Configuring Figure Regions with prepplot

Ulrike Grömping

13 April 2021

Contents

1 Purpose and concept of package prepplot

Base R graphics are very powerful, but the many different possibilities and graphical parameters can be overwhelming. Package **prepplot** has been developed for making it easier to customize figure regions for base R graphics. The concept is to prepare the figure region with `prepplot`, and to then add further graphical elements as desired; this can be done with any function that allows adding to an existing plot.

Inspired by Rahlf (2017), the default choice consists of a lightly colored background, and axis lines, tick marks and boxes are avoided. Orientation in the plot region can be supported by grid lines or background stripes with reasonable defaults regarding line type and color; customization is straightforward and, for example, allows to modify defaults such that a **ggplot2**-like look is obtained. Figure 1 shows a scatter plot on default `prepplot` background with grid lines and on a `prepplot` customization similar to the `ggplot` default, and compares it to a default `ggplot` scatter plot. The code for this figure is shown in the examples section.

prepplot consists of only two user-visible functions, `prepplot` and `stripes`. Function `prepplot` creates a plotting region for an x range and a y range, taking care of background color, and optionally specifying various further design and annotation elements of the figure region.

Function `prepplot` has many arguments, most of which have reasonable defaults and are thus optional. The documentation groups related arguments together in order to support locating arguments. While it is possible and convenient to specify most plot annotations within function `prepplot`, it is also possible to use `prepplot`

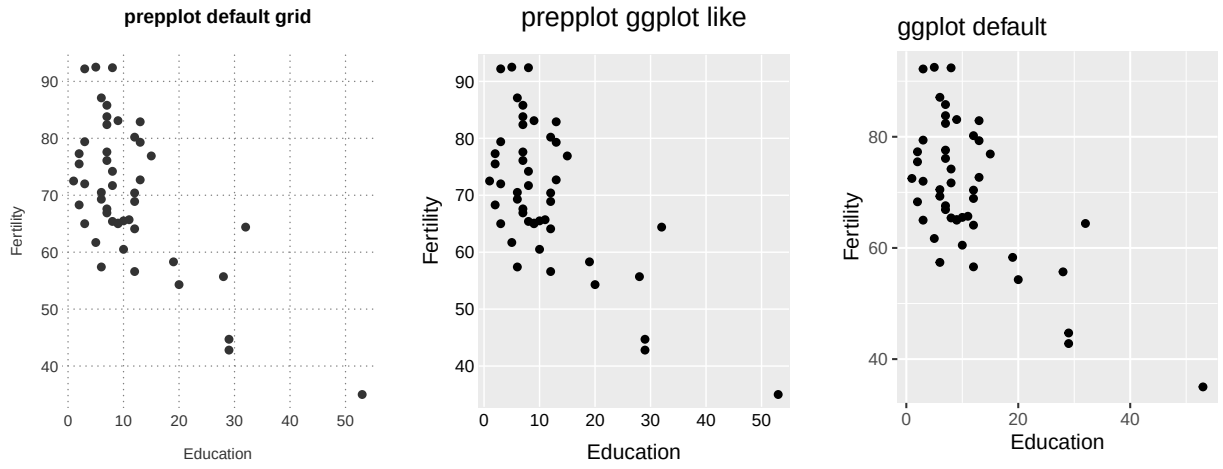

Figure 1: `prepplot` and `ggplot`

for background color and stripes only and to do all further annotation with functions like `mtext`, `title`, `axis`, `stripes` (from **prepplot**) or `abline`.

The only compulsory arguments `xlim` and `ylim` specify the data limits for the two axes. They can either contain numeric vectors of length two like usual, or longer numeric vectors from which the length two vectors are calculated by function `range`. Depending on the choices of the arguments `xaxs` and `yaxs` (or their values in the graphics state, as set/queried by the `par` function), the plot region in the respective direction is 4% larger than specified by these limits (setting `"r"`) or exactly matches these limits (setting `"i"`).

2 Overview of possibilities

2.1 Scope

`prepplot` requires numeric `xlim` and `ylim` specifications; it currently does not work with logarithmic axes. For a date, time or date-time axis, the user has to take care of conversion to numeric values before using `prepplot` (see the last example in the examples section). Axes with character-valued ticks are always based on numeric tick position specifications; see e.g. Figure 5 for a barplot example.

2.2 Default plot regions

The code below creates a default plot region with the default choice for `xaxs` and `yaxs` (assuming that the default has not been changed in `par`) and both choices modified to exact axis limits. The result is shown in Figure 2; the bottom margin has been reduced, because it would produce a lot of empty space between the figure and its caption.

```
x <- 0:10
y <- -5:5
par(mfrow=c(1,2), mar=c(2.1, 4.1, 4.1, 2.1))
prepplot(x, y, main='default axis extension', bg="grey92", cex.axis=0.8, mgpx=c(2,0.25,0))
points(x,y, pch=16)
mtext('xaxs="r"', line=0.25, family="mono")
prepplot(x, y, xaxs="i", yaxs="i", main='limits match axis extent', bg="grey92", cex.axis=0.8, mgpx=c(2
points(x,y, pch=16, xpd=TRUE)
mtext('xaxs="i"', line=0.25, family="mono")
```

`prepplot` obeys to most general graphical parameters as set with function `par` (see `?par`), e.g. `mar`. Where

