

Profiling a table layout using Grid

Baptiste Augu  

August 9, 2010

1 Setup

```
> library(grid)
> #library(profr)
> d <- head(iris)
> dlong <- head(iris, 20) # large dataset for final timing
> reps <- 5 # replicates of dlong for the first timing
> ## define some helping functions
> rowMax.units <- function(u, nrow){ # rowMax with a fake matrix of units
+   matrix.indices <- matrix(seq_along(u), nrow=nrow)
+   do.call(unit.c, lapply(seq(1, nrow), function(ii) {
+     max(u[matrix.indices[ii, ]])
+   })))
+ }
> colMax.units <- function(u, ncol){ # colMax with a fake matrix of units
+   matrix.indices <- matrix(seq_along(u), ncol=ncol)
+   do.call(unit.c, lapply(seq(1, ncol), function(ii) {
+     max(u[matrix.indices[, ii]])
+   })))
+ }
```

2 Creating lists of grobs

Two functions are defined here, `makeContent` and `makeContentInVp`. Both return a list of text grobs from a data.frame, but the second one assigns a named viewport to each grob. A third version, `makeContentInVp2` evaluates the possibility of editing the viewport in a list of grobs previously created.

```
> makeContent <- function(d){
+   content <- as.character(unlist(c(d)))
+
+   textii <- function(d, gp=gpar(), name="row-label-"){
+     function(ii)
+       textGrob(label=d[ii], gp=gp, name=paste(name, ii, sep=""))
+   }
+   makeOneLabel <- textii(d=content, gp=gpar(col="blue"), name="content-label-")
+   lg <- lapply(seq_along(content), makeOneLabel)
+
+   list(lg=lg, nrow=nrow(d), ncol=ncol(d))
+ }
```

```

+ }
> makeContentInVp <- function(d){
+   content <- as.character(unlist(c(d)))
+   nc <- ncol(d)
+   nr <- nrow(d)
+
+   ## convert an index to a 2D array of nrow and ncol indices
+   n2nm <- function(nr, nc){
+     expand.grid(seq(1, nr), seq(1, nc))
+   }
+
+   vp.ind <- n2nm(nr, nc)
+
+   textii <- function(d, gp=gpar(), name="content-label-"){
+     function(ii)
+       textGrob(label=d[ii], gp=gp,
+                 name=paste(name, ii, sep=""),
+                 vp=viewport(layout.pos.row=vp.ind[ii, 1],
+                             layout.pos.col=vp.ind[ii, 2]))
+   }
+
+   makeOneLabel <- textii(d=content, gp=gpar(col="blue"))
+
+   lg <- lapply(seq_along(content), makeOneLabel)
+
+   list(lg=lg, nrow=nrow(d), ncol=ncol(d))
+ }
> makeContentInVp2 <- function(d){
+   content <- as.character(unlist(c(d)))
+   nc <- ncol(d)
+   nr <- nrow(d)
+
+   n2nm <- function(nr, nc){
+     expand.grid(seq(1, nr), seq(1, nc))
+   }
+
+   vp.ind <- n2nm(nr, nc)
+   ## here we edit the viewport of all grobs
+   editVp <- function(glist){
+
+     for(ii in seq_along(glist))
+       glist[[ii]] <- editGrob(glist[[ii]],
+                               vp=viewport(layout.pos.row=vp.ind[ii, 1],
+                                           layout.pos.col=vp.ind[ii, 2]))
+
+     glist
+   }
+
+   textii <- function(d, gp=gpar(), name="content-label-"){
+     function(ii)

```

```

+       textGrob(label=d[ii], gp=gp,
+               name=paste(name, ii, sep=""))
+   }
+
+   makeOneLabel <- textii(d=content, gp=gpar(col="blue"))
+
+   lg <- lapply(seq_along(content), makeOneLabel)
+   lg <- editVp(lg)
+
+   list(lg=lg, nrow=nrow(d), ncol=ncol(d))
+ }
> summary(content <- makeContent(d))

```

```

      Length Class  Mode
lg      30      -none- list
nrow    1      -none- numeric
ncol    1      -none- numeric

```

```

> summary(content2 <- makeContentInVp(d))

```

```

      Length Class  Mode
lg      30      -none- list
nrow    1      -none- numeric
ncol    1      -none- numeric

```

We can evaluate the relative performance of `makeContentInVp` and `makeContentInVp2`,

```

> ## profiling the creation of grob lists
> ## replicate 5 times the iris dataset
> d <- do.call(rbind, lapply(1:reps, function(ii) dlong))
> system.time(makeContentInVp(d))

```

```

  user  system elapsed
1.016   0.008   1.024

```

```

> system.time(makeContentInVp2(d))

```

```

  user  system elapsed
1.260   0.000   1.259

```

No big difference so far.

