A Naming Scheme for Lattice Grobs

Paul Murrell

A Naming Scheme for lattice Grobs

All grobs produced by **lattice** should be provided with names using the following guidelines:

- All names should be generated using a call to `trellis.grobname()`.

  The argument `name` provides the main descriptive name for the grob.

  The argument `prefix` is used to distinguish between different plots on the same page (this defaults using `lattice.getStatus()`). This prefix is prepended to all grob names.

  The argument `type` is used to distinguish between grobs that are once-per-plot and grobs that are once-per-panel. The latter are produced by specifying `"panel"`, `"strip"`, or `"strip.left"` for `type`. In those cases, a suffix is added to the grob name of the form `"panel.i.j"`, where the `i` an `j` come from the arguments `row` and `column` and those arguments get default values from `lattice.getStatus()`). The naming of strip grobs is similar, except that it can be more complicated when there are multiple conditioning variables (and hence multile strips per panel). In that case, the grob suffix will be of the form `"given.k.strip.i.j"`, where `k` comes from the argument `which.given`, but that only occurs if the argument `which.panel` has length greater than 1. Both of those arguments get their default values from `lattice.getStatus()` so there is usually no need to explicitly provide them.

  The argument `group` is used to distinguish between grobs that are once-per-group (within a panel). A value greater than zero causes a suffix of the form `"group.i"` to be appended to the grob name (before any panel suffix).

  The argument `type` can also be `"key"` or `"colorkey"`. These are used to name the components of `draw.key()` and `draw.colorkey()`. The latter is simple because all components are distinct, so you get names of the form `"plot_01.colorkey.border"`. The `draw.key()` components are a little more complex because the contents of a legend can be quite flexible. There are some straightforward components like `"plot_01.key.title"`, but the general contents are of the form `"plot_01.key.text.1.1"` where the `"1.1"` refers to the appropriate column and row of the legend contents.

- In some cases, particularly for once-per-plot grobs, there are one-off **grid** function calls. A good example is the background rect for the entire plot that is drawn by `plot.trellis()`. Such cases only require a descriptive name and `type=""`. For example ...

  ```
  grid.rect(name=trellis.grobname("background", type=""),
            gp = gpar(fill = bg, col = "transparent"))
  ```

  ... produces something like `"plot_01.background"`.

  There are other examples of this sort of once-per-plot grob in some panel functions.

- Most drawing of grobs occurs through a call to one of the primitive functions, `ltext()`, `llines()`, etc (or their "skins", `panel.text()`, `panel.lines()`, etc). The

  These functions produce a basic descriptive name, e.g., `"text"` for `ltext()` and, by default, add panel information. These functions also add group information if appropriate (this is based on detecting a special argument that `panel.superpose()` passes down, i.e., a test that the function has been called by `panel.superpose()`). For example, a direct call to `panel.points()` will produce a grob name of the form `"plot_01.points.panel.1.1"`.

  These functions also have an `identifier` argument which, if non-`NULL`, gets prepended to the descriptive grob name. This allows higher-level panel functions to provide a distinctive prefix for primitive grobs. For example, a call to `panel.xyplot()`, which calls `panel.points()`, will produce a grob name of the form `"plot_01.xyplot.points.panel.1.1"`.

  The `identifier` argument also allows the user to produce distinct names for primitive grobs when writing a panel function. For example, ...

  ```
  mypanel <- function(x, y, ...) {
      panel.xyplot(x, y, ...)
      panel.points(a, b, identifier = "extra")
  }
  ```

  ... will produce something like `"plot_01.xyplot.points.panel.1.1"` for the default points and `"plot_01.extra.points.panel.1.1"` for the custom points.

  There is also a `name.type` argument, which is a character value indicating whether to add panel (or strip) information to the grob name. This corresponds directly to the `type` argument to `trellis.grobname()`. The additional information can be left out by specifying `name.type=""`.

- Higher-level panel functions come in two types. Some, like `panel.abline()`, make direct **grid** calls, so they have a similar set up to the primitive panel functions above. Others make calls to the primitive panel functions, in which case they provide descriptive `identifier` arguments for those calls.

  For example, `panel.densityplot()` may call `panel.points()` to draw raw values and `panel.lines()` to draw the density curve. For both calls, it specifies `identifier = "density"`, so that the resulting grobs have names of the form `"plot_01.density.points.panel.1.1"` and `"plot_01.density.lines.panel.1.1"`.

  These higher-level panel functions also have an `identifier` argument so that users can call them in custom panel functions and provide a unique identifier for grobs that they add.

- Any user customisations to a **lattice** plot that use raw **grid** calls can specify whatever `name` for a grob that they like.