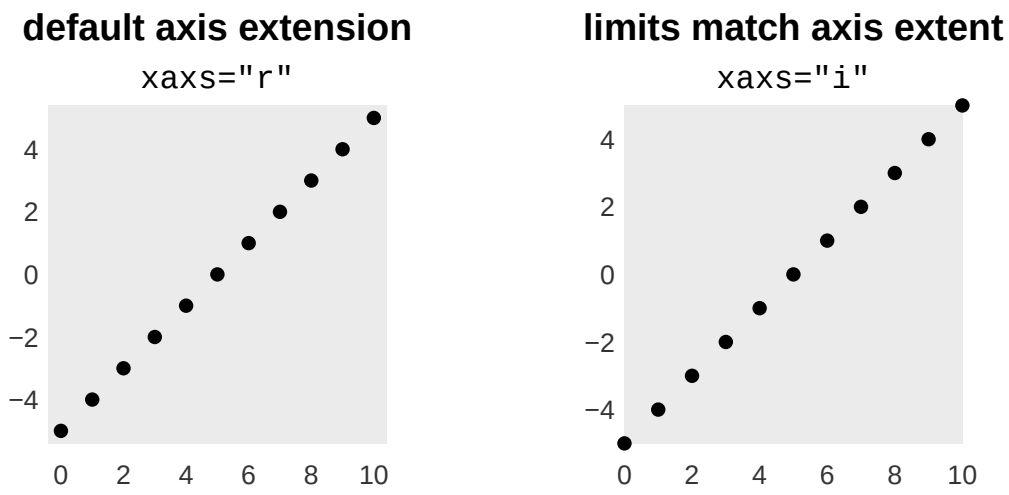

Figure 2: plot regions in `prepplot`, depending on `xaxs` and `yaxs`, where `xlim` and `ylim` correspond to the data ranges. The only structural element is the light grey background color.

further graphical parameters can be modified from within `prepplot` (`bg`, `xaxs`, `yaxs`, `xaxt`, `yaxt`, `cex`), they default to the current `par` settings; this is partly also true for parameters that are set for specific components, like `mgp`, `lwd` or `col`; `las` is an exception: it is per default 1 in `prepplot`, regardless of `par("las")`. `prepplot` does not actively change the graphics state (as queried by `par()`).

For default plot regions,

- the axes are annotated using default tick positions,
- all axis annotations are parallel to the horizontal axis (`las=1`), i.e., easily readable from the reader's perspective (the y axis label, if specified, is parallel to the y axis),
- axis annotation color is not black but a dark grey (`grey20`),
- neither axis lines nor tick marks are drawn (`lwd.axis=0`),
- the border color is the same as the background color (when changing `lwd.axis` to a positive value, the default border color becomes the annotation color specified with `col.axis`),
- no axis labels are printed (`xlab` and `ylab` are `NULL`),
- neither stripes nor grid lines for orientation are drawn (because it is hard to set suitable general defaults for them),
- and the plot background color is identical to the figure background color.

Default plot regions are therefore very unobtrusive: they are neither highlighted by a background color nor bounded by a border line nor indicated by visible axis lines or tick marks. In most cases, one will want to activate one or more structural elements:

- grid lines (`gridx`, `gridy`), possibly major and minor;
- a color difference between plot region and figure region by changing `bg` (e.g. to "grey92", which used to be its default in version~0.7 of the package): `bg` in `prepplot` affects the plot region only, which is the `usr` area calculated by R from the axis limits by extending them by 4% (arguments `xaxs` and/or `yaxs` equal to `"r"` or `NULL`, assuming unchanged `par` specifications) or not at all (arguments `xaxs` and/or `yaxs` equal to `"i"`);
- background stripes (`stripex`, `stripey`) as an alternative to grid lines;
- added box and or axis lines with or without ticks: from within `prepplot` by setting positive `lwd.axis` (box and axis lines and ticks) or with subsequent `axis` and/or `box` statements.

2.3 The axis extent (xaxs and yaxs) and the background area

If `xaxs="r"`, the horizontal extent of the plot region is 4% larger than the range obtained from the `xlim` specification; if `xaxs="i"`, the x extent of the plot region exactly coincides with `range(xlim)`. If `xaxs=NULL`, `xaxs` is determined from the current graphics state `par("xaxs")`. `yaxs` behaves analogously for the vertical extent of the plot region.

The background color colors the plot region, whose coordinates can also be queried by `par("usr")`, and the plot region is framed by the color specified with the `border` argument. If an axis line is not requested (default, `lwd.axis=0`), the default border color equals the background color; otherwise, the default border color equals the axis color.

Note that `xaxs="i"` or `yaxs="i"` should usually not be specified, if data values equal to one of the axis limits exist (see Figure 2). On the other hand, if the axis ranges have been chosen liberally, they may be a good choice, because the orientation marks often look more beautiful.

2.4 Stripes and / or grid lines for orientation

`prepplot` offers stripes and / or grid lines. Respective arguments are handled separately for x and y axes (e.g., `gridx`, `stripesx`). Stripes and grid lines cannot be simultaneously requested for the same axis within function `prepplot`; if both are desired, stripes should be done with `prepplot`, and subsequently grid lines can be added with function `abline`. The simplest use is to activate grid lines or stripes by, e.g., `gridx=TRUE` or `stripesy=TRUE`; in that case, axis tick positions define the locations of grid lines or stripes. For stripes, swapping the colors between `bg` and `col.stripes` can make a big difference, as can be seen in Figure 6 by comparing the top and bottom chart in the middle column.

