S Graphics

Paul Murrell

paul@stat.auckland.ac.nz

The University of Auckland

Overview

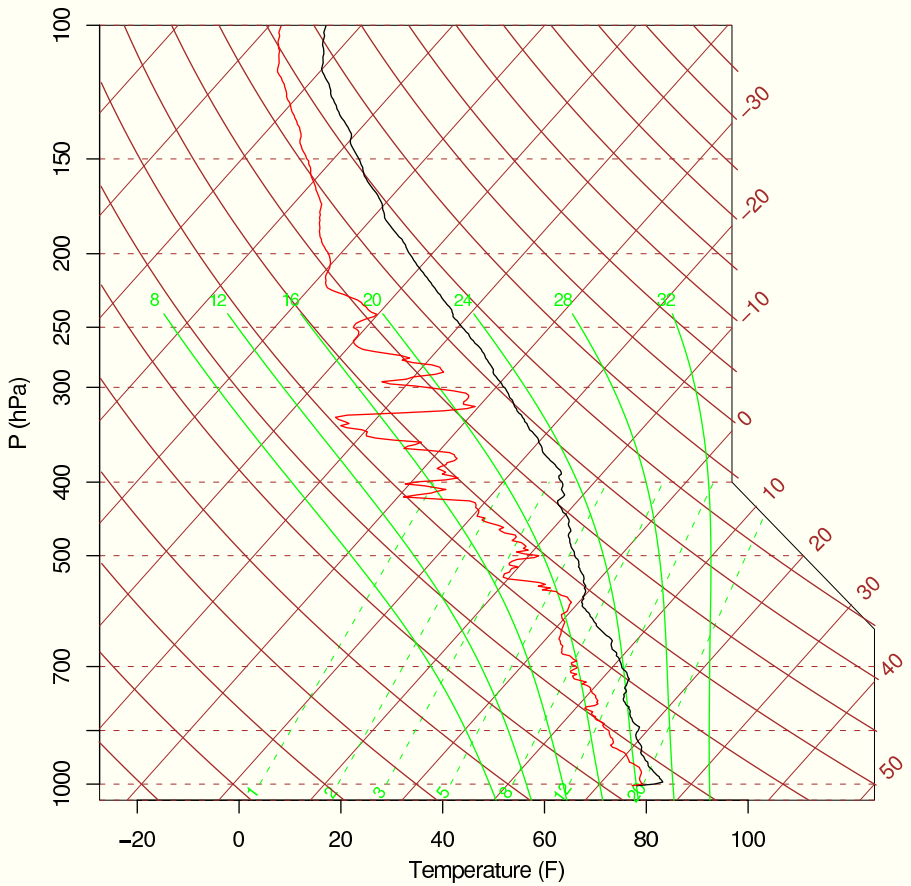
- Things that are covered
  - Command-line Graphics
  - Customising Traditional **S** Graphics
  - Customising Traditional Trellis Graphics

- Things that are not covered
  - GUI Graphics
  - Editable Graphics
  - Traditional High-Level Graphics functions

