R Graphics

Paul Murrell

paul@stat.auckland.ac.nz

The University of Auckland

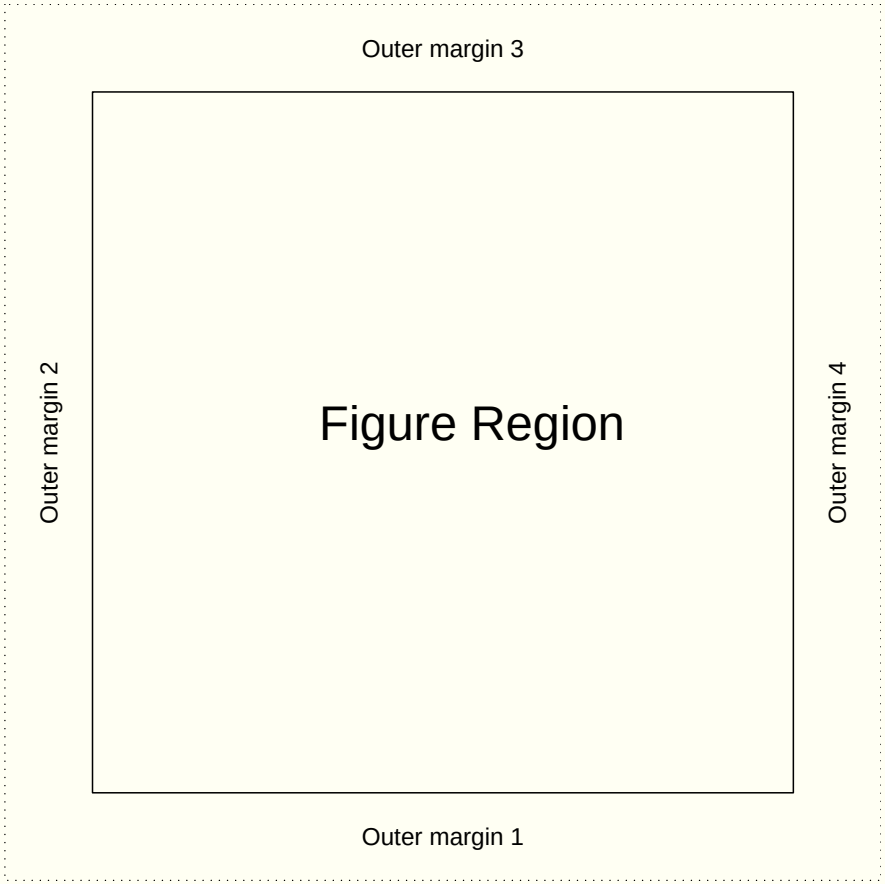
Overview

- 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

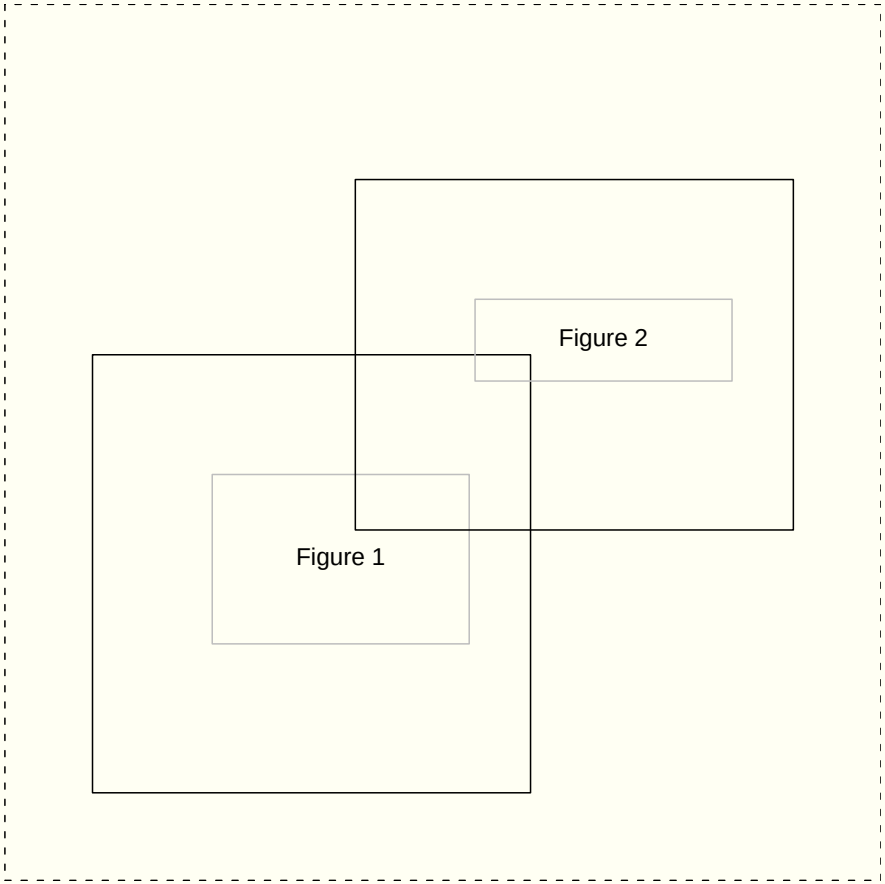
- Graphics Regions and Coordinate Systems
  - Outer Margins
  - Figure Regions
  - Figure Margins
  - Plot Regions
- Directing Graphics Output
  - Which graphics functions to use
- Producing Graphics Output
  - Graphical parameters

Outer Margins and Figure Region



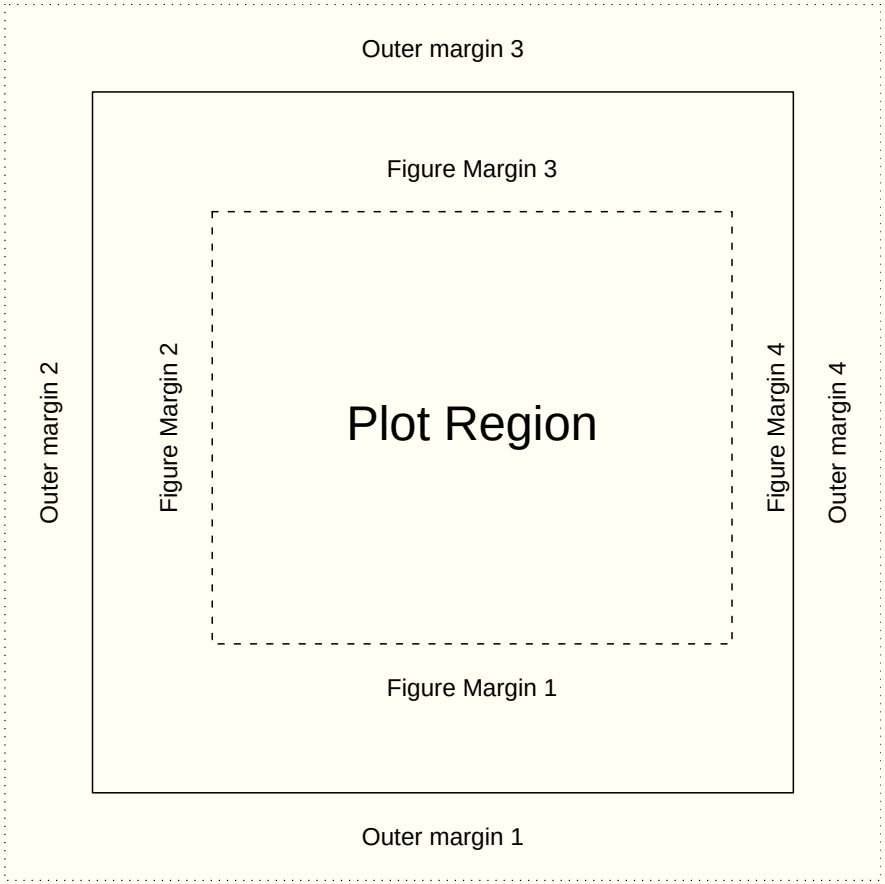

```
par(oma=c(0, 0, 0, 0), omi=)
par(mfrow=c(1, 1), mfcol=c(1, 1), fig=, fin=)
```