If grid lines are activated (`gridx` and/or `gridy` set to `TRUE`), these are dotted, whenever there are no minor grid lines and solid otherwise. Their default color is darker than the default background color but lighter than the default axis and annotation (`grey75`). Per default, if activated, minor grid lines are half as wide as major grid lines, whose width is governed by `par("lwd")` per default. If stripes are activated without specifying specific stripes boundaries (`stripesx` and/or `stripesy` set to `TRUE`), the background is interrupted by stripes (default color light grey, `grey90`) between tick positions. Depending on the device's default background color, this default may or may not be adequate.

Apart from using stripes or grid lines for orientation w.r.t. the axis scale, they can also be used for displaying information independent of the tick positions, e.g. grid lines for time points of interest or upper limits of harmful substances, or stripes for indicating time periods of interest in time series presentations (see, e.g., the last example). Such stripes or grid lines can be requested with function `prepplot`, or subsequently with functions `abline` or `stripes`.

2.5 Axis labeling

Per default, `prepplot` displays tick position labelings without ticks, and_if axis labels are given_axis labels. Axis annotation color is controlled by `col.axis`, which defaults to `grey20`, i.e. a dark grey. Tick position labels default to tick positions, but can be modified by `xticklabs`/`yticklabs`, e.g. for character or date values.

If default tick positions from R's calculations are not appropriate, `xticks` can be used for specifying custom tick positions for the horizontal axis. If `xticks` is not given (i.e. remains `NULL`), a vector-valued `gridx` or (if that is not available either) `stripesx` determines the x tick positions; default R tick positions are only calculated if neither of these is given. By giving both `xticks` and one of `gridx` or `stripesx`, stripes or lines corresponding to the horizontal axis can be specified independent of tick mark positions, as mentioned before. `yticks` and friends are completely analogous.

The distance of the tick position labels and axis labels from the plot region (i.e. the colored background, which coincides with the rectangle specified by `par("usr")` coordinates) is controlled by the `mgp` graphical parameter, a three-element vector (default `c(3, 1, 0)`), which gives the number of lines the axis label, the tick mark labels and the axis line are outside the `usr` rectangle (the specified line refers to the closest position,

wide tick position labels will extend into the margin much more than the specified value). Options `mgpx` and `mgpy` can be used to control these positionings independent of the current `par("mgp")` setting and separately for the two axes; both default to `par("mgp")`, and `mgpy` defaults to `mgpx`, if that is specified. One should always think about `mgp` in relation to the margin settings (`par("mar")`); in the author's opinion, both `mgp` and `mar` settings are often too large, and occasionally much too narrow (e.g. for the left margin of horizontal barplots). Whether or not the margin size is appropriate also depends on whether or not non-NULL axis labels (`xlab` and `ylab`) are specified.

Per default, there is no axis line and ticks at the tick positions are not drawn. If `lwd.axis` is set to a positive value, a typical R axis is drawn, per default with a same color border around the plot region, and the axis observes parameter settings like `par("tcl")`; the `...` argument of function `prepplot` can also be used for specifying such parameters, e.g. `tcl=-0.2` specifies shorter tick marks on the outside of the `usr` area than the R default (effective only for positive `lwd.axis`).

Suppressing axes can also be useful, e.g. when subsequently using functions for which it is difficult to suppress axes; an example of this is given in Section 3.3.

2.6 Further annotation

The ...argument of `prepplot` allows to specify a title and / or sub title (with `main` and / or `sub`), and further arguments like `col.main` or `font.main` can be used for their formatting. Alternatively and even additionally, subsequent annotation with `axis`, `mtext` or `title` can be used.

2.7 Axis arrows

`prepplot` allows to add arrows to the plot region: option `axis.arrow` takes care of that. The default axes can be provided with an arrow by setting `axis.arrow` to `TRUE`; this draws axes with width equal to `1.5*lwd.grid`, i.e. `lwd.axis=0` does not prevent axis arrows from being drawn. With `axis.arrow=TRUE`, it is usually advisable to keep the respective `xaxs` or `yaxs` at its default `"r"` (or `NULL`).

A two-element numeric vector `axis.arrow` (instead of a logical) requests axes with arrows in positions given by the two vector elements. `prepplot` keeps axis annotation at the margin of the plot region (oberving `mgp` specification), regardless where axis arrows are placed. An example with default axes with arrows is in the `prepplot` help, and Section 3.5 of this vignette has an example with a differently placed axis arrow.

3 Some examples

The examples in this section illustrate a few uses of function `prepplot`.

