.

To align *text* with the labels generated by the `\xaxis`, set *vpos* to `b` and `\dy=-\dyb`. Likewise, for the vertical axis, set *hpos* to `l` and `\dx=-\dxl`.

`\place{`*x*`}{`*y*`}{`*text*`}`

`\nudge{`*x*`}{`*y*`}{`*text*`}`

Both commands left align *text* at (x, y). The first one `\place` puts the baseline at y, while `\nudge` puts the baseline slightly (-.25\pc) below y.

Warning 1: space needed for the labels should be manually inserted before and after the drawing. JLdraw does not take account of it and labels may overwrite surrounding text.

Warning 2: the position of the labels relative to the plot area may change when the plot is centered.

2.3 Data

`\linedata[`*psoption*`]` ⟨*data*⟩ `\enddata`

`\markerdata[`*psoption*`]` ⟨*data*⟩ `\enddata`

`\areadata[`*psoption*`]` ⟨*data*⟩ `\enddata`

`\functiondata[`*psoption*`]` ⟨*data*⟩ `\enddata`

`\ccolumndata[`*psoption*`]` ⟨*data*⟩ `\enddata`

`\rcolumndata[`*psoption*`]` ⟨*data*⟩ `\enddata`

`\cityscapedata[`*psoption*`]` ⟨*data*⟩ `\enddata`

`\wiskdata[`*psoption*`]` ⟨*data*⟩ `\enddata`

`\hwiskdata[`*psoption*`]` ⟨*data*⟩ `\enddata`

`\boxdata[`*psoption*`]` ⟨*data*⟩ `\enddata`

`\hboxdata[`*psoption*`]` ⟨*data*⟩ `\enddata`

The data ⟨*data*⟩ must be given as a space separated list of pairs

$$x_1\ y_1\ x_2\ y_2\ \ldots\ x_n\ y_n$$

except for `\wiskdata`, `\hwiskdata`, `\boxdata` and `\hboxdata` for which triplets must be provided.

The `\wiskdata` and `\boxdata` require for each point the three coodinates x, y_1, y_2 and draw respectively at each x an error bar or a box that extends from y_1 to y_2. Likewise, `\hwiskdata` and `\hboxdata` require the three coordinates y, x_1, x_2 and draw horizontal error bars or boxes.

The *psoption* is a string with postscript code and possibly commands that generate postscript code. It may be used to define the marker type for `\markerdata` and `\linedata` (the markers are described in Section 2.4), the color (see Section 2.5) and the width of the lines drawn from the data. To set the line width, use the postscript command `setlinewidth`. For example,

```
\def\dgray{\gray{50}}
\linedata[\mecircle{2} \dgray .25 pt setlinewidth]
```

defines the marker symbol as an empty circle, selects dark gray as color for the lines and sets the line width to .25pt (do not forget the space between .25 and pt.)

The same width is used for the markers and the line. To get different widths, draw the line and the markers separately. For instance

```
\linedata[\nomarker .75 pt setlinewidth]
\markerdata[ \mecircle{2} .25 pt setlinewidth].
```

You may also use `slw` as an alias for `setlinewidth`.

Examples of chart specifications and the resulting outcomes are given in section 3.

2.4 Markers

Markers have been optimized so they have approximately the same area for the same value of their argument.

`\drawmarker[`*marker*`]`

draws a marker. To position the marker, use `\an`. For example:
`\an{3}{5}{\drawmarker[\mediamond{4}]}cm`

`\nomarker`

`\markcross{`*size*`}`

Solid markers:

`\mcircle{`*size*`}`

`\msquare{`*size*`}`

`\mtriangle{`*size*`}`

`\mtriangledown{`*size*`}`

`\mdiamond{`*size*`}`

Markers with an empty interior:

`\mecircle{`*size*`}`

`\mesquare{`*size*`}`

`\metriangle{`*size*`}`

`\metriangledown{`*size*`}`

`\mediamond{`*size*`}`

The marker commands generate the postscript definition of the markers. They are intended to be used in the optional argument of the data commands or `\drawmarker`.

2.5 Colors

`\white`

`\black`

`\gray{`*percent*`}`

These command generate the postscript code for switching to the specified color. They are intended as assignment values for

`\bkcolor` (background color)

`\ficolor` (filling color)

as in the next example:

```
\let\bkcolor=\white
\newcommand\lgray=\gray{12}
\let\ficolor=\lgray
```

They can also be passed as argument of the `\draw` command or `\JLdraw` environment.

