R Graphics

Paul Murrell

paul@stat.auckland.ac.nz

The University of Auckland
Overview

- Standard (base) R graphics
- grid graphics
  - Graphics Regions and Coordinate Systems
  - Directing Graphics Output
  - Producing Graphics Output
  - Plots from First Principles
- grid and lattice
R Graphics Fundamentals

- Graphics Regions and Coordinate Systems
  - Outer Margins
  - Figure Regions
  - Figure Margins
  - Plot Regions

- Directing Graphics Output
  - Which graphics functions to use

- Producing Graphics Output
  - Graphical parameters


```r
par(oma=c(0, 0, 0, 0), omi=)
par(mfrow=c(1, 1), mfcol=c(1, 1), fig=, fin=)
```
Arbitrary Figure Regions

```r
par(fig=c(0.1, 0.6, 0.1, 0.6))
par(new=T)
par(fig=c(0.4, 0.9, 0.4, 0.8))
```
par(mar=c(5.1, 4.1, 4.1, 2.1), mai=)
par(pty="m", pin=, plt=)
User Coordinates

\[
\text{The location } (x_i, y_i)
\]

\[
\text{par(xaxs="r", yaxs="r")}
\]

\[
\text{<plot.function>(..., xlim=, ylim=)}
\]
Figure Margin Coordinates
Outer Margin Coordinates

Outer Margin 1

Current Plot

0 lines

3 lines

0 1 11

0 0 11 1
# Directing Graphics Output

<table>
<thead>
<tr>
<th>Plot Region</th>
<th>Figure Margins</th>
<th>Outer Margins</th>
</tr>
</thead>
<tbody>
<tr>
<td>text()</td>
<td>mtext()</td>
<td>mtext()</td>
</tr>
<tr>
<td>points()</td>
<td>axis()</td>
<td></td>
</tr>
<tr>
<td>lines()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>arrows()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>polygon()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>segments()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>box()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>abline()</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Graphical Parameters

- **Permanent settings**
  \( \text{par(}<\text{param}>=) \)

- **Temporary settings**
  \( \langle\text{plot.function}\rangle(..., \ <\text{param}>=) \)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>col</strong></td>
<td>colour of lines, text, ...</td>
</tr>
<tr>
<td><strong>lwd</strong></td>
<td>line width</td>
</tr>
<tr>
<td><strong>lty</strong></td>
<td>line type</td>
</tr>
<tr>
<td><strong>font</strong></td>
<td>font face (plain, bold, italic)</td>
</tr>
<tr>
<td><strong>pch</strong></td>
<td>type of plotting symbol</td>
</tr>
<tr>
<td><strong>srt</strong></td>
<td>string rotation</td>
</tr>
</tbody>
</table>
Plots from First Principles
Plots from First Principles

Create regions and coordinate systems

```r
> par(omi=rep(0, 4), mar=c(5.1, 4.1, 4.1, 2.1),
   mfrow=c(1, 1))
> plot(0, type="n", xlim=c(0, 10), ylim=c(0,10),
      axes=F, xlab="", ylab="")
```

Draw data symbols in plot region

```r
> par(col=1, lty=1, lwd=1, cex=1, srt=0)
> points(1:10)
```

Draw axes and labels in the figure margins

```r
> box()
> axis(1)
> axis(2)
> mtext("1:10", side=2, line=3)
```
Plots from First Principles
Plots from First Principles

Create area for barplot, leaving room for legend.

```r
par(fig=c(0, 0.8, 0, 1), mar=c(4, 4, 4, 2))
```

Draw barplot.

```r
barplot(matrix(sample(1:4, 16, replace=T),
              ncol=4),
       angle=45, density=1:4*10, col=1)
```

Stay on same page and set up region and coordinates for legend.

```r
par(new=T)
par(fig=c(0.8, 1, 0, 1), mar=c(4, 0, 4, 2))
plot(0, xlim=c(0, 1), ylim=c(0, 5), axes=F,
     xlab="", ylab="", type="n")
```
Figure out what 0.5” is in user coordinates.

```r
size <- par("cxy")/par("cin")*.5
```