Arbitrary Figure Regions



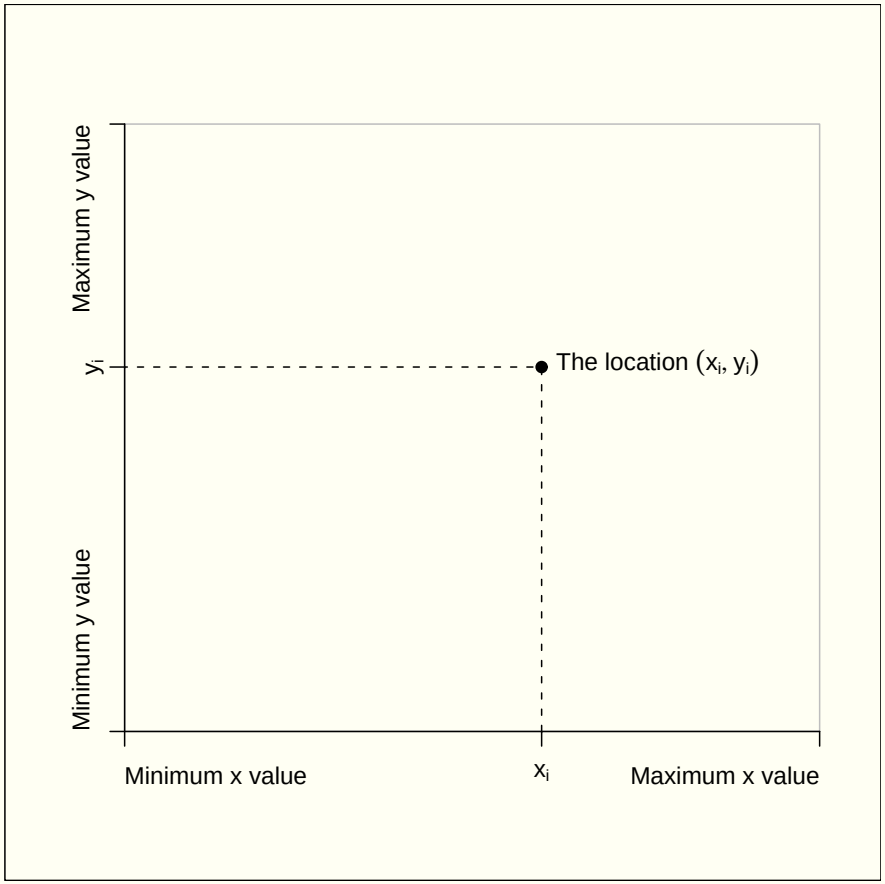

```
par(fig=c(0.1, 0.6, 0.1, 0.6))
par(new=T)
par(fig=c(0.4, 0.9, 0.4, 0.8))
```

Figure Margins and Plot Region



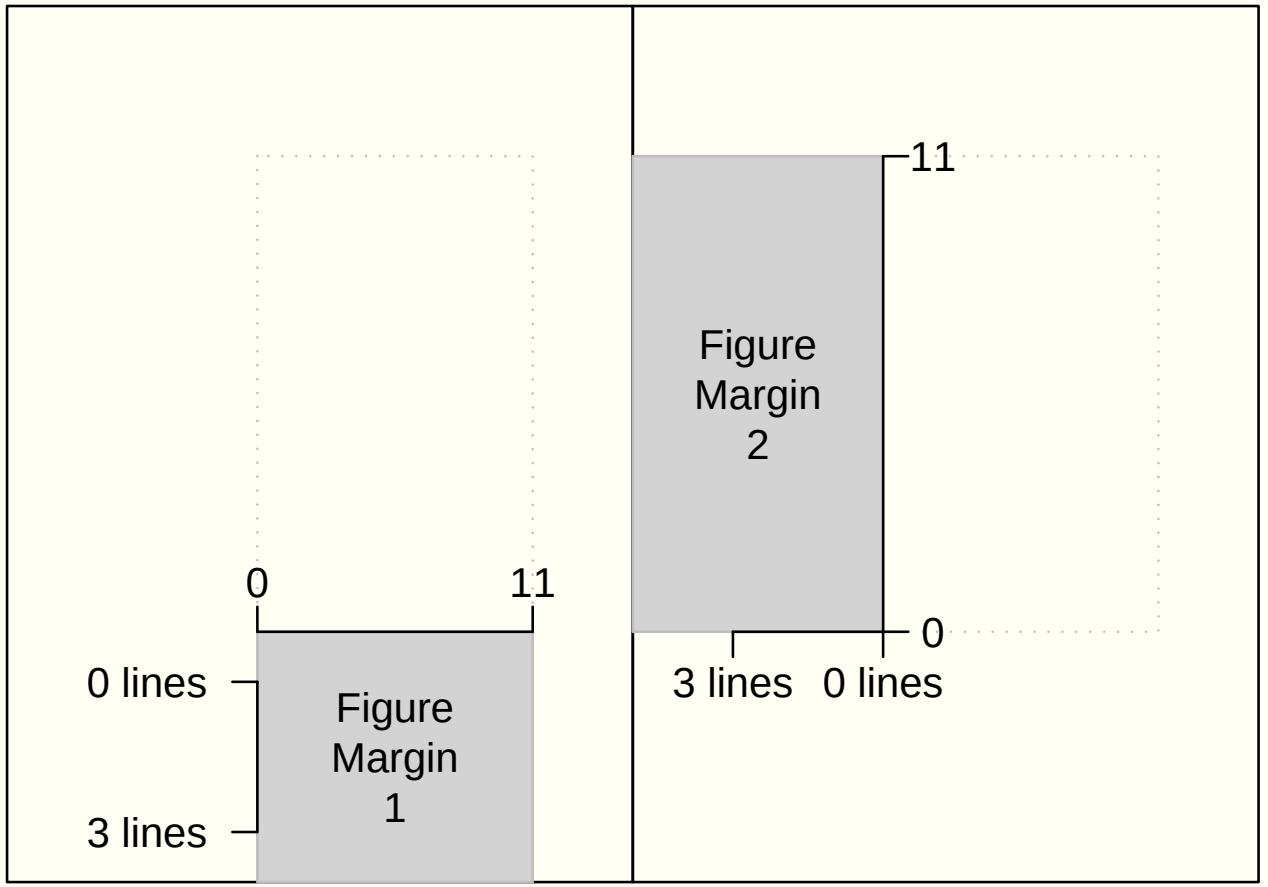

```
par(mar=c(5.1, 4.1, 4.1, 2.1), mai=)
par(pty="m", pin=, plt=)
```