2.6 Lines and Arrows

JLdraw allows also some elementary drawing with straight lines. Here are the available commands:

`\hl{`x`}{`y`}{`*length*`}`

`\vl{`x`}{`y`}{`*length*`}`

draws an horizontal (`\hl`) or vertical (`\vl`) line of length *length* starting at point (x, y).

`\rl{`x`}{`y`}{`dx`}{`dy`}`

draws a line between (x, y) and $(x + dx, y + dy)$.

Arrow headed lines:

`\ra{`x`}{`y`}{`*length*`}`

`\la{`x`}{`y`}{`*length*`}`

`\ta{`x`}{`y`}{`*length*`}`

`\ba{`x`}{`y`}{`*length*`}`

`\rla{`x`}{`y`}{`dx`}{`dy`}`

`\arl{`x`}{`y`}{`dx`}{`dy`}`

draw respectively an horizontal line with an arrow head at right (`\ra`) or at left (`\la`), a vertical line with an arrow head at top (`\ta`) or at bottom (`\ba`), or an oblique line with an arrow head at end point (`\rla`) or at start point (`\arl`).

The line thickness can be controlled with `\rth=`*length*.

The line type can be controlled with the following commands which apply also to grid lines. For lines produced by the data commands and for markers use the postscript *setlinewidth* command described in section 2.4.

`\setdash{`*dlength*`}{`*wlength*`}`

where *dlength* is the length of the dash and *wlength* that of the white space.

`\setdashdot{`*dlength*`}{`*wlength*`}`

defines the line type as a dash lines with a dot between dashes.

`\ldash`, `\mdash`, `\ddash`

`\nulldash`

sets the line type to continuous (equivalent to setting *wlength* to 0pt).

2.7 Boxes and Diamonds

`\cbx{`*dx*`}{`*width*`}{`*height*`}{`*text*`}` box with centered text

`\lbx{`*dx*`}{`*width*`}{`*height*`}{`*text*`}` box with left aligned text

`\cdm{`*dx*`}{`*width*`}{`*height*`}{`*text*`}` diamond with centered text where the argument are the relative horizontal offset *dx*, the *width* and the *height* of the box or diamond, and *text* the object to be placed in the box.

There is also a special annotation command to place a label at a corner of the diamond:

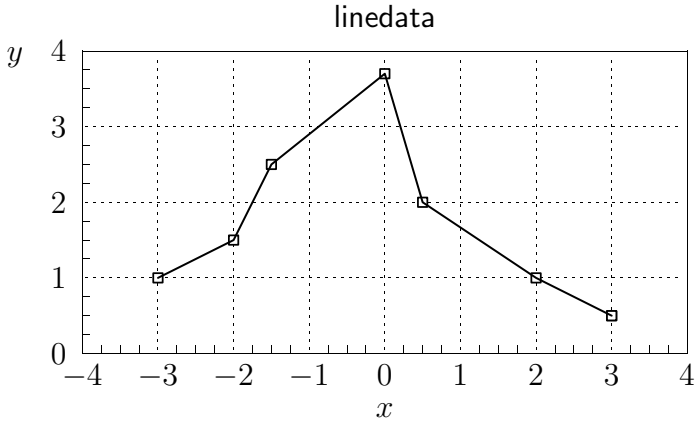
`\dman[`*dx/dy*`]{`*x*`}{`*y*`}{`*text*`}{`*corner*`}{`*pos*`}`

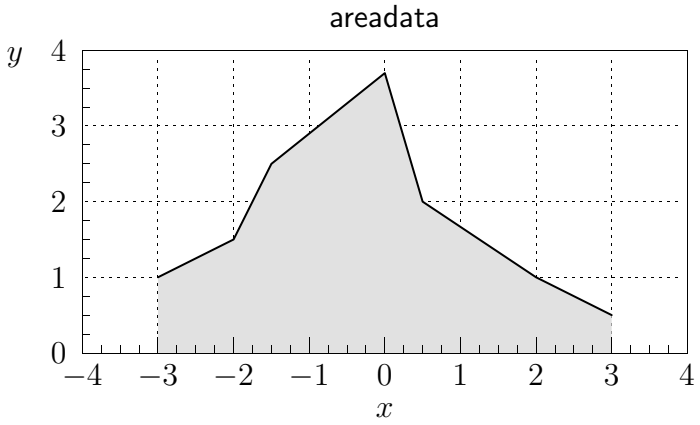
The command is similar to `\an` (Section 2.2) except for *corner* (l, t, r or b) that indicates at which corner of the diamond the annotation applies and *pos* that determines how the label is positioned at that corner: l, t, r or b.