Draw legend elements and a dashed border.

```r
box(lty=2)
for (i in 1:4)
  polygon(c(0.5 - size[1]/2, 0.5 - size[1]/2, 0.5 + size[1]/2, 0.5 + size[1]/2),
          c(i, i + size[2], i + size[2], i),
          angle=45, density=i*10)
  text(0.5, i-0.2, paste("Group", i))
```
grid Graphics Fundamentals

- Graphics Regions and Coordinate Systems
  - Viewports
  - Layouts

- Directing Graphics Output
  - Units

- Producing Graphics Output
  - Graphical primitives and components
  - Graphical parameters
viewport(x = 0.5, y = 0.5, width = 0.5, height = 0.25, angle=45)
pushing and popping viewports

vp1 <- viewport(x=0, y=0.5, w=0.5, h=0.5, just=c("left", "bottom"))
vp2 <- viewport(x=0.5, y=0, w=0.5, h=0.5, just=c("left", "bottom"))
push.viewport(vp1)
grid.text("Some drawing in graphics region 1", y=0.8)
pop.viewport()
push.viewport(vp2)
grid.text("Some drawing in graphics region 2", y=0.8)
pop.viewport()
push.viewport(vp1)
grid.text("MORE drawing in graphics region 1", y=0.2)
pop.viewport()
Pushing and Popping Viewports

Some drawing in graphics region 1

MORE drawing in graphics region 1

Some drawing in graphics region 2
The Viewport Stack

vp <- viewport(width = 0.5, height = 0.5)
push.viewport(vp)
grid.rect(gp=gpar(col="grey"))
grid.text("quarter of the page",
         y=0.85)
push.viewport(vp)
grid.rect()
grid.text("quarter of the\nprevious viewport")
pop.viewport(2)
The Viewport Stack

quarter of the page

quarter of the previous viewport
push.viewport(
    viewport(y=unit(3, "lines"),
        width=0.9,
        height=0.8, just="bottom",
        xscale=c(0, 100))
)
grid.rect(gp=gpar(col="grey"))
grid.xaxis()
push.viewport(
    viewport(x=unit(60, "native"),
        y=unit(0.5, "npc"),
        width=unit(1, "strwidth",
            "coordinates for everyone"),
        height=unit(3, "inches")))
grid.rect()
grid.text("coordinates for everyone")
pop.viewport(2)
Directing Graphics Output

coordinates for everyone
<table>
<thead>
<tr>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;npc&quot;</td>
<td>Normalised Parent Coordinates. Treats the bottom-left corner of the current viewport as the location ((0, 0)) and the top-right corner as ((1, 1)).</td>
</tr>
<tr>
<td>&quot;native&quot;</td>
<td>Locations and sizes are relative to the x- and y-scales for the current viewport.</td>
</tr>
<tr>
<td>&quot;inches&quot;</td>
<td>Locations and sizes are in terms of physical inches. For locations, ((0, 0)) is at the bottom-left of the viewport.</td>
</tr>
<tr>
<td>&quot;cm&quot;</td>
<td>Same as &quot;inches&quot;, except in centimetres.</td>
</tr>
</tbody>
</table>
Units

"char" Locations and sizes are specified in terms of multiples of the current nominal fontheight.

"lines" Locations and sizes are specified in terms of multiples of the height of a line of text (dependent on both the current fontsize and the current lineheight).

"snpc" Square Normalised Parent Coordinates. Locations and size are expressed as a proportion of the smaller of the width and height of the current viewport.
# Units

<table>
<thead>
<tr>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;strwidth&quot;</td>
<td>Locations and sizes are expressed as multiples of the width of a given string (dependent on the string and the current fontsize).</td>
</tr>
<tr>
<td>&quot;strheight&quot;</td>
<td>Like &quot;strwidth&quot;.</td>
</tr>
<tr>
<td>&quot;grobwidth&quot;</td>
<td>Locations and sizes are expressed as multiples of the width of a given graphical object (dependent on the current state of the graphical object).</td>
</tr>
<tr>
<td>&quot;grobheight&quot;</td>
<td>Like &quot;grobwidth&quot;.</td>
</tr>
</tbody>
</table>
Working with Units