Overview

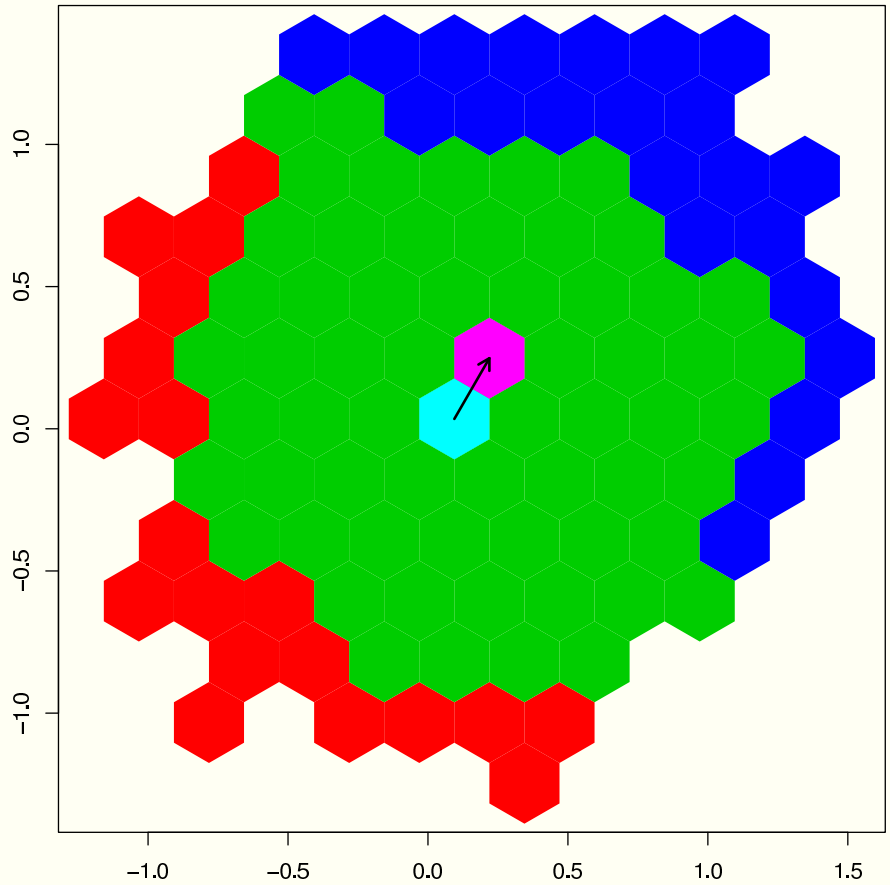
- Why customize?
  - Annotation
  - New Plots
  - Diagrams
- **S** Graphics Fundamentals
  - Graphics Regions and Coordinate Systems
  - Directing Graphics Output
  - Producing Graphics Output
- Adding to Existing Plots
- Plots from First Principles
- Trellis Graphics