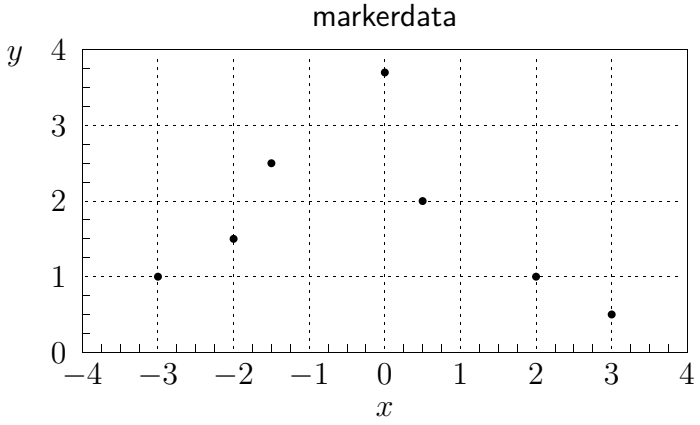
3 Examples

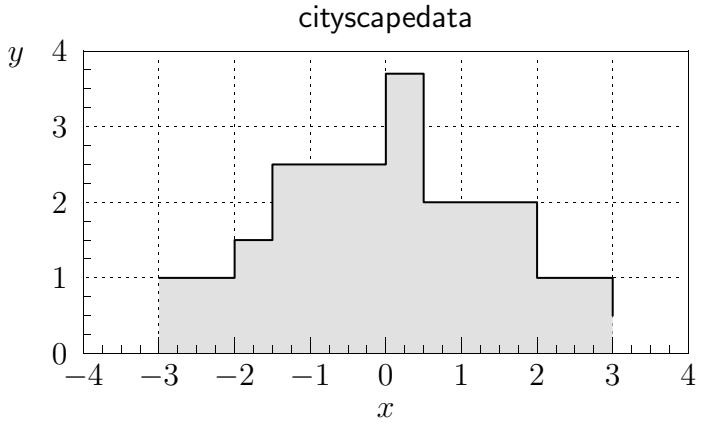
Here are various displays obtained using a same set of data. The code for the first figure is

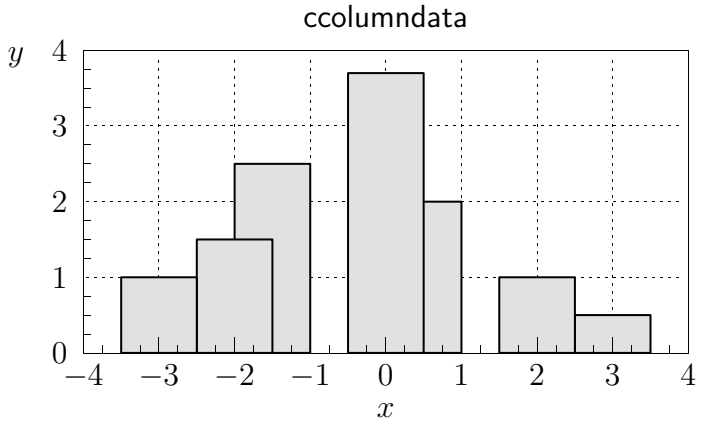
```
\begin{center}
  \begin{JLdraw}{8\JLunit}{4\JLunit}
    \xrange{-40}{40}{10}
     \xaxis[\Gridtrue]{-4}1{1}{8}3
    \yrange{0}{4}{1}
     \yaxis[\Gridtrue]{0}{1}{1}43
    \xlabel{$x$}
    \toplabel{linedata}
    \marginlabel{$y$}
    \linedata[ \mesquare{2} ]
    -3 1
    -2 1.5
    -1.5 2.5
     0 3.7
     0.5 2
     2 1
     3 .5
    \enddata
    \drawframe
  \end{JLdraw}
\end{center}
```