3 Functions that create a table from a list of grobs

- `table1` uses `frameGrob` and `packGrob`
- `table2` uses `frameGrob` but calculates the sizes manually and uses `placeGrob`
- `table3` creates a `grid.layout` and draws the grobs in the different viewports.
- `table4` creates a `grid.layout` and draws grobs that had a previously specified viewport.

```

> ## table1 uses grid.pack
> table1 <- function(content){
+
+   gcells = frameGrob(name="table.cells",
+     layout = grid.layout(content$nrow, content$ncol))
+
+   label.ind <- 1   # index running accross labels
+
+   for (ii in seq(1, content$ncol, 1)) {
+     for (jj in seq(1, content$nrow, 1)) {
+       gcells = packGrob(gcells, content$lg[[label.ind]], row=jj, col=ii, dynamic=TRUE,
+       label.ind <- label.ind + 1
+     }
+   }
+   grid.draw(gTree(children=gList(gcells)))
+ }

> ## table2 uses grid.place
> table2 <- function(content){
+
+   padding <- unit(4, "mm")
+   lg <- content$lg
+   ## retrieve the widths and heights of all textGrobs
+   wg <- lapply(lg, grobWidth) # list of grob widths
+   hg <- lapply(lg, grobHeight) # list of grob heights
+
+   ## concatenate this units
+   widths.all <- do.call(unit.c, wg) # all grob widths
+   heights.all <- do.call(unit.c, hg) #all grob heights
+
+   ## matrix-like operations on units to define the table layout
+   widths <- colMax.units(widths.all, content$ncol) # all column widths
+   heights <- rowMax.units(heights.all, content$nrow) # all row heights
+
+   gcells = frameGrob(name="table.cells",
+     layout = grid.layout(content$nrow, content$ncol,
+       width=widths+padding, height=heights+padding))
+
+   label.ind <- 1   # index running accross labels
+
+   for (ii in seq(1, content$ncol, 1)) {
+     for (jj in seq(1, content$nrow, 1)) {
+       gcells = placeGrob(gcells, content$lg[[label.ind]], row=jj, col=ii)
+       label.ind <- label.ind + 1
+     }
+   }
+   grid.draw(gTree(children=gList(gcells)))
+ }

> ## table3 uses grid.layout
> table3 <- function(content){

```

```

+
+ padding <- unit(4, "mm")
+ lg <- content$lg
+ ## retrieve the widths and heights of all textGrobs
+ wg <- lapply(lg, grobWidth) # list of grob widths
+ hg <- lapply(lg, grobHeight) # list of grob heights
+
+ ## concatenate this units
+ widths.all <- do.call(unit.c, wg) # all grob widths
+ heights.all <- do.call(unit.c, hg) #all grob heights
+
+ ## matrix-like operations on units to define the table layout
+ widths <- colMax.units(widths.all, content$ncol) # all column widths
+ heights <- rowMax.units(heights.all, content$nrow) # all row heights
+
+ cells = viewport(name="table.cells", layout =
+   grid.layout(content$nrow, content$ncol,
+     width=widths+padding, height=heights+padding) )
+
+ pushViewport(cells)
+
+ label.ind <- 1 # index running accross labels
+
+ ## loop over columns and rows
+ for (ii in seq(1, content$ncol, 1)) {
+   for (jj in seq(1, content$nrow, 1)) {
+     ## push a viewport for cell (ii,jj)
+     pushViewport(vp=viewport( layout.pos.row=jj, layout.pos.col=ii))
+     grid.draw( lg[[label.ind]]) # draw the text
+     upViewport()
+
+     label.ind <- label.ind + 1
+   }
+ }
+ upViewport()
+ }

> ## table4 uses grobs that already have a viewport assigned
> table4 <- function(content){
+
+ padding <- unit(4, "mm")
+ lg <- content$lg
+ ## retrieve the widths and heights of all textGrobs
+ wg <- lapply(lg, grobWidth) # list of grob widths
+ hg <- lapply(lg, grobHeight) # list of grob heights
+
+ ## concatenate this units
+ widths.all <- do.call(unit.c, wg) # all grob widths
+ heights.all <- do.call(unit.c, hg) #all grob heights
+
+ ## matrix-like operations on units to define the table layout

```

```

+ widths <- colMax.units(widths.all, content$ncol) # all column widths
+ heights <- rowMax.units(heights.all, content$nrow) # all row heights
+
+ vp <- viewport(layout=grid.layout(content$nrow, content$ncol,
+                                w=widths+padding, h=heights+padding))
+
+ grid.draw(gTree(children=do.call(gList, lg), vp=vp))
+
+ }

```

```

5.1  3.5  1.4  0.2  1
4.9   3   1.4  0.2  1
4.7  3.2  1.3  0.2  1
4.6  3.1  1.5  0.2  1
 5   3.6  1.4  0.2  1
5.4  3.9  1.7  0.4  1

```

table1(content)

```

5.1  3.5  1.4  0.2  1
4.9   3   1.4  0.2  1
4.7  3.2  1.3  0.2  1
4.6  3.1  1.5  0.2  1
 5   3.6  1.4  0.2  1
5.4  3.9  1.7  0.4  1

```

table2(content)

```

5.1  3.5  1.4  0.2  1
4.9   3   1.4  0.2  1
4.7  3.2  1.3  0.2  1
4.6  3.1  1.5  0.2  1
 5   3.6  1.4  0.2  1
5.4  3.9  1.7  0.4  1

```

table3(content)

```

5.1  3.5  1.4  0.2  1
4.9   3   1.4  0.2  1
4.7  3.2  1.3  0.2  1
4.6  3.1  1.5  0.2  1
 5   3.6  1.4  0.2  1
5.4  3.9  1.7  0.4  1

```

table4(content2)

```

> d <- dlong
> content <- makeContent(d)
> content2 <- makeContentInVp(d)
> pdf("test-timing-iris.pdf", height=45)
> system.time(table1(content))

```

```

      user  system elapsed
6.693    0.032    6.726

> grid.newpage()
> system.time(table2(content))

      user  system elapsed
0.968    0.020    0.989

> grid.newpage()
> system.time(table3(content))

      user  system elapsed
0.628    0.000    0.627

> grid.newpage()
> system.time(table4(content2))

      user  system elapsed
0.84     0.00     0.84

> dev.off()

```

pdf
2