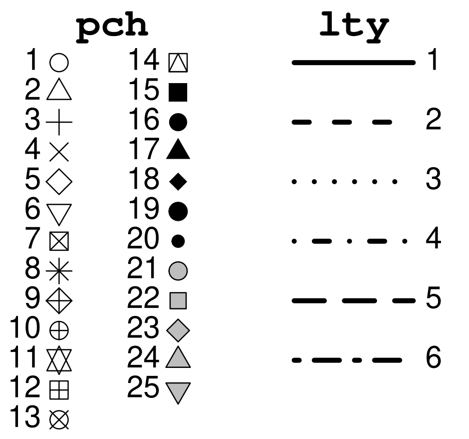
Cheat sheet for **`par()`** graphical parameters, annotation, and **`prepplot`**

Ulrike Grömping

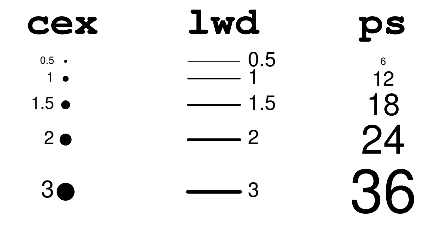
08 April 2021

Cheat sheet for `par()` graphical parameters, annotation, and `prepplot`. Page 1.

Symbols and lines



Sizes and widths

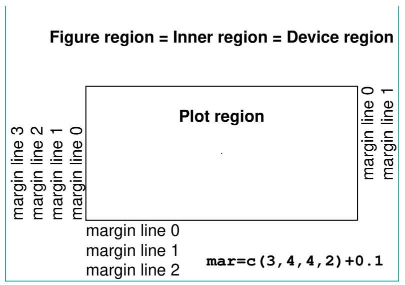


General `par` settings

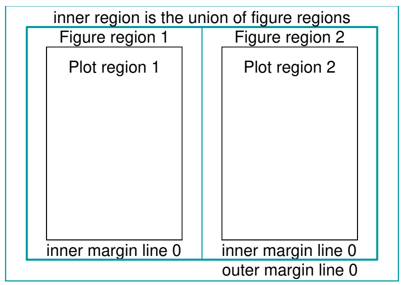
- `bg` background colour for device region (only opaque colors)
- `fg` foreground colour
- `col` default plotting colour
- `font` font type (normal, bold, italic, bold and italic, symbol)
- `xpd` clipping is reduced by `TRUE` and even more by `NA`
- `pty` maximal (default) or square plotting region

Regions: plot, figure, inner and device

Single figure



Arrangement of figures

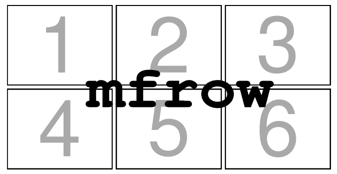


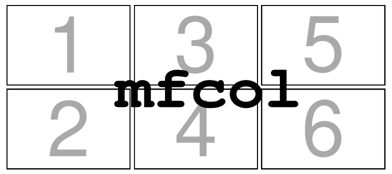
Function `box` draws a box around the `plot`, `figure`, `inner` or `outer` region.

Arrangements

The default `cex` is automatically adapted to the chosen arrangement.

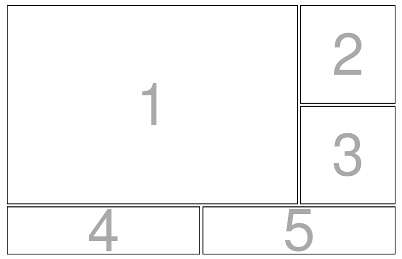
`par(mfrow=c(2,3))` or ditto with `mfcol`





`layout`: more advanced than using only `par`

```
mm <- rbind(c(1,1,2),
            c(1,1,3),
            c(4,5,5))
ws <- c(2,1,1)
hs <- c(2,2,1)
layout(mat=mm,
       widths=ws,
       heights=hs)
```



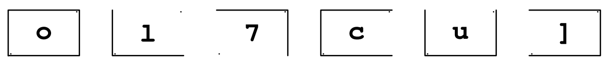
Coordinate systems and extents

Normalized coordinate systems refer to $[0,1] \times [0,1]$, with (0,0) lower left and (1,1) upper right. Coordinates can be queried with `par`, e.g. `par("usr")` (and, more advanced, set). They are vectors `c(x1, x2, y1, y2)`. See function `convertXY` for conversion between systems.