> unit(1, "npc")
[1] 1npc

> unit(1:3/4, "npc")
[1] 0.25npc 0.5npc 0.75npc

> unit(1:3/4, "npc")[2]
[1] 0.5npc
Working with Units

> unit(1:3/4, "npc") + unit(1, "inches")
[1] 0.25npc+1inches 0.5npc+1inches 0.75npc+1inches

> min(unit(0.5, "npc"), unit(1, "inches"))
x[1] min(0.5npc, 1inches)

> unit.c(unit(0.5, "npc"),
+ unit(2, "inches") + unit(1:3/4, "npc"),
+ unit(1, "strwidth", "hi there"))
[1] 0.5npc 2inches+0.25npc
[3] 2inches+0.5npc 2inches+0.75npc
[5] 1strwidth
push.viewport(viewport(layout=grid.layout(4, 5)))
grid.rect(gp=gpar(col="grey"))
grid.segments(c(1:4/5, rep(0, 3)),
  c(rep(0, 4), 1:3/4),
  c(1:4/5, rep(1, 3)),
  c(rep(1, 4), 1:3/4),
gp=gpar(col="grey"))
push.viewport(viewport(layout.pos.col=2:3,
  layout.pos.row=3))
grid.rect(gp=gpar(lwd=3))
pop.viewport(2)
Layouts
grid.layout(4, 4,
  widths=unit(c(3, 1, 1, 1),
              c("lines", "null", "null", "cm")),
  heights=unit(c(1, 1, 2, 3),
              c("cm", "null", "null", "lines")))
## Producing Graphics Output

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>grid.text</td>
<td>Can specify angle of rotation.</td>
</tr>
<tr>
<td>grid.rect</td>
<td></td>
</tr>
<tr>
<td>grid.circle</td>
<td></td>
</tr>
<tr>
<td>grid.polygon</td>
<td></td>
</tr>
<tr>
<td>grid.points</td>
<td>Can specify type of plotting symbol.</td>
</tr>
<tr>
<td>grid.lines</td>
<td></td>
</tr>
<tr>
<td>grid.segments</td>
<td>Convenience function for drawing grid lines.</td>
</tr>
<tr>
<td>grid.grill</td>
<td></td>
</tr>
<tr>
<td>grid.move.to</td>
<td></td>
</tr>
<tr>
<td>grid.line.to</td>
<td></td>
</tr>
<tr>
<td>grid.xaxis</td>
<td>Top or bottom axis</td>
</tr>
<tr>
<td>grid.yaxis</td>
<td>Left or right axis</td>
</tr>
</tbody>
</table>
Graphical Parameters

Specify using `gp` argument of viewport or graphical object.

Viewport settings are “inherited” by subsequent viewports and graphical objects.

---

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>col</code></td>
<td>colour of lines, text, ...</td>
</tr>
<tr>
<td><code>fill</code></td>
<td>colour for filling polygons, ...</td>
</tr>
<tr>
<td><code>lwd</code></td>
<td>line width</td>
</tr>
<tr>
<td><code>lty</code></td>
<td>line type</td>
</tr>
<tr>
<td><code>fontface</code></td>
<td>font face (plain, bold, italic)</td>
</tr>
<tr>
<td><code>fontfamily</code></td>
<td>font family (Helvetica, Hershey, ...)</td>
</tr>
<tr>
<td><code>fontsize</code></td>
<td>font size (points)</td>
</tr>
</tbody>
</table>
Graphical Parameters

push.viewport(
  viewport(gp=gpar(fill="grey",
              fontface="italic")))

grid.rect()
grid.rect(width=0.8, height=0.6,
          gp=gpar(fill="white"))

grid.text("This text and the inner rectangle
          have specified their own gpar settings",
            y=0.75, gp=gpar(fontface="plain"))

grid.text("This text and the outer rectangle
          accept the gpar settings of the viewport",
            y=0.25)