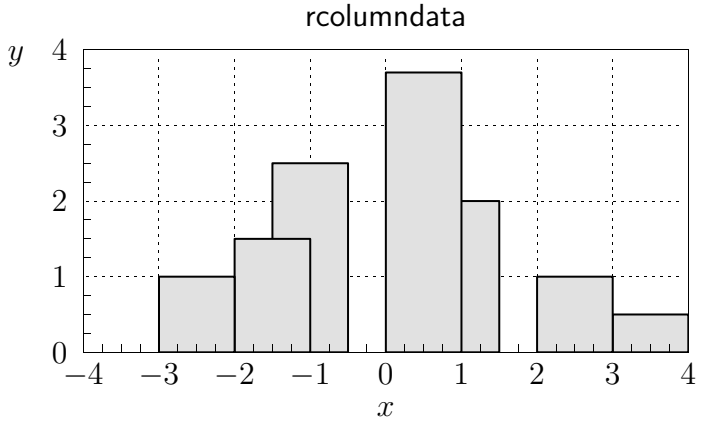

linedata
y
4
3
2
1
0
−4
−3
−2
−1
0
1
2
3
4
x


areadata
y
4
3
2
1
0
−4
−3
−2
−1
0
1
2
3
4
x


markerdata
y
4
3
2
1
0
−4
−3
−2
−1
0
1
2
3
4
x

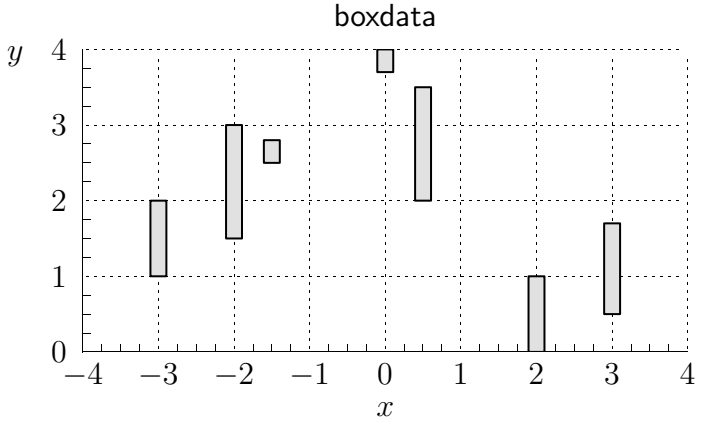

cityscapedata
y
4
3
2
1
0
−4
−3
−2
−1
0
1
2
3
4
x

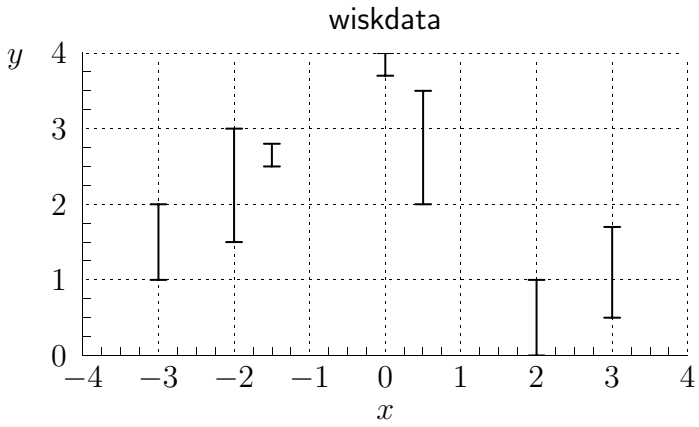

ccolumndata
y
4
3
2
1
0
−4
−3
−2
−1
0
1
2
3
4
x


rcolumndata
y
4
3
2
1
0
−4
−3
−2
−1
0
1
2
3
4
x

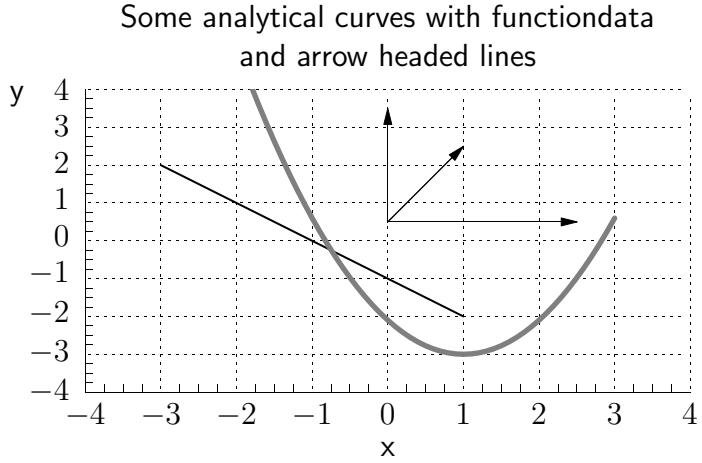
A third column has been added to the data set to display box and error bar plots. The code for the `\boxdata` plot is:

```
\begin{center}
  \begin{JLdraw}{8\JLunit}{4\JLunit}
    \xrange{-40}{40}{10}
     \xaxis[\Gridtrue]{-4}1{1}{8}3
    \yrange{0}{4}{1}
     \yaxis[\Gridtrue]{0}{1}{1}43
    \xlabel{$x$}
    \toplabel{boxdata}
    \marginlabel{$y$}
    \boxdata
    -3 1 2
    -2 1.5 3
    -1.5 2.5 2.8
    0 3.7 4
    0.5 2  3.5
    2 1 0
    3 .5  1.7
    \enddata
  \end{JLdraw}
\end{center}
```