User Coordinates

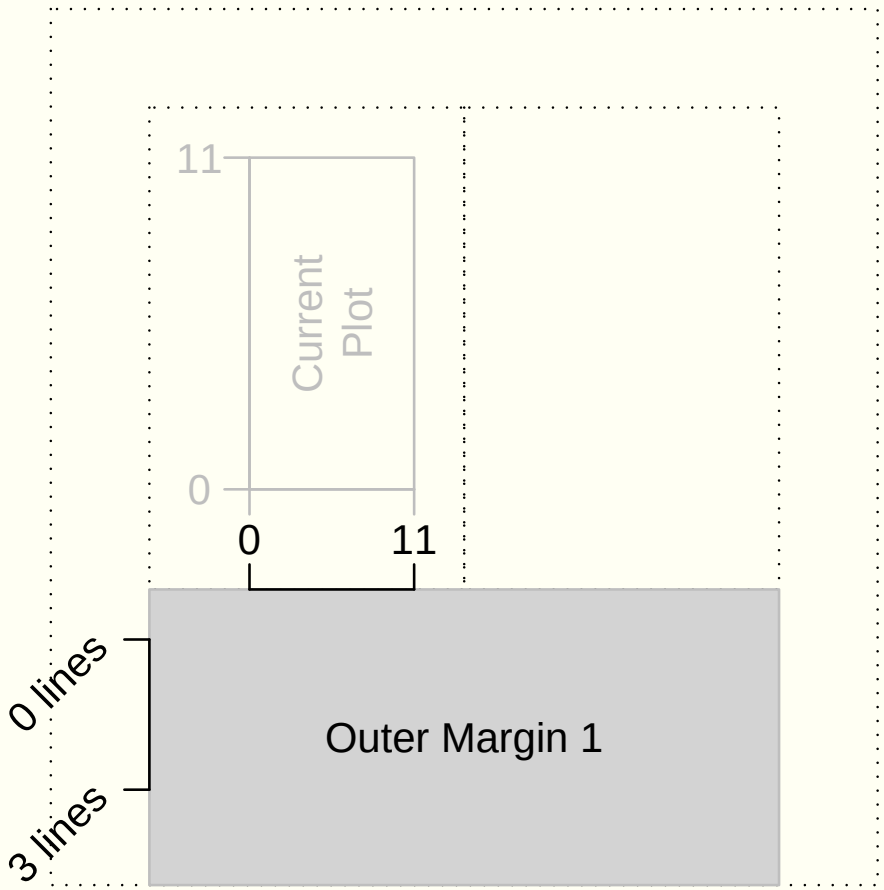


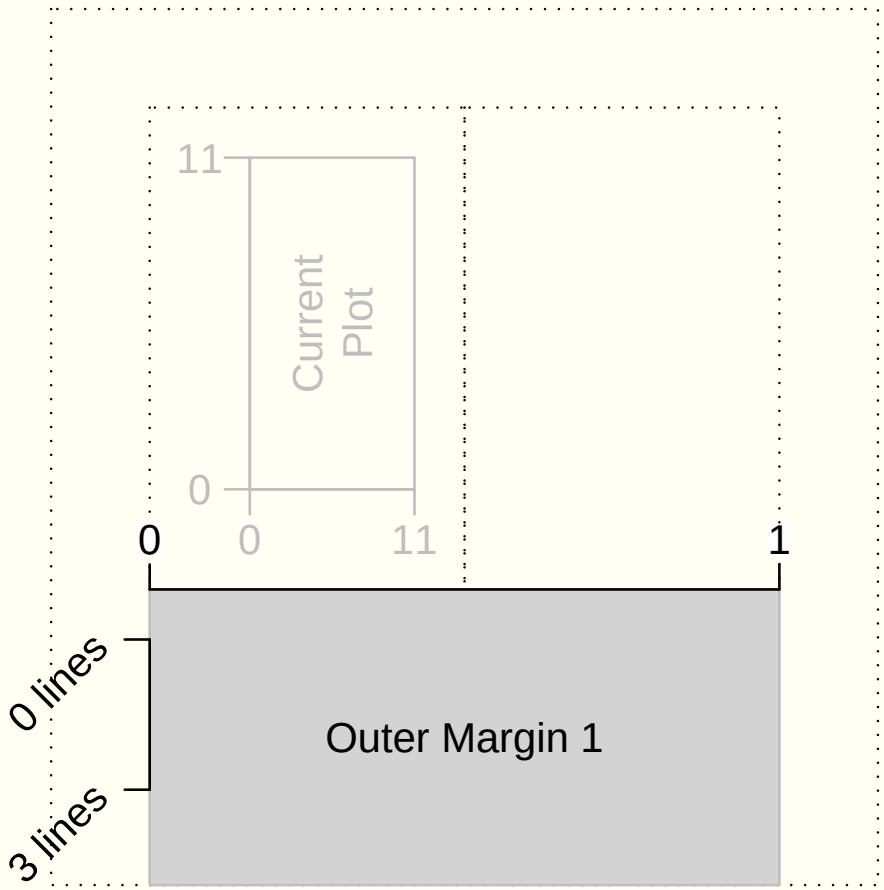

```
<plot.function>(..., xlim=, ylim=)
par(xaxs="r", yaxs="r")
```