- `usr` axis extremes in user coordinates
- `fig` corners of current figure region on device (as $[0,1] \times [0,1]$)
- `plt` corners of current plot region in figure region (as $[0,1] \times [0,1]$)
- `omd` corners of "region inside outer margins" on device (as $[0,1] \times [0,1]$); these appear to exclude the most outward inner margins, i.e. they are *not* the corners of what function `box` fences in as the inner region.
- `din`, `fin`, `pin`, `cin`: sizes in inch as `(width, height)` for device, figure, plot, character
- `cin`, `cxy`, `cra`: size of a character as `(width, height)` in inches, user coordinates or pixels (not precise, see help for suggestions)
- `mar`, `mai`: bottom, left, top, right margins, in lines or inches
- `oma`, `omi`: bottom, left, top, right outer margins, in lines or inches
- `mex`: character size expansion factor for margins (if larger, `mai` increases relative to `mar`)

Axes

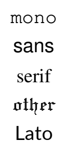
- `tck`, `tcl`: tick length in different units
- `las` tick label orientation
- `lab` for default number of tick marks
- `mgp` distance (lines) of axis elements from plot region
- `xaxs`, `yaxs`: handling of range limits
- `xaxp`, `yaxp`: extreme tick marks and number of intervals (for linear axes)
- `ann` (`FALSE` for suppressing all axis and overall titles)
- `bty` for box type (`n` for none)

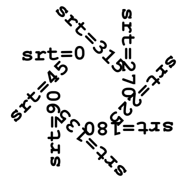


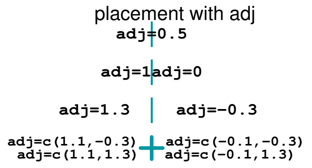
Cheat sheet for **`par()`** graphical parameters, annotation, and **`prepplot`**. Page 2.

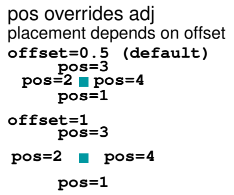
Text elements, e.g. placed with `text(x, y, "mytext", ...)`

- `cex` or `ps`: text size (`cex` is not limited to text size); for margins, there is the separate expansion factor `mex`, which affects the relation between `mar` and `mai`
- `font`: type face of the font (default 1=normal)
- `col`: colour of the font
- `cex`, `font` and `col` have separate versions for axis, labels, main title and sub title, respectively (with `.axis`, `.lab`, `.main` or `.sub`)
- `las`: text orientation (`las=1` often recommended)
- `family`: `mono`, `sans` or `serif`; many others are possible (e.g. `?Hershey`, package **extrafont**).
- `srt`: string rotation (`crt` for character rotation doesn't work on any device I tried)
- `adj` and `pos` (not in `par`) control adjustment of text relative to its x/y position



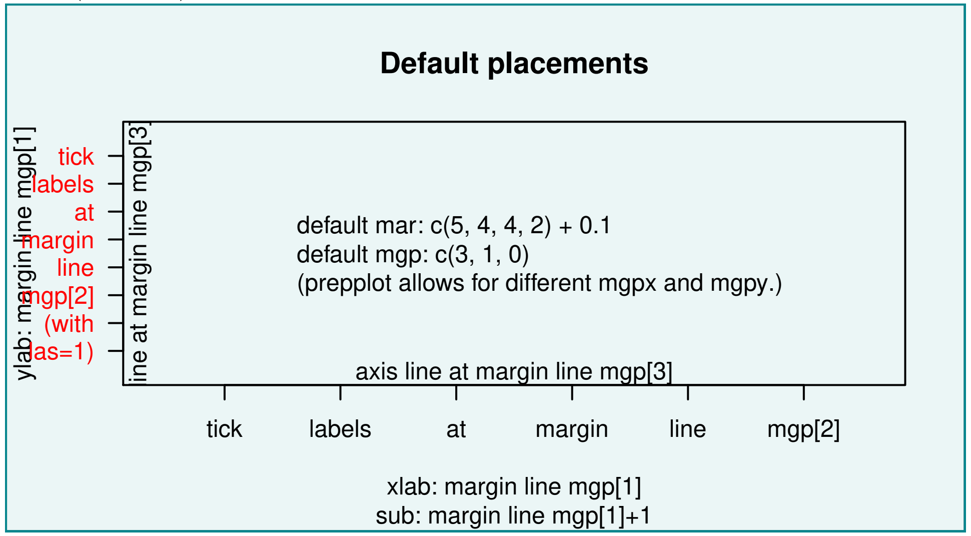






Placing annotation

- `main` (the title) is per default vertically placed in the center of the top margin.