3.1 Grid line customization

The following simple example uses major and minor grid lines on white background. The result is shown in Figure 3.

```
par(mar=par("mar")-1, mgp=c(2,0.3,0))
prepplot(c(0,10), c(25,30), yticks=25:30, bg="white",
         lwd.grid.minor=1, gridyminor=4, gridx=TRUE,
         xlab="x axis label", ylab="y axis label",
         main="Major/minor grid on white background")
```

The overview section already showed the default gridlines and customized gridlines mimicking `ggplot` style (Figure 1). The code for that example is shown below; note that it uses package **gridBase** for placing the grid-based `ggplot` figure in the third position of the base graphics layout.

```
par(mfrow=c(1, 3), mgp=c(2,0.2,0), mar=c(3.1,4.1,4.1,1.6))
## prepplot grid default
```

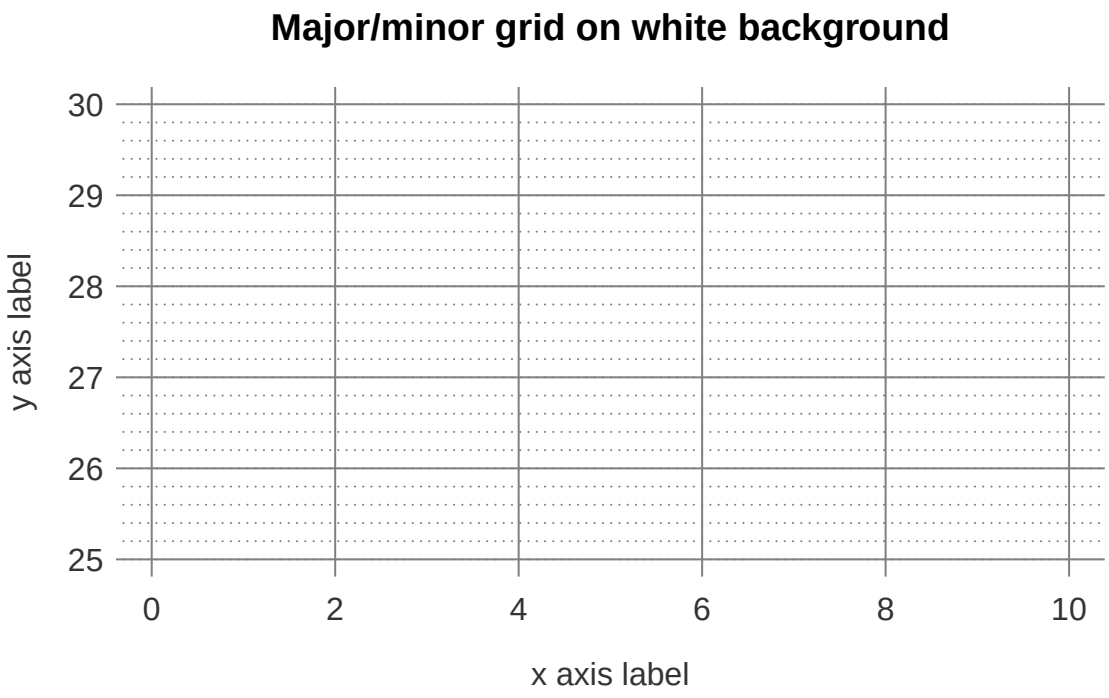

Figure 3: Major and minor grid lines on white background

```
prepplot(swiss$Education, swiss$Fertility,
         ## manage grid
         gridx=TRUE, gridy=TRUE,
         ## annotation
         xlab="Education", ylab="Fertility",
         main="prepplot default grid")
points(Fertility ~ Education, swiss, pch=16, col="grey20", cex=1.3)
## prepplot ggplot like
prepplot(swiss$Education, swiss$Fertility,
         ## background color (like in version 0.7 of prepplot)
         bg = "grey92",
         ## manage grid
         gridx=TRUE, gridy=TRUE, col.grid="white", lty.grid="solid",
         ## manage axis annotation
         cex=0.75, cex.lab=1.35, col.axis="black",
         xlab="Education", ylab="Fertility",
         ## manage title
         main="prepplot ggplot like",
         cex.main=1.75, font.main=1, col.main="black")
points(Fertility ~ Education, swiss, pch=16, cex=1.3)
## add the default ggplot plot
## following the answer in StackOverflow question
# https://stackoverflow.com/questions/14124373/
# combine-base-and-ggplot-graphics-in-r-figure-window
plot.new()
vps <- baseViewports()
```

```
pushViewport(vps$figure) ## in the third plot region
vp1 <-plotViewport(c(0,0.5,1,0.25)) ## create new vp with margins
p <- ggplot(swiss, aes(Education, Fertility)) +
  geom_point() + labs(title="ggplot default")
print(p, vp = vp1)
```

3.2 Using graphical parameters

It will often be of interest to adapt some `par` settings before calling function `prepplot`, e.g. `mar`, `mgp`, `tcl`, `xaxs`, `yaxs`, ... The following code exemplifies such parameter settings, together with a selection of layouts; the results are shown in Figure 4.

```
par(mfrow=c(2,2), mar=c(3.1, 4.1, 3.1, 2.1), mgp=c(2,0.5,0), xaxs="i", yaxs="i")
prepplot(xlim=c(0,9), ylim=c(0,12), bg="grey92", col.grid="grey98",
         lty.grid=1, lwd.grid=1.5, gridx=0:9, gridy=seq(0,12,2),
         xlab="x", ylab="y", cex = 0.7, mgpx=c(1.5,0.25,0),
         mgpy=c(1.75,0.5,0), main="ggplot style")

prepplot(xlim=c(0,9), ylim=c(0,12),
gridx=0:9, gridy=seq(0,12,2),
         xlab="x", ylab="y", cex = 0.7,
         main="default grid lines")

prepplot(xlim=c(0,9), ylim=c(0,12), gridx=0:9, gridy=seq(0,12,2),
         xlab="x", ylab="y", cex = 0.7, lty.grid="solid",
         mgpx=c(2,0.5,0), main="default solid grid lines")

prepplot(ylim=c(0,12), xlim=c(0,9), stripesx=0:9,
         xlab="x", yaxt="n", cex = 0.7, mgpx=c(2,0.5,0),
         main="default vertical stripes, no y axis")
```