Figure Margin Coordinates



Outer Margin Coordinates





Directing Graphics Output

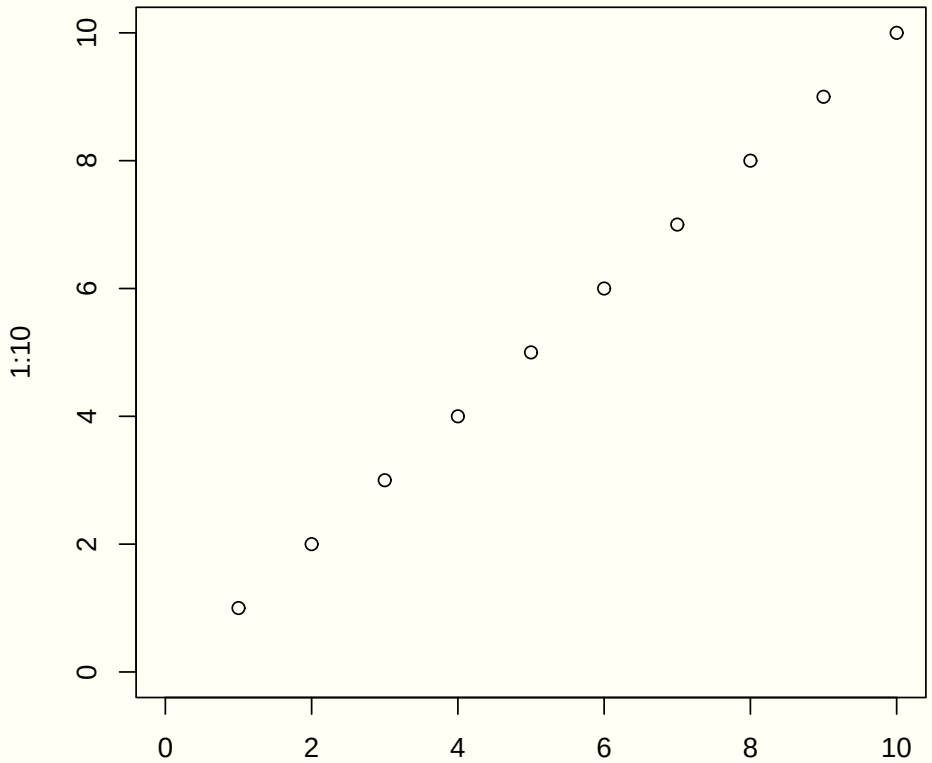
Plot Region	Figure Margins	Outer Margins
`text()`	`mtext()`	`mtext()`
`points()`	`axis()`	
`lines()`		
`arrows()`		
`polygon()`		
`segments()`		
`box()`		
`abline()`		

Graphical Parameters

- Permanent settings
  `par(<param>=)`
- Temporary settings
  `<plot.function>(..., <param>=)`

`col`	colour of lines, text, ...
`lwd`	line width
`lty`	line type
`font`	font face (plain, bold, italic)
`pch`	type of plotting symbol
`srt`	string rotation

Plots from First Principles



Plots from First Principles

- Create regions and coordinate systems

```
> 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="")
```

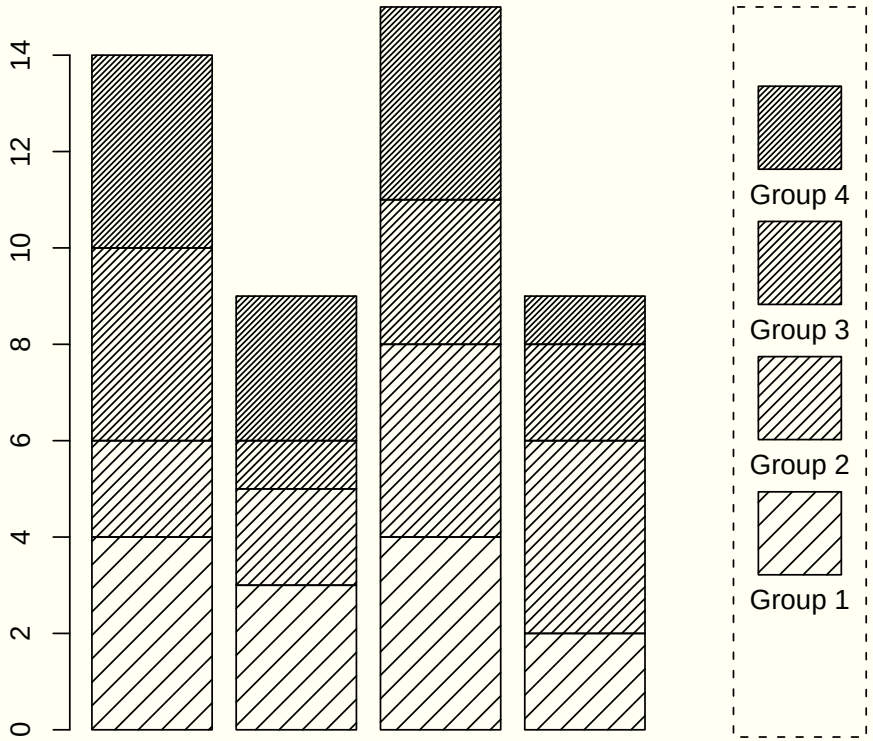
- Draw data symbols in plot region

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

- Draw axes and labels in the figure margins

```
> 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.

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

- Draw barplot.

```
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.

```
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")
```

Plots from First Principles

- Figure out what 0.5” is in user coordinates.