Default placements
default mar: c(5, 4, 4, 2) + 0.1
default mgp: c(3, 1, 0)
(prepplot allows for different mgpx and mgpy.)
axis line at margin line mgp[3]
tick
labels
at
margin
line
mgp[2]
xlab: margin line mgp[1]
sub: margin line mgp[1]+1
ylab: margin line mgp[1]
tick labels at margin line mgp[2] (with las=1)
axis line at margin line mgp[3]

- `par("adj")` governs default horizontal (or parallel to axis) adjustments of `main`, `sub`, `xlab` and `ylab` (one-for-all `par("adj")` is rarely suitable).

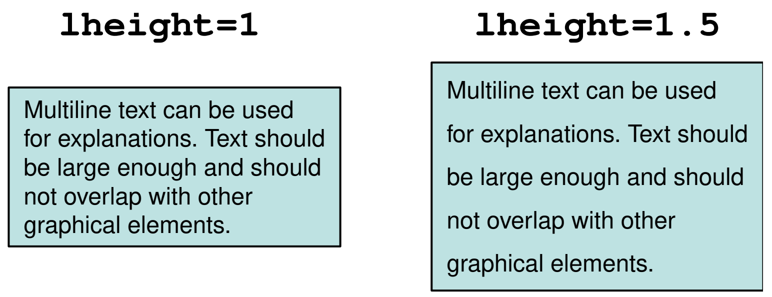
Customize placement of annotation

- `par("mgp")`, `par("adj")`, and font-related `par` settings affect all uses, e.g. in high-level plotting functions.
- Suppress initial annotation for more customization:
  - `axes=FALSE` or `xaxt="n"`/`yaxt="n"` suppresses axes
  - empty strings (e.g. `xlab=" "`) or `ann=FALSE` suppress titles
- custom axis with tick labels: `axis` command(s) (NOT for axis titles)
- `main`, `sub`, `xlab`, `ylab`: `title` command(s)
  - `line` argument (real-valued) allows changing the margin line.
  - possibly, several title commands, even with label for same axis
- `mtext` places text in margins: `line` (real-valued) provides margin line.
  For `outer=FALSE`,
  - `adj` is relative to plot region, `padj` ditto,
  - `at` refers to user coordinates.

  For `outer=TRUE`,
  - `adj` is relative to device region, `padj` ditto,,
  - `at` refers to device coordinates: (0,0)=bottom left, (1,1)=top right.

An aside: Multiline text boxes

- `lheight` line height multiplier for multi-line text (combined with `cex` for actual line height)
- Function `strwrap` creates multiple strings from one long string, `paste` with `collapse="\n"` makes this into a multiline string (`textstring` in the following).
- Functions `strheight` and `strwidth` calculate the height and width required for printing (multiline) texts. Boxes centered at (0.5, 0.5) have been obtained:



lheight=1

Multiline text can be used for explanations. Text should be large enough and should not overlap with other graphical elements.

lheight=1.5

Multiline text can be used for explanations. Text should be large enough and should not overlap with other graphical elements.

```r
w <- strwidth(textstring, cex=1)
h <- strheight(textstring, cex=1)
rect(0.5 - w/2 - w/20, 0.5 - h/2 - h/10,
     0.5 + w/2 + w/20, 0.5 + h/2 + h/10,
     col = mycol, xpd=NA)
```

Cheat sheet for **`par()`** graphical parameters, annotation, and **`prepplot`**. Page 3.