3.3 Stripes as background for barplots

The following code creates barplots with transparent bars on striped background. Function `prepplot` provides the plot region and the x axis annotation, while the vertical axis annotation comes from function `barplot`, which is called with the `add=TRUE` argument. Figure 5 shows the result.

```
tab <- c(one=10, two=5, three=8, four=7, five=12, six=10, seven=9)
par(mfrow=c(1,2), mar=c(3.1, 4.1, 3.1, 2.1))
prepplot(ylim=c(0,8.5), xlim=c(-0.2,13),
         yaxs="i", stripesx=seq(0,12,2),
         xlab="Frequency", yaxt="n", cex = 0.8, mgpx=c(2,0.5,0),
         main="Default colors", cex.main=1.2)
barplot(tab, col=rgb(0.5, 0.5, 0.5, 0.7),
        border=FALSE, add=TRUE, xaxt="n",las=2, horiz=TRUE,
        cex.names=0.8)
hks51 <- rgb(0, 152/256, 161/256)
prepplot(ylim=c(0,8.5), xlim=c(-0.2,13),
         yaxs="i",
         xticks=seq(0,12,2), stripesx=TRUE,
         xlab="Frequency",
         bg=rgb(235/256, 246/256, 246/256),
         col.stripes = rgb(190/256, 226/256, 226/256, 0.4),
         col.axis=hks51, cex=0.8, col.main=hks51,
```

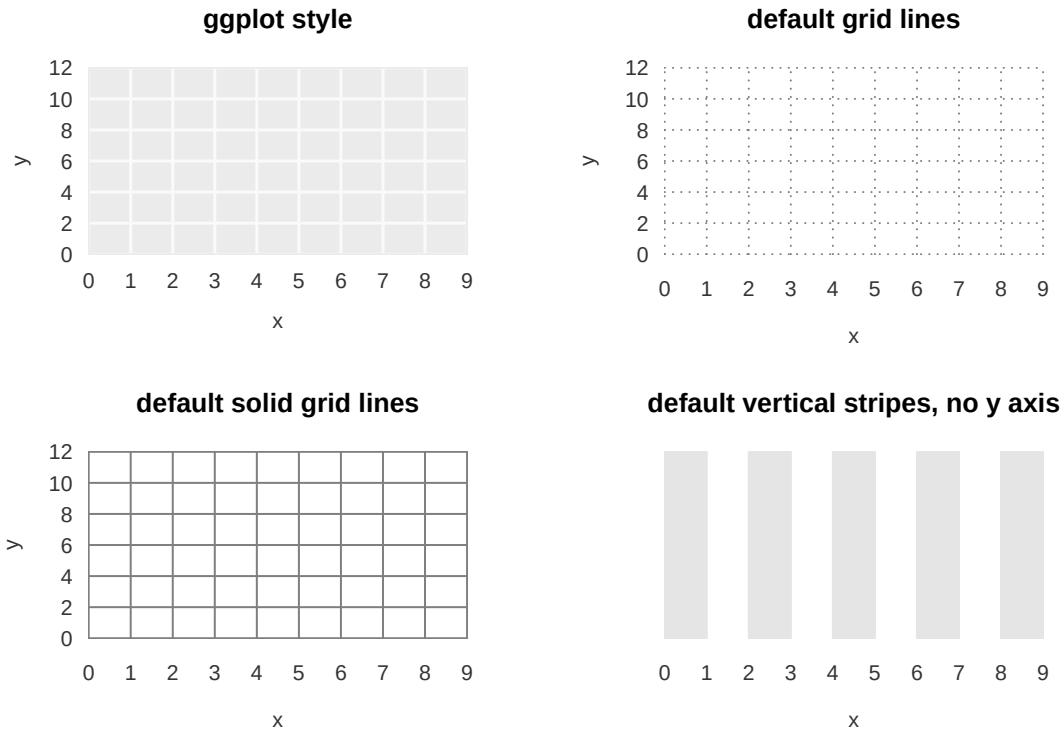

Figure 4: Some examples with `xaxs="i" and yaxs="i"`

```
         yaxt="n", tcl=-0.2, mgpx=c(2,0.5,0),
         main="Beuth colors",
         cex.main=1.2, col.main=hks51)
barplot(tab,
        col=rgb(0,152/256,161/256,0.7), border=FALSE, add=TRUE,
        xaxt="n",las=2, horiz=TRUE, cex.names=0.8, col.axis=rgb(0,152/256,161/256))
```