Annotation



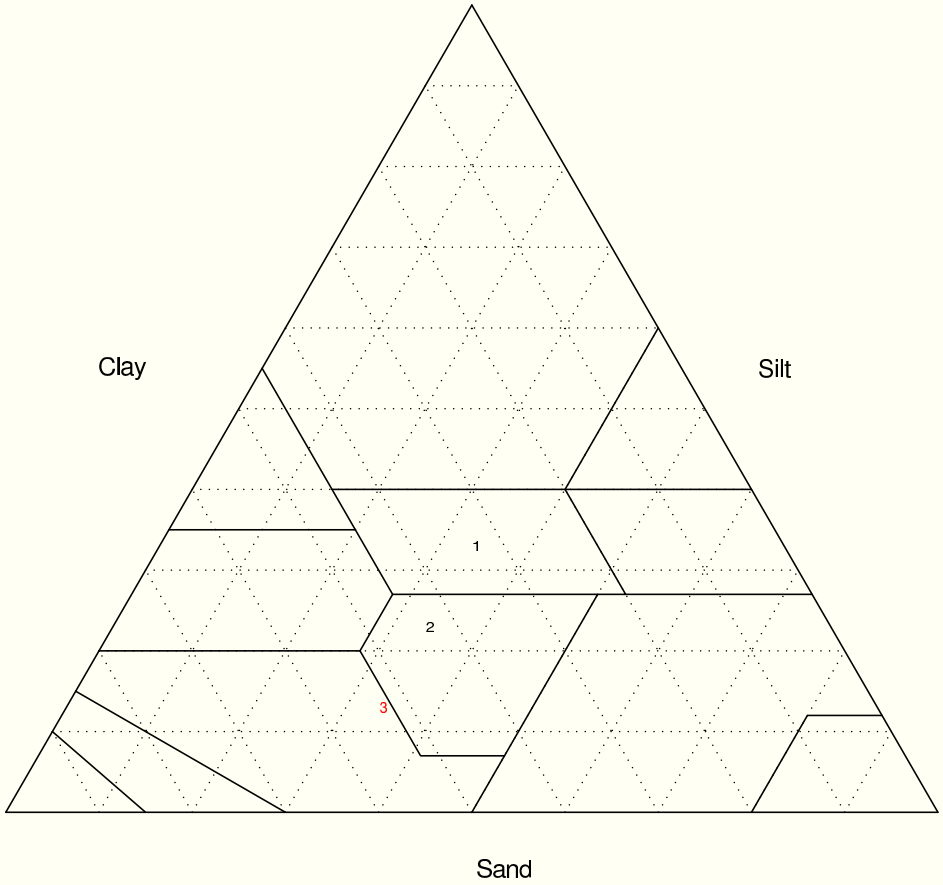
Tim Hoar, Eric Gilleland, Doug Nychka

New Plots - 1



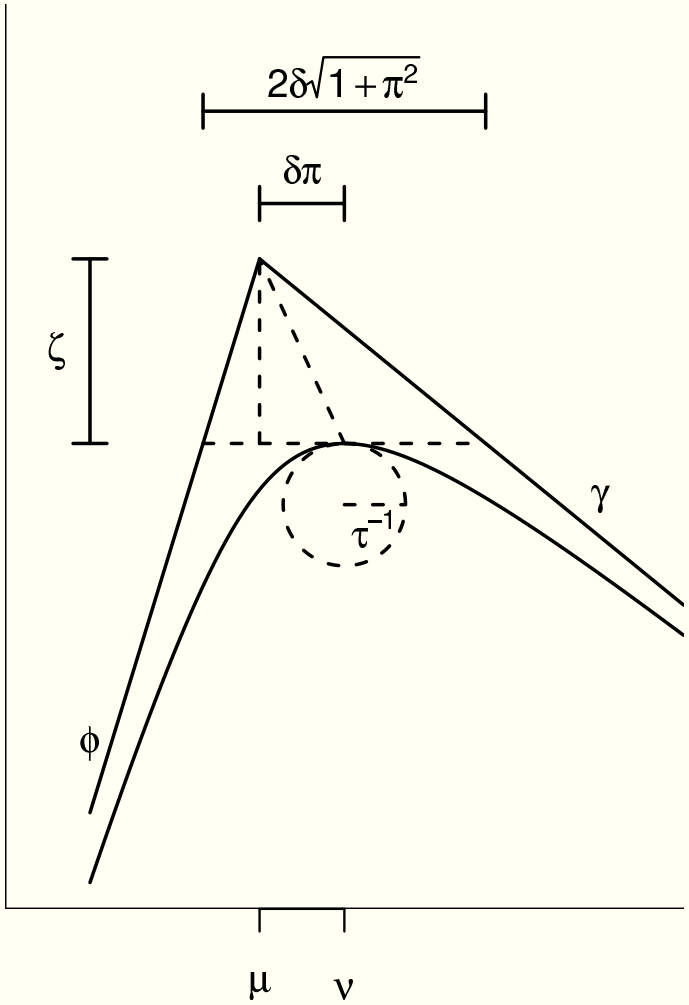
The Bioconductor Project

New Plots - 2



Corey A. Moffet, Texas Tech University

Diagrams

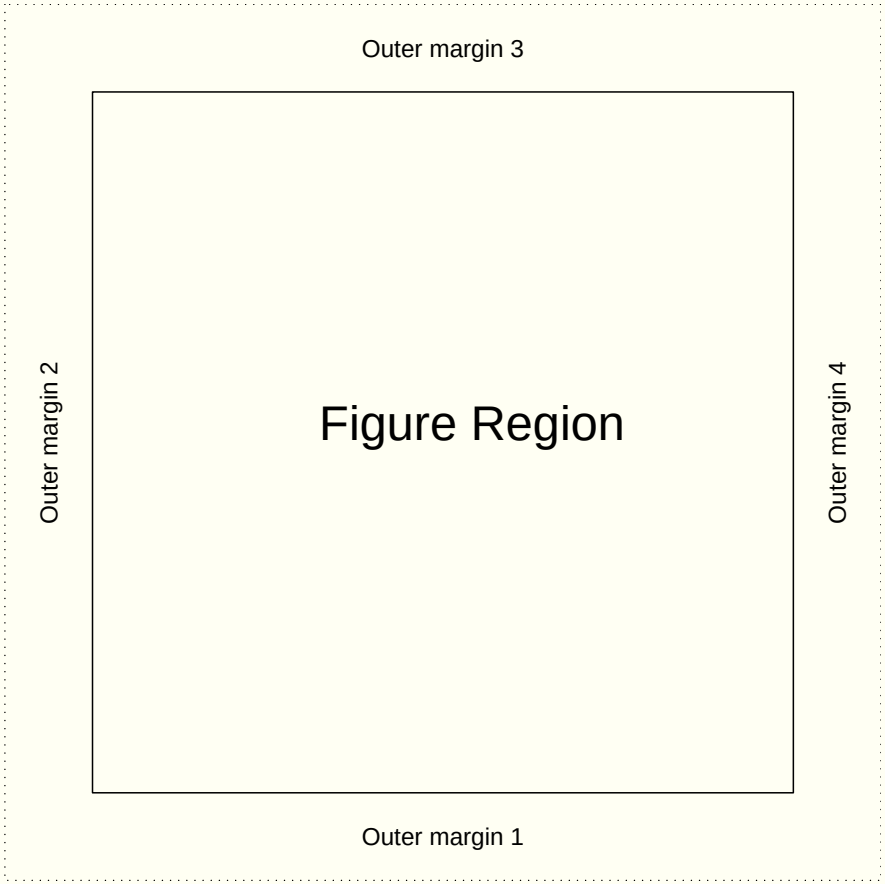


David Scott, The University of Auckland