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

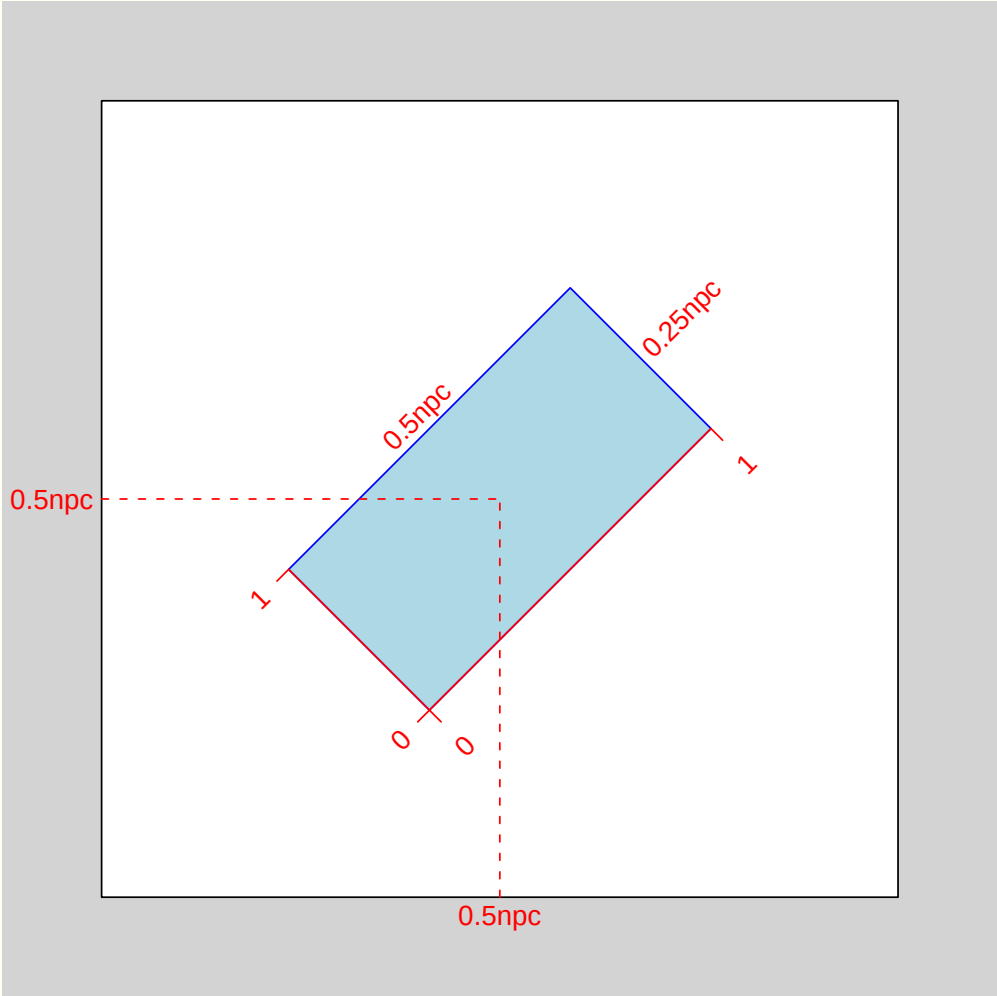
- Draw legend elements and a dashed border. .

```
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

Viewports

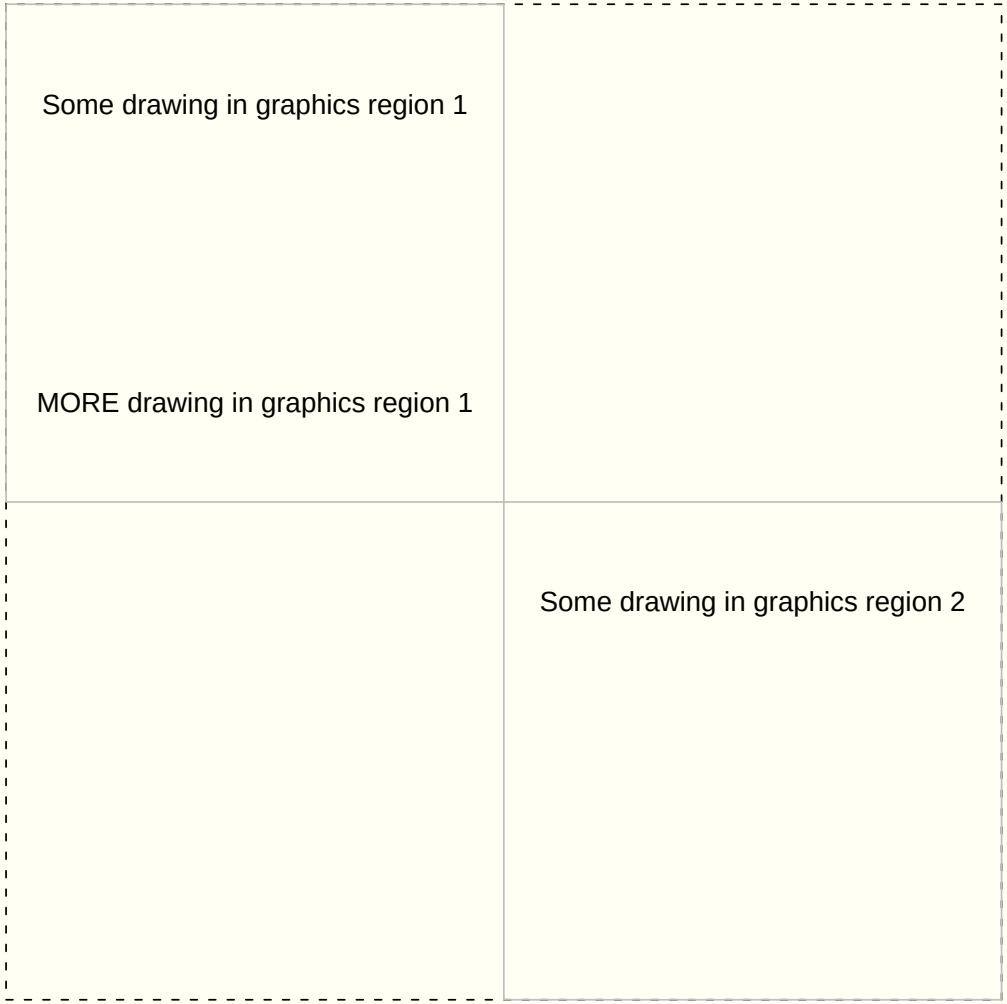



```
viewport(x = 0.5, y = 0.5, width = 0.5,
         height = 0.25, angle=45)
```

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()
```