3.4 Various choices for orientation help

The following code creates scatter plots on various backgrounds, illustrating various orientation strategies, which are on offer for function `prepplot`. The result is shown in Figure 6. Note: The charts use the default axis extent (ranges of `Education` and `Fertility`, extended by 4%); a custom axis with well-chosen limits and suppressing extension by 4% could yield constant widths for all stripes, also the outer ones. Also remember the previous remark on the effect of swapping `bg` and `col.stripes`.

```
par(mfrow=c(2,3), oma=c(2,0,0,0), mgp=c(2.25,0.5,0),
    mar=c(3.1, 4.1, 3.1, 2.1))
## default axis system with grid lines
prepplot(xlim=swiss$Education, ylim=swiss$Fertility, mgpy=c(2.5,0.5,0),
         xlab="Education", ylab="Fertility",
         gridx=TRUE, gridy=TRUE, cex=0.8,
         main="grid lines")
points(swiss$Education, swiss$Fertility, pch=16, col="grey20")

## stripes and grid lines
prepplot(xlim=swiss$Education, ylim=swiss$Fertility,
```

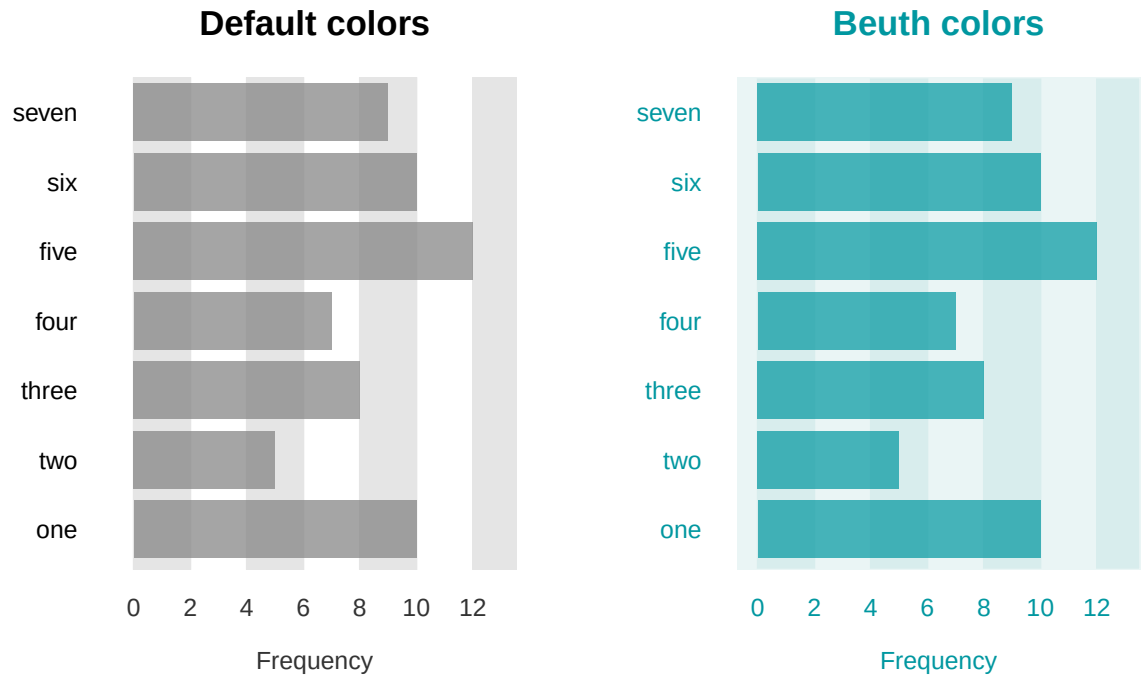

Figure 5: Barplots on striped background, with slightly transparent bars

```
         mgpy=c(2.5,0.5,0),
         xlab="Education", ylab="Fertility",
         stripesx=seq(-5,60,5), gridy=seq(30,100,10), cex=0.8,
         xticks=seq(0,50,10), yticks = seq(40,90,10),
         main="stripes and grid lines")
points(swiss$Education, swiss$Fertility, pch=16, col="grey20")

prepplot(xlim=range(swiss$Education), ylim=range(swiss$Fertility),
         mgpy=c(2.5,0.5,0),
         xlab="Education", ylab="Fertility",
         gridx=seq(0,60,10), stripesy=seq(30,100,5), cex=0.8,
         xticks=seq(0,50,10), yticks = seq(40,90,10),
         main="stripes and grid lines")
points(swiss$Education, swiss$Fertility, pch=16, col="grey20")

## stripes only
prepplot(xlim=range(swiss$Education), ylim=range(swiss$Fertility),
         mgpy=c(2.5,0.5,0), bg="grey90", col.stripes = "white",
         xlab="Education", ylab="Fertility",
         stripesx=seq(-5,60,5), stripesy=seq(30,100,5), cex=0.8,
         xticks=seq(0,50,10), yticks = seq(40,90,10),
         main="crossing stripes")
points(swiss$Education, swiss$Fertility, pch=16, col="grey20")

prepplot(xlim=range(swiss$Education), ylim=range(swiss$Fertility),
         mgpy=c(2.5,0.5,0),
         xlab="Education", ylab="Fertility",
```

```
         stripesx=seq(-5,60,5), cex=0.8,
         xticks=seq(0,50,10), yticks = seq(40,90,10),
         main="only vertical stripes", bg="grey90", col.stripes = "white")
points(swiss$Education, swiss$Fertility, pch=16, col="grey20")

prepplot(xlim=range(swiss$Education), ylim=range(swiss$Fertility),
         mgpy=c(2.5,0.5,0),
         xlab="Education", ylab="Fertility",
         stripesy=seq(30,100,5), cex=0.8,
         yticks = seq(40,90,10), xticks=seq(0,50,10),
         main="only horizontal stripes")
points(swiss$Education, swiss$Fertility, pch=16, col="grey20")

mtext(side=1, line=0.5,
  'Some strategies are better than others ...',
  cex=0.8, outer=TRUE, col="grey20")
```