S Graphics Fundamentals

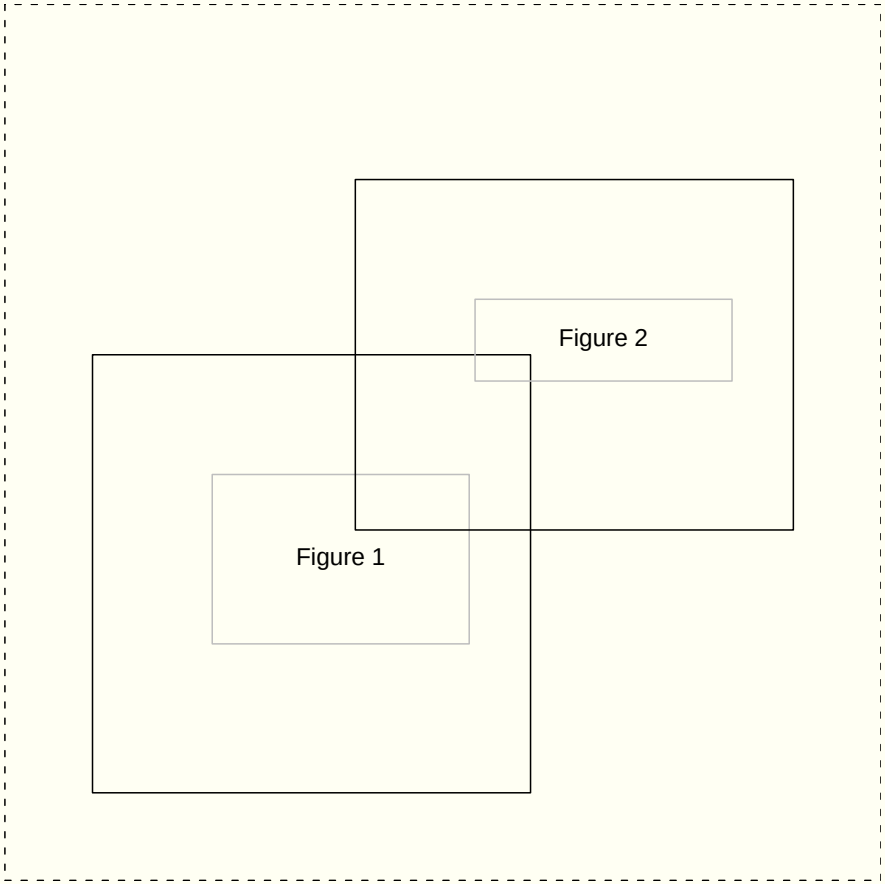
- Graphics Regions and Coordinate Systems
  - Outer Margins
  - Figure Regions
  - Figure Margins
  - Plot Regions
- Directing Graphics Output
  - Which graphics functions to use
- Producing Graphics Output
  - Graphical parameters