The curves and arrows in the next display



have been generated with the following code

```
\begin{JLdraw}{8\JLunit}{4\JLunit}
    \xrange{-40}{40}{10}
     \xaxis[\Gridtrue]{-4}1{1}{8}3
    \yrange{-4}{4}{1}
     \yaxis[\Gridtrue]{-4}{1}{1}83
    \xlabel{x}
    \marginlabel{y}
    \toplabel{Some analytical curves with functiondata
              \\and arrow headed lines}
```

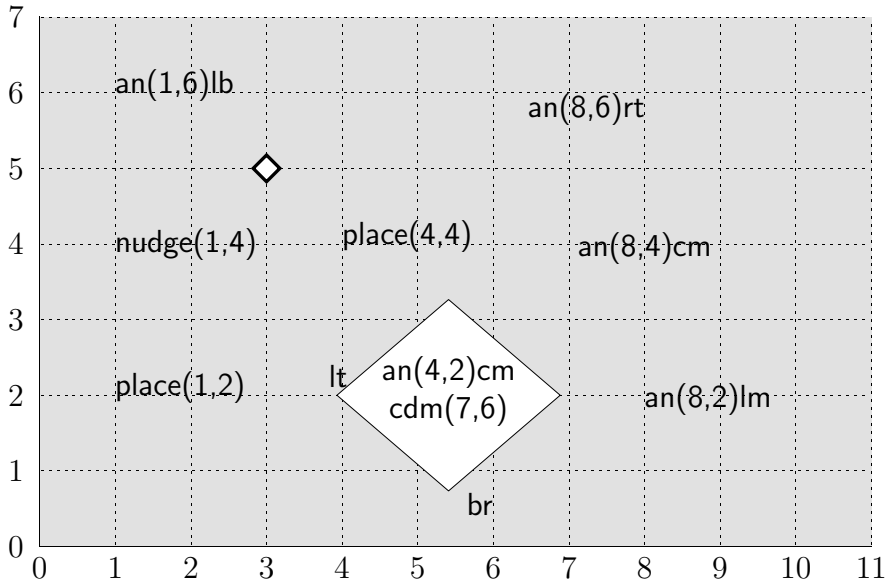
```
    %%%%%  y = -1 - x   between x=-3 and x=1 with step 4
    \functiondata
    /func{neg -1 add}def -3 4 1.05
    \enddata

    %%%%% y = (x-1)^2 - 3  between x=-2 and x=3 with step 0.1
    \functiondata[ 2 pt slw \dgray ]
    /func{-1 add dup mul 0.9 mul -3 add}def -2 0.1 3.05
    \enddata

    %%%%% some arrow headed lines
    \nulldash
    \ra{0}{0.5}{2.5}
    \ta{0}{0.5}{3}
    \rla{0}{0.5}{1}{2}

\end{JLdraw}
```

Here is an example showing the differences between `\place`, `\nudge` and `\an`. It shows also how to use `\cdm` and `\dman`.



The code that generates this example is:

```
\begin{JLdraw}[\lgray]{11\pc}{7\pc}
  \ldash
  \xrange{0}{11}{1}
  \yrange{0}{7}{1}
  \xaxis[\Gridtrue]{0}{1}{1}{11}{0}
  \yaxis[\Gridtrue]{0}{1}{1}{7}{0}

  \nudge{1}{4}{nudge(1,4)}
  \place{4}{4}{place(4,4)}
  \an{8}{4}{an(8,4)cm}cm
  \place{1}{2}{place(1,2)}
  \an{8}{2}{an(8,2)lm}lm
  \an{1}{6}{an(1,6)lb}lb
  \an{8}{6}{an(8,6)rt}rt
  \an{3}{5}{\drawmarker[\mediamond{4}]}cm
  \an{4}{2}{\cdm{0}{7}{6}{\vskip .25\pc an(4,2)cm\\cdm(7,6)\vfil}
            \dman{lt}lt\dman{br}br}cm
\end{JLdraw}
```

Index