Prepare plotting with function `prepplot` of package **prepplot**

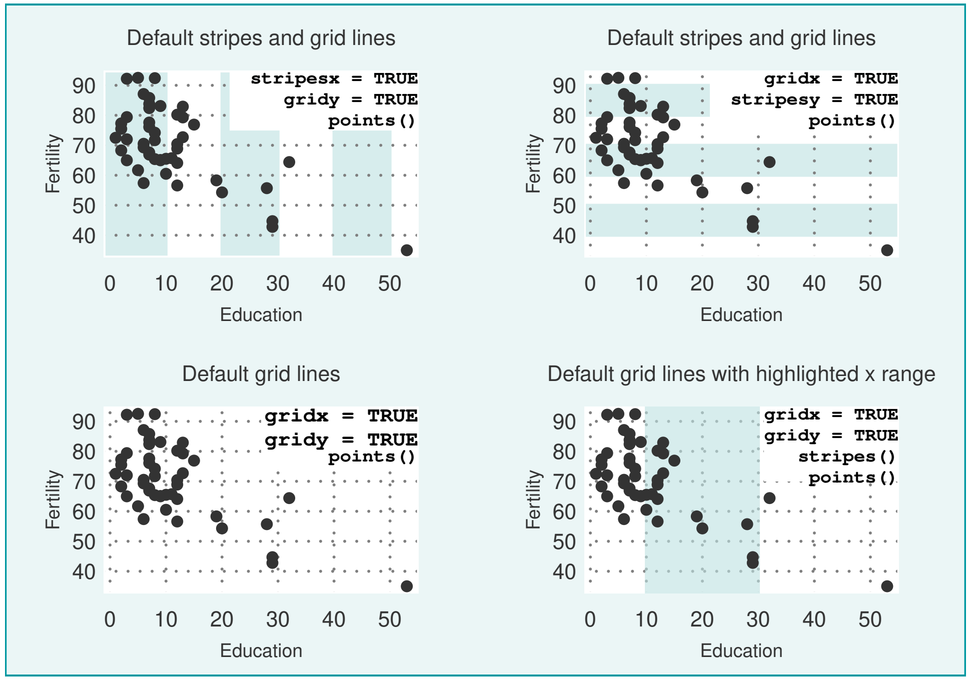
Philosophy of **prepplot**

Package **prepplot** supports custom preparation of the figure area. Data information can then be added. In particular, **prepplot** makes it easier to

- provide a background colour for the plotting region
- allow background stripes in addition to gridlines
  - optionally distinguish minor and major gridlines
- highlight specific value ranges with stripes
  - from within `prepplot` (if no grid lines are needed)
  - or with function **`stripes`** and a transparent stripe colour

`prepplot` respects many `par` settings. It overrides `bty` (always `o`), several colours, and `las` (default 1).

Stripes and gridlines



Everything except `mgp`, font sizes and colour choices is default.

Using high-level plotting functions with **prepplot**

- Highlevel plotting functions with an **`add`** argument can be directly used on **`prepplot`** backgrounds, setting **`add=TRUE`**.
  Example functions: **`barplot`**, **`curve`**, **`plot.histogram`**.
- Many highlevel plotting functions invisibly return relevant plot information, for example:
  - **`barplot`** returns a matrix whose columns are midpoints of (grouped) bars (e.g. for custom labelling of bars).
  - **`hist`** returns a list of class **`histogram`** with all relevant information.
  - **`density`** returns a list of class **`density`** with, among other things, an **`x`** and **`y`** element.
  - **`boxplot`** returns a list of relevant statistics. Numeric locations on the group axis are the position numbers of the **`names`** element of that list.
  - . . .

  These often also permit to suppress plotting (**`plot=FALSE`**).
- A typical workflow would
  - run a plot function with plotting suppressed,
  - use result for determining **`prepplot`** axis limits, tick positions and more,
  - use plot or lines method on stored object, or rerun plot function with **`add=TRUE`**.

Miscellaneous remarks on **prepplot**

- Settings in **`prepplot`** do not modify settings in **`par`**.
- **`mgpx`** defaults to **`par("mgp")`**, **`mgpy`** defaults to **`mgpx`**. Neither modifies **`par("mgp")`**.
- **`xlim`** and **`ylim`** can have more than two elements, their **`range`** is then taken.
  Caution: Make sure the axes contain necessary reference values, e.g. zero on the vertical axis of a histogram.

Colors

- Colors **`"grey0"`** (equal to **`"black"`**) to **`"grey100"`** (equal to **`"white"`**) can be used for quick grey shading, function **`grey.colors`** can provide a palette of grey values.
- Packages like **RColorBrewer**, **pals**, . . . should be used for high quality color palettes.
- Transparent colors should be used, where plot points overlap or background should remain partly visible. Transparency can be achieved with functions **`col2rgb`** and **`rgb`**:
  - **`col2rgb("grey20")`** returns vector of RGB values (here: 51, 51, 51).
  - **`rgb(col[1], col[2], col[3], alpha, maxColorValue = 255)`** adds transparency to a color with RGB values in **`col`** (**`alpha=255`** is opaque, **`alpha=0`** fully transparent).
- Possibilities for color legends:
  - base function **`legend`** (but not good for fills, and placement can be awkward)
  - **`pals::pal.bands`** can showcase a palette
    (use for legend in **`layout`** arrangement on a long horizontal template)
  - more thinking required, but much more flexible: **`plotrix::color.legend`** places a legend rectangle anywhere in the plot region.