Creating Tables of Text Using `grid`

Paul Murrell

July 9, 2003

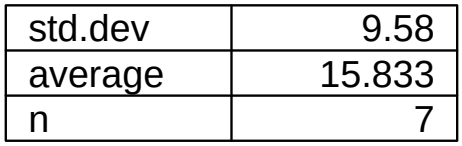
This document discusses the creation of tables of text as graphical output using `grid`. Here is some sample data to be entered into a table:

```
> labels <- c("n", "average", "std.dev")
> values <- c(7, 15.833, 9.58)
```

1 Tables By Hand

Here's a simple table with the calculations done completely by hand: the column widths are specified by hand; the placement of borders is done by hand; the placement of text is done by hand ...

```
> push.viewport(viewport(h = unit(3, "lines"), w = unit(2, "inches")))
> grid.rect()
> grid.grill(v = unit(1, "inches"), h = unit(1:2, "lines"), gp = gpar(col = "black"))
> grid.text(labels, x = unit(0.1, "inches"), y = unit(seq(0.5,
+     2.5, 1), "lines"), just = c("left", "centre"))
> grid.text(values, x = unit(1.9, "inches"), y = unit(seq(0.5,
+     2.5, 1), "lines"), just = c("right", "centre"))
> pop.viewport()
```



Problems with this approach are:

- If you want to change the width of columns, you have to change the placement of borders and text as well.
- The widths of the columns are fixed. If the text ends up wider than 1" (actually, more like (1 - 2*0.1)") then the table will look useless.

2 Tables by Hand with Calculated Widths

The table can be made to depend on the width of the text that is entered. This is achieved using `"strwidth"` (and/or `"strheight"`) units.

```
> push.viewport(viewport(h = unit(3, "lines"), w = unit(4 * 0.1,
+     "inches") + max(unit(rep(1, 3), "strwidth", data = as.list(labels))) +
+     max(unit(rep(1, 3), "strwidth", data = as.list(as.character(values))))))
> grid.rect()
> grid.grill(v = unit(2 * 0.1, "inches") + max(unit(rep(1, 3),
+     "strwidth", data = as.list(labels))), h = unit(1:2, "lines"),
+     gp = gpar(col = "black"))
> grid.text(labels, x = unit(0.1, "inches"), y = unit(seq(0.5,
+     2.5, 1), "lines"), just = c("left", "centre"))
> grid.text(values, x = unit(3 * 0.1, "inches") + max(unit(rep(1,
+     3), "strwidth", data = as.list(labels))) + max(unit(rep(1,
+     3), "strwidth", data = as.list(as.character(values)))), y = unit(seq(0.5,
+     2.5, 1), "lines"), just = c("right", "centre"))
> pop.viewport()
```

std.dev	9.58
average	15.833
n	7

This change means that the table will always accomodate the specified text. However, the calculations for positioning the elements of the table are still inconvenient to specify and modify.

3 Tables Using Layouts

Here's the same table using `grid` layouts.

```
> the.layout <- grid.layout(3, 2, widths = unit(c(1, 1), "inches"),
+     heights = unit(rep(1, 3), "lines"))
> push.viewport(viewport(layout = the.layout))
> for (row in 1:3) {
+     push.viewport(viewport(layout.pos.col = 1, layout.pos.row = row))
+     grid.rect()
+     grid.text(labels[row], x = unit(0.1, "inches"), just = c("left",
+         "centre"))
+     pop.viewport()
+     push.viewport(viewport(layout.pos.col = 2, layout.pos.row = row))
+     grid.rect()
```

```
+     grid.text(values[row], x = unit(1, "npc") - unit(0.1, "inches"),
+         just = c("right", "centre"))
+     pop.viewport()
+ }
> pop.viewport()
```

n	7
average	15.833
std.dev	9.58

The advantages of using layouts to create the table are that the widths of columns can be modified without affecting the placement of the contents of each cell. Also, the placement of the contents of each cell are relative to the cell boundaries, which is easier and more natural to specify.