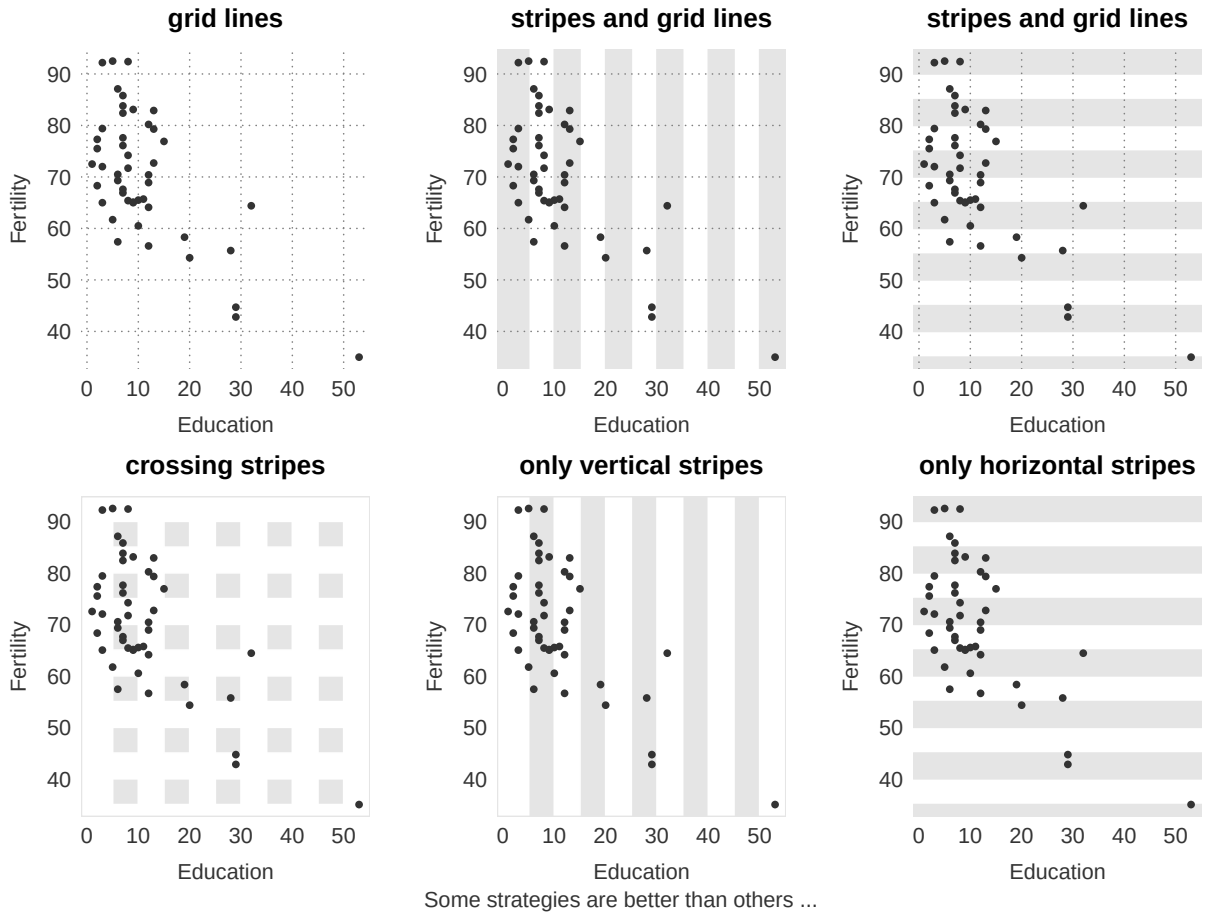

Figure 6: Scatter plots with different orientation mark strategies

3.5 Miscellaneous

This section exemplifies axis arrows (first example, Figure 7), simultaneous use of stripes and grid lines on the same dimension and a date-time axis (second example, Figure 8).

```
prepplot(0:10, -5:5, yticks=seq(-5,5,5),
         stripesy=-5:5,
         xlab="x", ylab="y", main="with stripes and arrow axis", cex=0.8, cex.main=1.2,
         axis.arrow = c(0,NA),arrow.length = 0.5,arrow.width = 0.5,
         lwd=2)
```