pop.viewport()
Graphical Parameters

This text and the inner rectangle have specified their own gpar settings

This text and the outer rectangle accept the gpar settings of the viewport
Plots from First Principles

```r
x <- y <- 1:10
push.viewport(plotViewport(c(5.1, 4.1, 4.1, 2.1)))
push.viewport(dataViewport(x, y))
grid.rect()
grid.xaxis()
grid.yaxis()
grid.points(x, y)
grid.text("1:10", x=unit(-3, "lines"), rot=90)
pop.viewport(2)
```
bp <- function(barData) {
    nbars <- dim(barData)[2]
    nmeasures <- dim(barData)[1]
    barTotals <- rbind(rep(0, nbars), apply(barData, 2, cumsum))
    barYscale <- c(0, max(barTotals)*1.05)
    push.viewport(plotViewport(c(5, 4, 4, 1),
                               yscale=barYscale,
                               layout=grid.layout(1, nbars)))
    grid.rect()
    grid.yaxis()
    for (i in 1:nbars) {
        push.viewport(viewport(layout.pos.col=i, yscale=barYscale))
        grid.rect(x=rep(0.5, nmeasures),
                  y=unit(barTotals[1:nmeasures, i], "native"),
                  height=unit(diff(barTotals[,i]), "native"),
                  width=0.8, just="bottom", gp=gpar(fill=boxColours))
        pop.viewport()
    }
    pop.viewport()
}
Barplot with Legend

```r
leg <- function(legLabels) {
  nlabels <- length(legLabels)
  push.viewport(viewport(layout=grid.layout(4, 1)))
  for (i in 1:nlabels) {
    push.viewport(viewport(layout.pos.row=i))
    grid.rect(width=boxSize, height=boxSize, 
              just="bottom",
              gp=gpar(fill=boxColours[i]))
    grid.text(legLabels[i], 
              y=unit(0.5, "npc") - unit(1, "lines"))
    pop.viewport()
  }
  pop.viewport()
}
```
```
barData <- matrix(sample(1:4, 16, replace=T), ncol=4)
boxColours <- 1:4
legLabels <- c("Group A", "Group B", "Group C", "Something Longer")
boxSize <- unit(0.5, "inches")

legend.width <- max(unit(rep(1, length(legLabels)), "strwidth", as.list(legLabels)) +
                      unit(2, "lines"),
                      unit(0.5, "inches") + unit(2, "lines"))
push.viewport(viewport(layout=grid.layout(1, 2,
widths=unit.c(unit(1,"null"), legend.width))))
push.viewport(viewport(layout.pos.col=1))
bp(barData)
pop.viewport()
push.viewport(viewport(layout.pos.col=2))
push.viewport(plotViewport(c(5, 0, 4, 0)))
leg(legLabels)
pop.viewport(3)
```
Barplot with Legend

Group A
Group B
Group C
Something Longer
Adding grid to lattice

```r
x <- rnorm(100)
y <- rnorm(100)
g <- sample(1:8, 100, replace=T)

print.trellis(
  xyplot(y ~ x | g,
    panel=function(x, y) {
      panel.xyplot(x, y);
      grid.lines(unit(c(0, 1), "npc"),
                 unit(0, "native"),
                 gp=gpar(col="grey"))
    }))
```

Adding grid to lattice
Adding lattice to grid

someText <- "A panel of text\nproduced using\nraw grid code\nthat describes\nthe plot\nto the right."
latticePlot <- xyplot(y ~ x | g, layout=c(2, 4))
grid.rect(gp=gpar(lty="dashed"))
push.viewport(viewport(layout=grid.layout(1, 2,
widths=unit.c(unit(1, "strwidth", someText) + unit(2, "cm"),
        unit(1, "null")))))
push.viewport(viewport(layout.pos.col=1))
grid.rect(gp=gpar(fill="light grey"))
grid.text(someText, x=unit(1, "cm"),
        y=unit(1, "npc") - unit(1, "inches"),
        just=c("left", "top"))
pop.viewport()
push.viewport(viewport(layout.pos.col=2))
print.trellis(latticePlot, newpage=FALSE)
pop.viewport(2)
Adding lattice to grid

A panel of text produced using raw grid code that describes the plot to the right.