4 Tables using Frames

`grid` frames may be used simply as a more convenient interface to layouts. To do this, specify a layout in the creation of the frame object, then use the `grid.place()` function (as shown below).

```
> gf <- grid.frame(layout = grid.layout(3, 2, widths = unit(c(1,
+     1), "inches"), heights = unit(rep(1, 3), "lines")), draw = FALSE)
> for (row in 1:3) {
+     grid.place(gf, grid.rect(draw = FALSE), row = row, col = 1,
+         draw = FALSE)
+     grid.place(gf, grid.rect(draw = FALSE), row = row, col = 2,
+         draw = FALSE)
+     grid.place(gf, grid.text(labels[row], x = unit(0.1, "inches"),
+         just = "left", draw = FALSE), row = row, col = 1, draw = FALSE)
+     grid.place(gf, grid.text(values[row], x = unit(1, "npc") -
+         unit(0.1, "inches"), just = "right", draw = FALSE), row = row,
+         col = 2, draw = FALSE)
+ }
> grid.draw(gf)
```

n	7
average	15.833
std.dev	9.58

`grid` frames may also be used to create a more dynamic table. To do this, do not specify the full layout when creating the table, and use `grid.pack` to add the table elements (as shown below).

```
> gf <- grid.frame(layout = grid.layout(3, 6, widths = unit(c(0.1,
+     1, 0.1, 0.1, 1, 0.1), "inches"), heights = unit(rep(1, 3),
+     "lines")), draw = FALSE)
> label1 <- grid.text(labels[1], x = 0, just = "left", draw = FALSE)
> label2 <- grid.text(labels[2], x = 0, just = "left", draw = FALSE)
> label3 <- grid.text(labels[3], x = 0, just = "left", draw = FALSE)
> groblabels <- list(label1, label2, label3)
> for (row in 1:3) {
+     grid.place(gf, grid.rect(draw = FALSE), row = row, col = 1:3,
+         draw = FALSE)
+     grid.place(gf, grid.rect(draw = FALSE), row = row, col = 4:6,
+         draw = FALSE)
+     grid.pack(gf, groblabels[[row]], row = row, col = 2, draw = FALSE)
+     grid.pack(gf, grid.text(values[row], x = 1, just = "right",
+         draw = FALSE), row = row, col = 5, draw = FALSE)
+ }

> gf <- grid.frame(layout = grid.layout(3, 6, widths = unit(c(0.1,
+     1, 0.1, 0.1, 1, 0.1), "inches"), heights = unit(rep(1, 3),
+     "lines")), draw = FALSE)
> label1 <- grid.text(labels[1], x = 0, just = "left", draw = FALSE)
> label2 <- grid.text(labels[2], x = 0, just = "left", draw = FALSE)
> label3 <- grid.text(labels[3], x = 0, just = "left", draw = FALSE)
> groblabels <- list(label1, label2, label3)
> for (row in 1:3) {
+     grid.place(gf, grid.rect(draw = FALSE), row = row, col = 1:3,
+         draw = FALSE)
+     grid.place(gf, grid.rect(draw = FALSE), row = row, col = 4:6,
+         draw = FALSE)
+     grid.pack(gf, groblabels[[row]], row = row, col = 2, draw = FALSE)
+     grid.pack(gf, grid.text(values[row], x = 1, just = "right",
+         draw = FALSE), row = row, col = 5, draw = FALSE)
+ }
> grid.draw(gf)
```

n	7
average	15.833
std.dev	9.58

The difference here is that the layout for the table, in particular the widths of columns and the heights of rows, is based on the widths and heights of the elements that are placed in the table. For example, the widths of each column will be the maximum of 1" – the initial value – and the widths of the pieces of text placed in each column.

Furthermore, the objects representing the contents of the table are retained. This means that the text may be edited and the table will be updated automatically. For example, the size of the text may be modified or the very text itself can be changed.

```
> grid.edit(groblabels[[3]], label = "Standard Deviation")
```

n	7
average	15.833
Standard Deviation	9.58