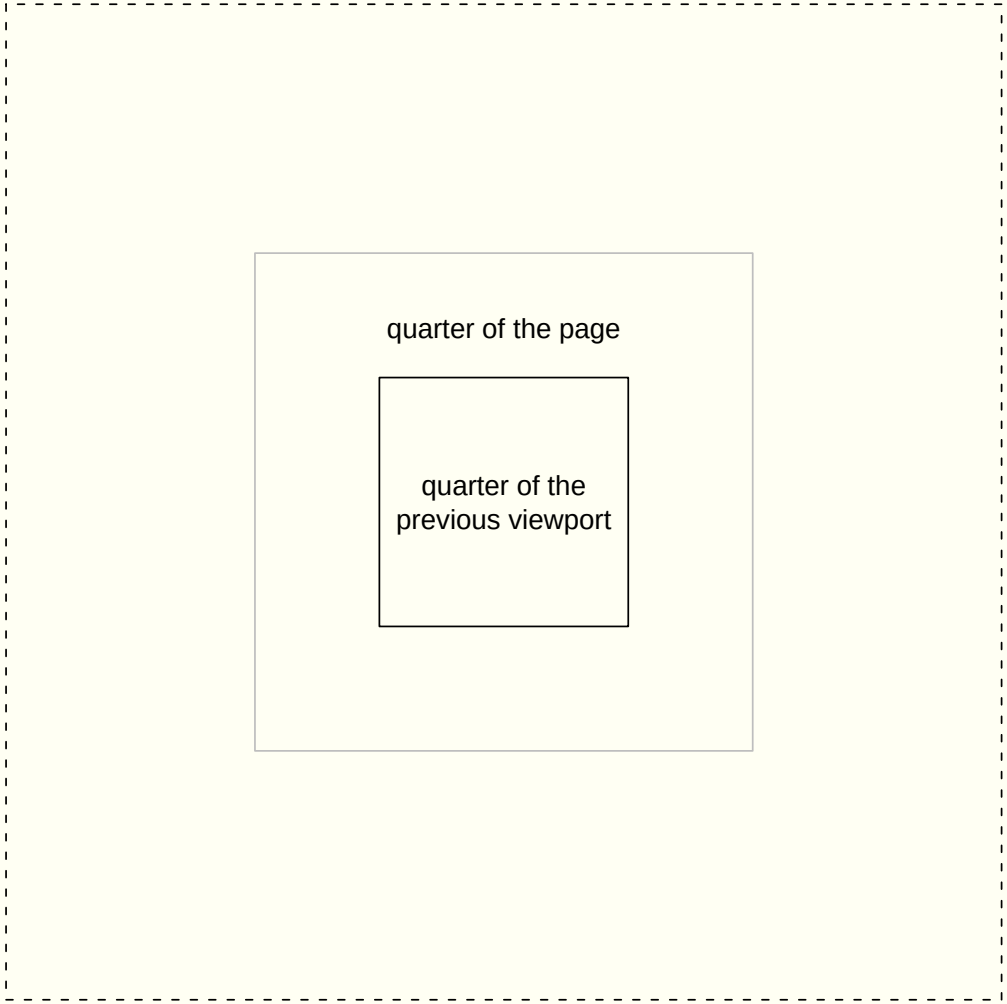
Pushing and Popping Viewports



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)
```

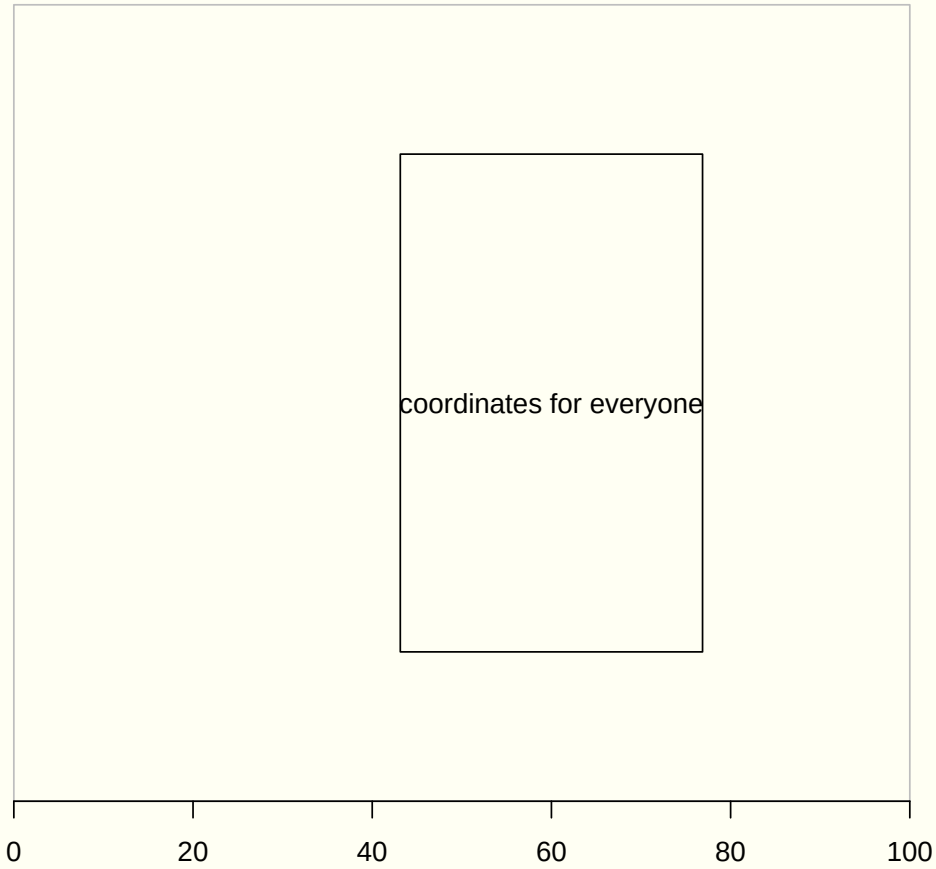
The Viewport Stack



Directing Graphics Output

```
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



Units

"npc"	Normalised Parent Coordinates. Treats the bottom-left corner of the current viewport as the location $(0, 0)$ and the top-right corner as $(1, 1)$.
"native"	Locations and sizes are relative to the x- and y-scales for the current viewport.
"inches"	Locations and sizes are in terms of physical inches. For locations, $(0, 0)$ is at the bottom-left of the viewport.
"cm"	Same as **"inches"**, except in centimetres.

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

`"strwidth"`	Locations and sizes are expressed as multiples of the width of a given string (dependent on the string and the current `fontsize`).
`"strheight"`	Like `"strwidth"`.
`"grobwidth"`	Locations and sizes are expressed as multiples of the width of a given graphical object (dependent on the current state of the graphical object).
`"grobheight"`	Like `"grobwidth"`.

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
```

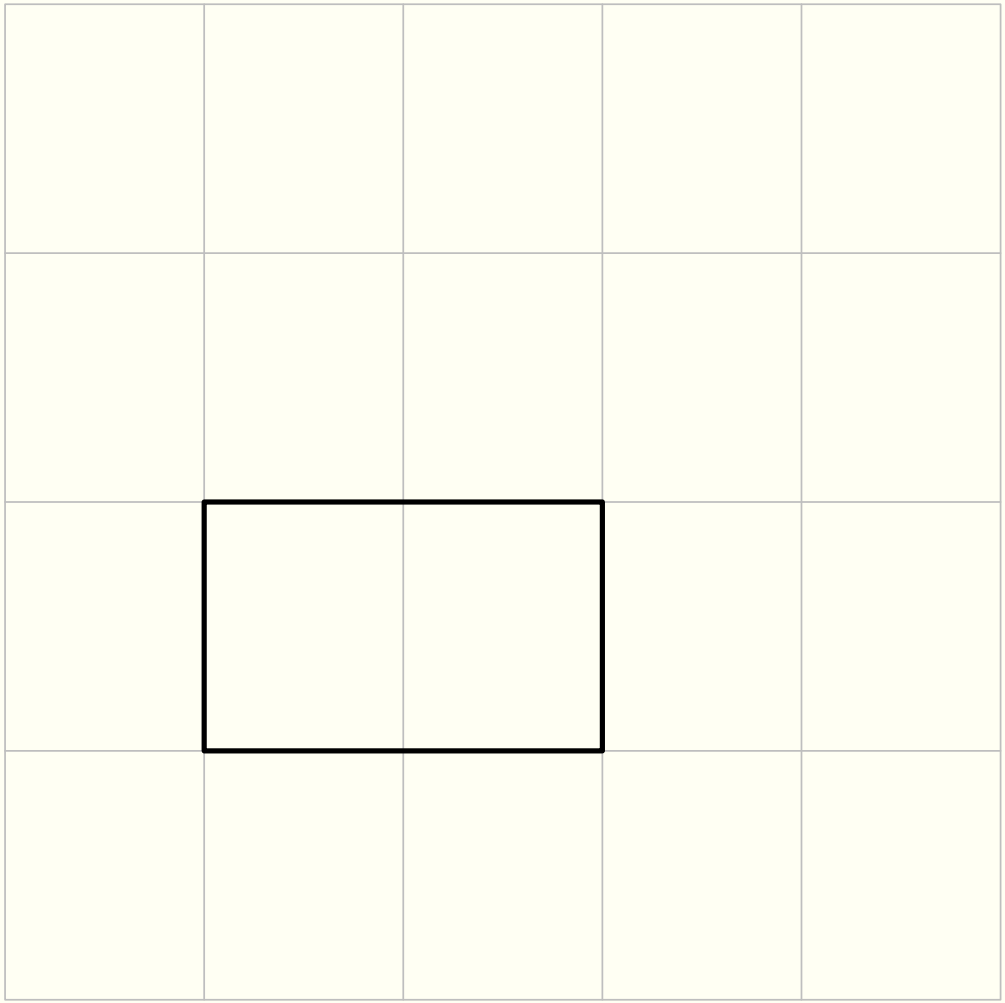
Working with Units

