An introduction to grid graphics

John W. Emerson (Jay)

*Department of Statistics*

*Yale University*

*john.emerson@yale.edu*

May, 2014
Outline

1. Grid Graphics Programming

Jay Emerson (john.emerson@yale.edu)
Example 1 (sorry, a little fuzzy)

From http://epi2012.yale.edu/dataexplorer/countryprofiles:
Example 2 (via package gpairs)
About **grid** graphics

- You’ve got it! Package **grid** comes with R.
- You may have used it! Packages **lattice** and **ggplot2**, for example, both use it.
- You may not need to use it directly! See above.
- If you do... I’ll help get you started today.
- Paul Murrell wrote **grid**. See
  
  http://www.stat.auckland.ac.nz/~paul/

  for lots of other cool stuff, too.
But what is grid?

- a low-level graphics package, or system/language for producing graphics
- a set of tools to draw and arrange basic shapes

From Paul’s book (Murrell 2006): “… [grid provides fundamental tools] for drawing graphical scenes (including plots). There are basic features such as functions for drawing lines, rectangles, and text, together with more sophisticated and powerful concepts such as viewports, layouts, and units, which allow basic output to be located and sized in very flexible ways.”
A simple example

Hello world
A simple example: code

```r
library(grid)
grid.newpage()
grid.rect()
  pushViewport(viewport(width=0.9, height=0.5))
  grid.rect()
  grid.text("Hello world", 0.5, 0.5)
popViewport(1)
```
npc versus native units

- All arguments in the previous example (e.g. `height=0.5`) were in npc units (for normalized parent coordinates, often the default); 0 is bottom (or left) and 1 is top (or right) of the viewport.

- Other units are self-evident (e.g. cm or in).

- The most important other unit is native, akin (usually) to the scale of data values being plotted.

- Use the `unit()` function to explicitly specify the desired units.

- Can’t see something you intended to plot? Then grid probably expected npc units but you were thinking in native units.
An expanded example: code

library(grid)
grid.newpage()
  grid.rect()
  pushViewport(viewport(width=0.9, height=0.5,
                        xscale=c(0, 1000), yscale=c(1,10)))
    grid.rect()
    grid.text("Hello world", 0.5, 0.5)
  grid.points(unit(0.5, "npc"), unit(9, "native"))
  pushViewport(viewport(x=unit(200, "native"),
                        y=unit(0, "npc"),
                        width=unit(0.25, "npc"),
                        height=unit(2, "native"),
                        just=c("left", "bottom")))
    grid.rect()
    grid.points(0.5, 0.5, pch=2)
  popViewport(1)
grid.newpage()
popViewport(1)
An expanded example

Hello world
Other things to note

- Extra options are available via the `gp=` option; see help on function `gpar` for the `grid` version of `base` graphics `par`.

- See help on `grid.lines`, `grid.xaxis`, for example, and explore from there...

- Both `lattice` and `ggplot2` graphics are built on `grid`, and there is some ability (if you really know what you are doing or have a lot of patience) to customize them using `grid` directly.

- The `layout` argument or `grid.layout()` function are used in `grid` and *not* the `layout()` function or an argument like `par(mfrow=...)`. 
An example from Yale

From [http://www.stat.yale.edu/~jay/TheGame/](http://www.stat.yale.edu/~jay/TheGame/):

Yale Victory
Harvard Victory
Y Yale's Score in The Game
T Tie Game

Yale's Ivy League wins prior to The Game:
Harvard's Ivy League wins prior to The Game:

To be played on November 17, 2007
Final example (we’ll examine the code)