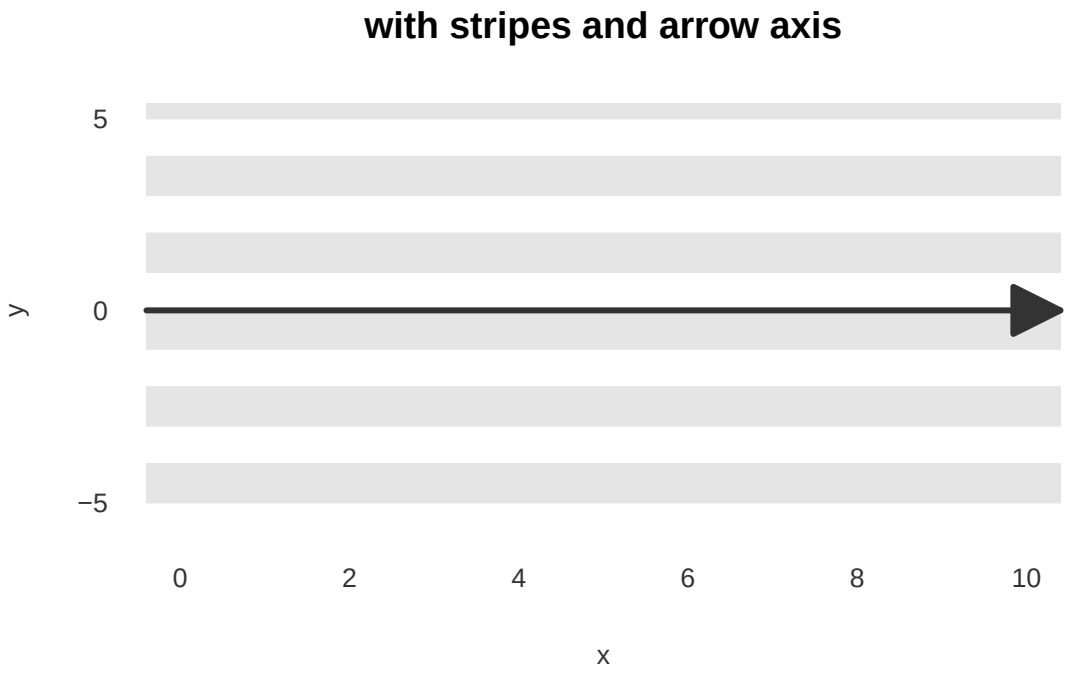

Figure 7: Plot region with horizontal stripes and a horizontal arrow axis at y=0

It sometimes make sense to combine stripes and grid lines for the same dimension, e.g. on a time axis. For example, think of some technical equipment with down times shown as stripes and days separated by grid lines. Such an application is exemplified below.

```
par(mar=c(1.5, 3, 3, 1), mgp=c(3,0.2,0))
xlim <- as.numeric(as.POSIXlt(c("2017-12-28 18:09:46","2018-01-02 09:15:22")))

stripelims <- as.numeric(as.POSIXlt(
  c("2017-12-29 10:30:00","2017-12-29 12:00:00",
    "2017-12-30 10:00:00","2017-12-30 12:30:00",
    "2017-12-31 10:00:00","2017-12-31 12:30:00",
    "2018-01-01 10:30:00","2018-01-01 11:30:00",
    "2018-01-01 16:30:00","2018-01-01 17:45:00")))
griddatts <- as.POSIXlt(c(
    "2017-12-29 00:00:00", "2017-12-30 00:00:00",
    "2017-12-31 00:00:00", "2018-01-01 00:00:00",
    "2018-01-02 00:00:00"))
gridposs <- as.numeric(griddatts)
tickposs <- gridposs + 12*3600
tickposs <- tickposs[tickposs < xlim[2]]
prepplot(xlim=xlim, ylim=c(-5,10),
```

```
         stripesx=stripelims, xaxt="n", cex=0.7,
         gridy=TRUE, gridyminor=4,
         main="Date-time data with stripes and grid lines on same axis")
abline(v=gridposs, col="grey20")
axis(side=1, at=tickposs,
     labels=as.Date(as.POSIXlt(tickposs, origin="1970-01-01 00:00.00 UTC")),
     lwd=0, cex.axis=0.7)
```

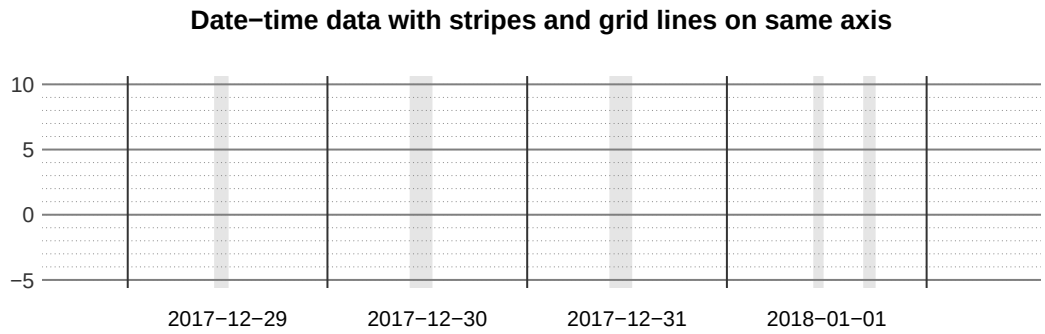

Figure 8: A time series. Stripes show down times, vertical lines show day boundaries

4 References

Murrell, P. (2011). R graphics. CRC, Boca Raton. https://www.stat.auckland.ac.nz/~paul/RG2e/

Rahlf, T. (2017). *Data visualisation with R*. Springer, New York.