```
> 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
```

Layouts

```
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)
```

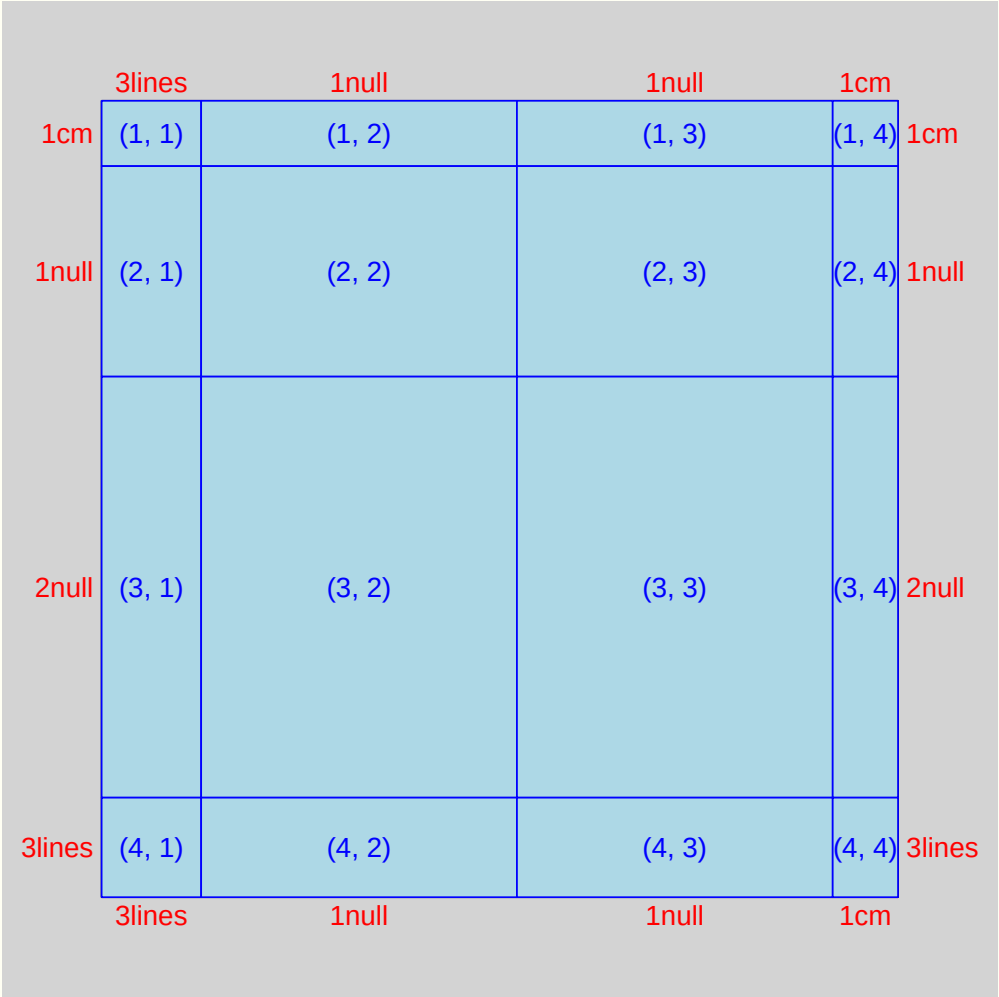
Layouts



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")))
```

Layouts


3lines
1null
1null
1cm
1cm
(1, 1)
(1, 2)
(1, 3)
(1, 4)
1cm
1null
(2, 1)
(2, 2)
(2, 3)
(2, 4)
1null
2null
(3, 1)
(3, 2)
(3, 3)
(3, 4)
2null
3lines
(4, 1)
(4, 2)
(4, 3)
(4, 4)
3lines
3lines
1null
1null
1cm

Producing Graphics Output

grid.text	Can specify angle of rotation.
grid.rect	
grid.circle	
grid.polygon	
grid.points	Can specify type of plotting symbol.
grid.lines	
grid.segments	
grid.grill	Convenience function for drawing grid lines
grid.move.to	
grid.line.to	
grid.xaxis	Top or bottom axis
grid.yaxis	Left or right axis

Graphical Parameters

- Specify using gp argument of viewport or graphical object.
- Viewport settings are “inherited” by subsequent viewports and graphical objects.

col	colour of lines, text, ...
fill	colour for filling polygons, ...
lwd	line width
lty	line type
fontface	font face (plain, bold, italic)
fontfamily	font family (Helvetica, Hershey, ...)
fontsize	font size (points)

Graphical Parameters

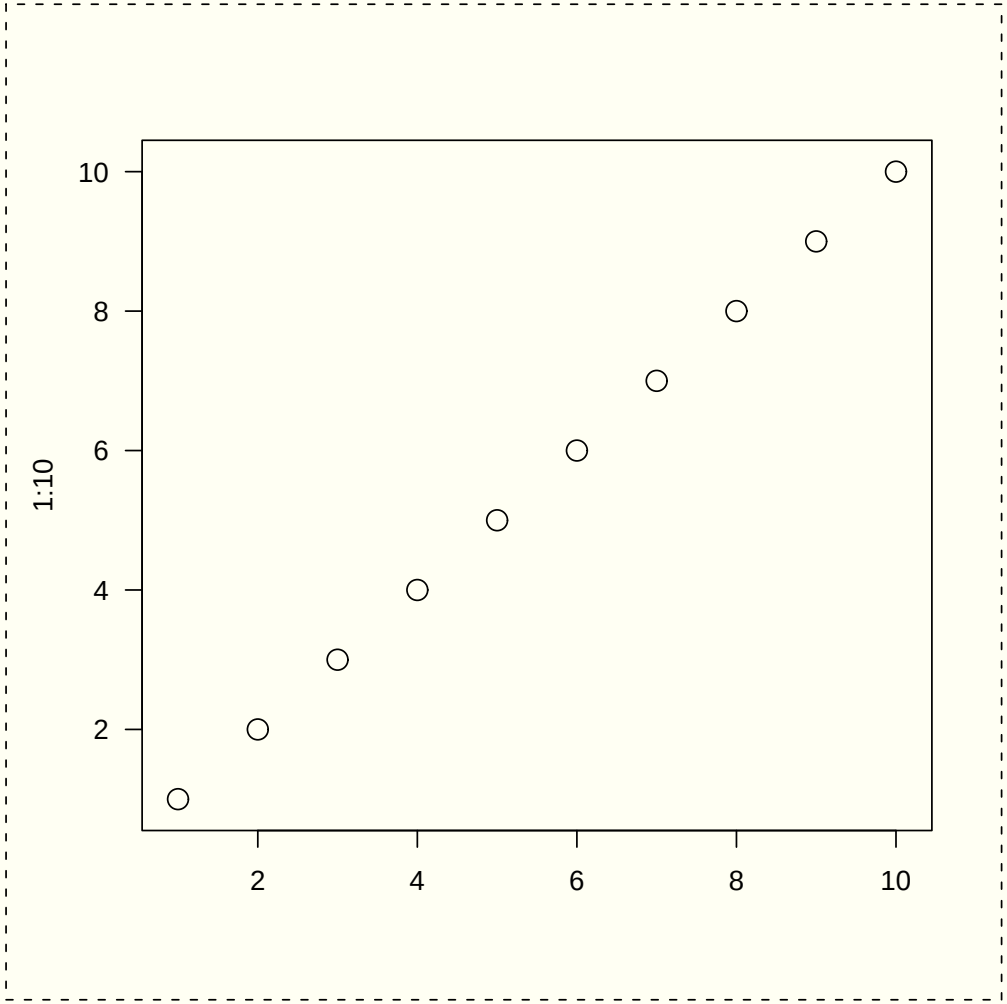
```
push.viewport(
  viewport(gp=gpar(fill="grey",
                   fontface="italic")))
grid.rect()
grid.rect(width=0.8, height=0.6,
          gp=gpar(fill="white"))
grid.text("This text and the inner rectangle\n
           have specified their own gpar settings",
          y=0.75, gp=gpar(fontface="plain"))
grid.text("This text and the outer rectangle\n
           accept the gpar settings of the viewport"
          y=0.25)
pop.viewport()
```

Graphical Parameters

Plots from First Principles

```
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)
```

Plots from First Principles



Barplot with Legend

```
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()
}
```

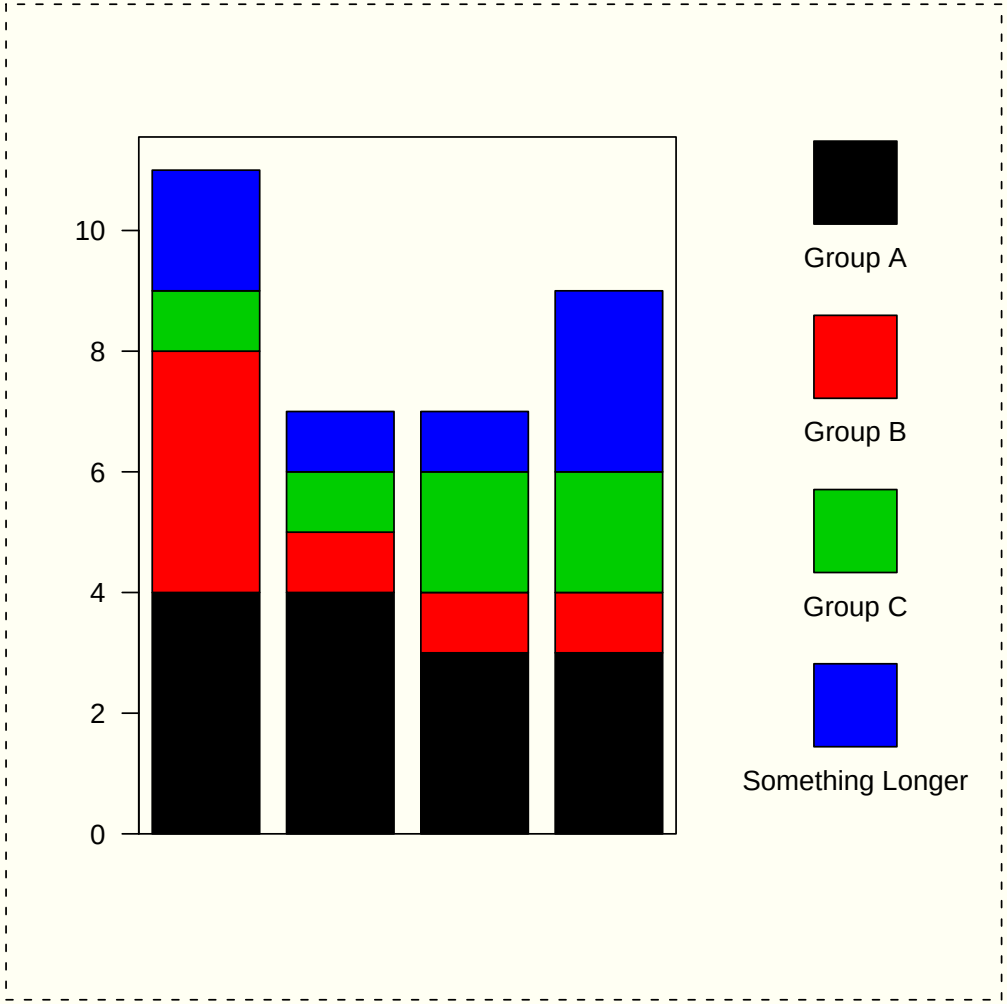
Barplot with Legend

```
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()
}
```

Barplot with Legend

```
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)
```

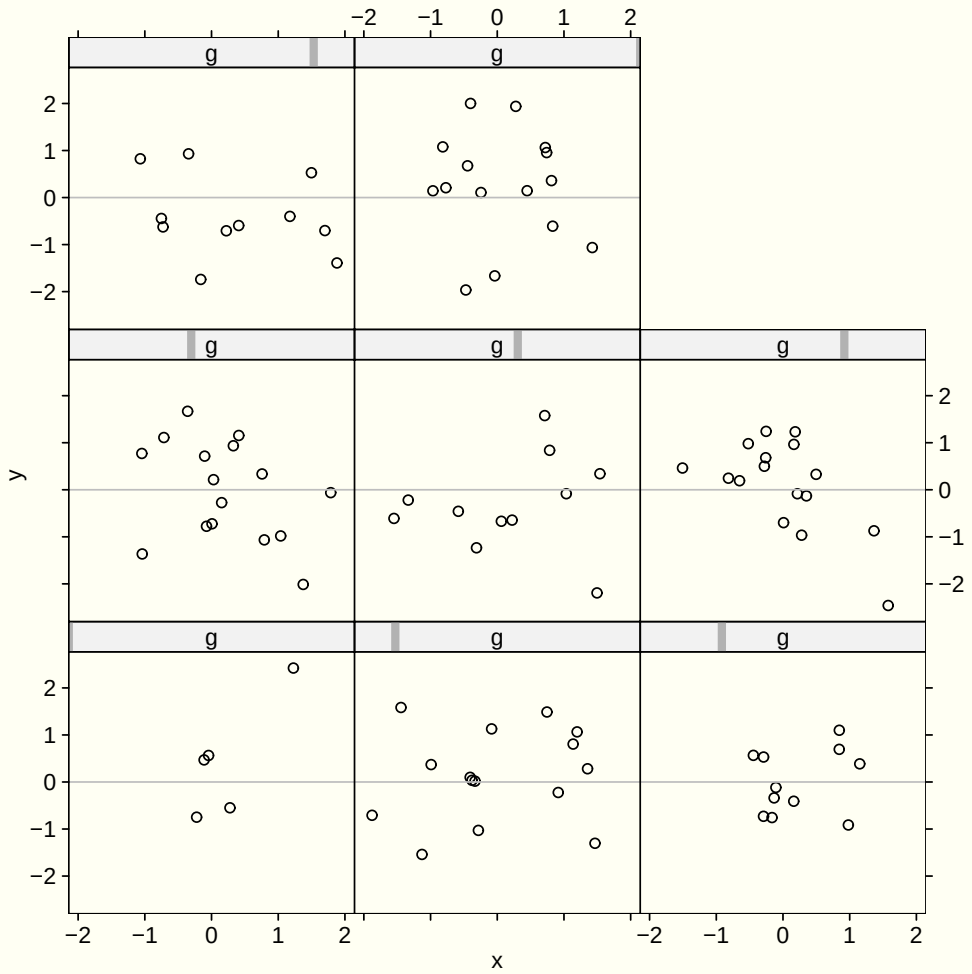
Barplot with Legend



Adding grid to lattice

```
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\n
             raw grid code\nthat describes\n
             the 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.