Outer Margins and Figure Region



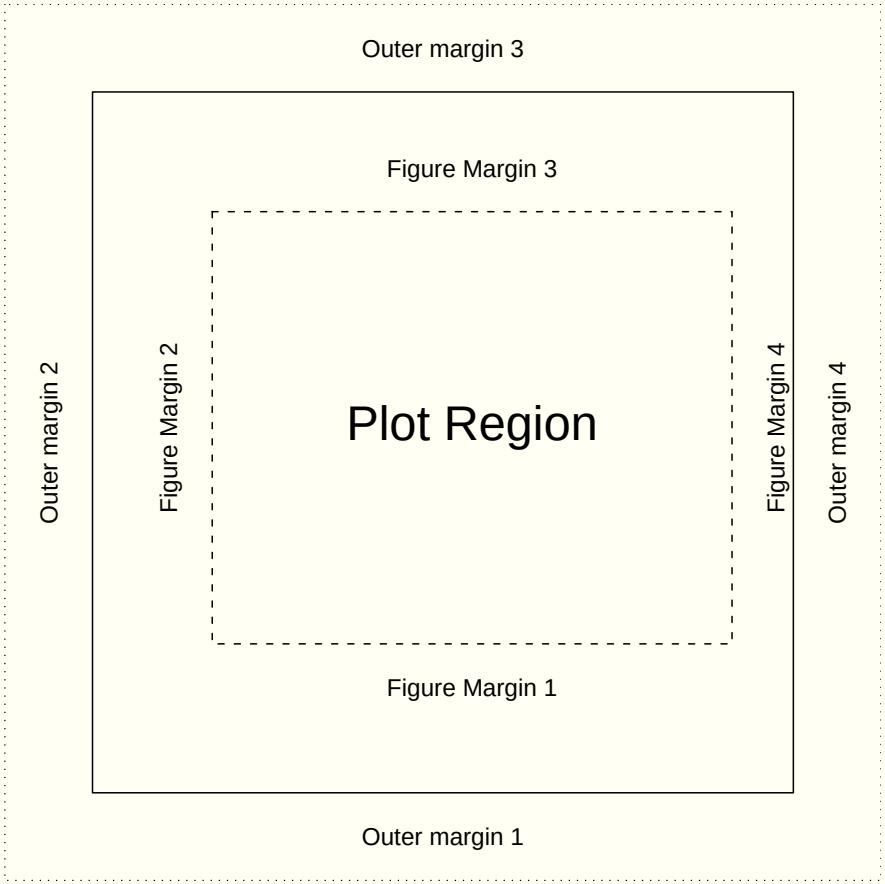

```
par(oma=c(0, 0, 0, 0), omi=)
par(mfrow=c(1, 1), mfcol=c(1, 1), fig=, fin=)
```

Arbitrary Figure Regions



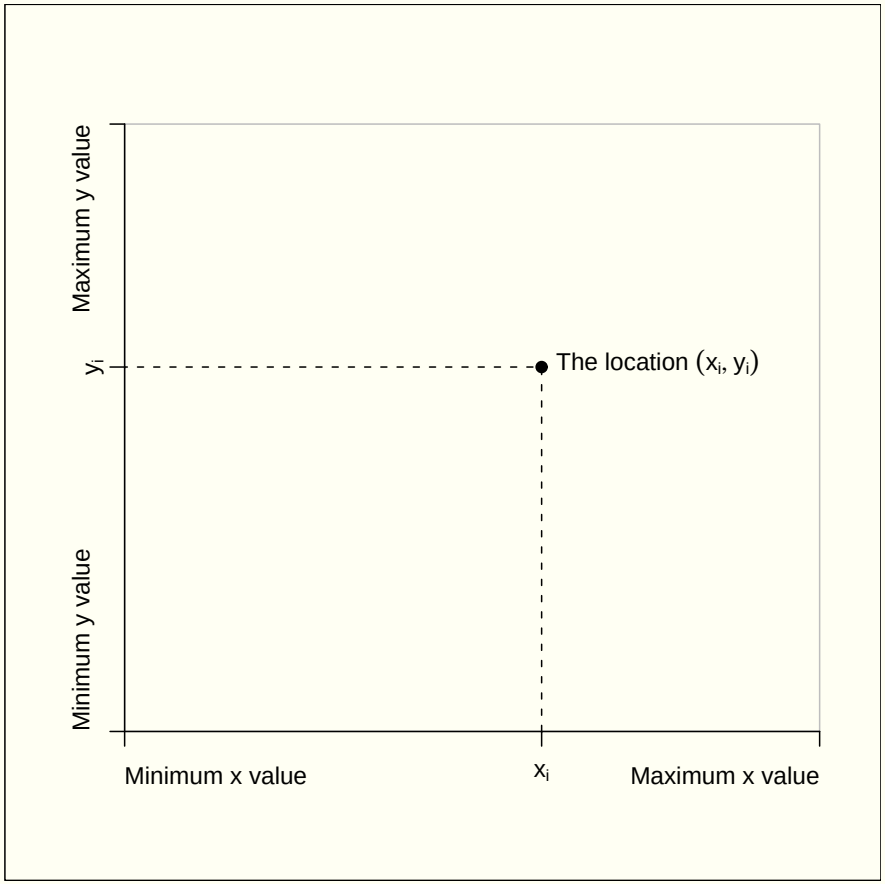

```
par(fig=c(0.1, 0.6, 0.1, 0.6))
par(new=T)
par(fig=c(0.4, 0.9, 0.4, 0.8))
```

Figure Margins and Plot Region



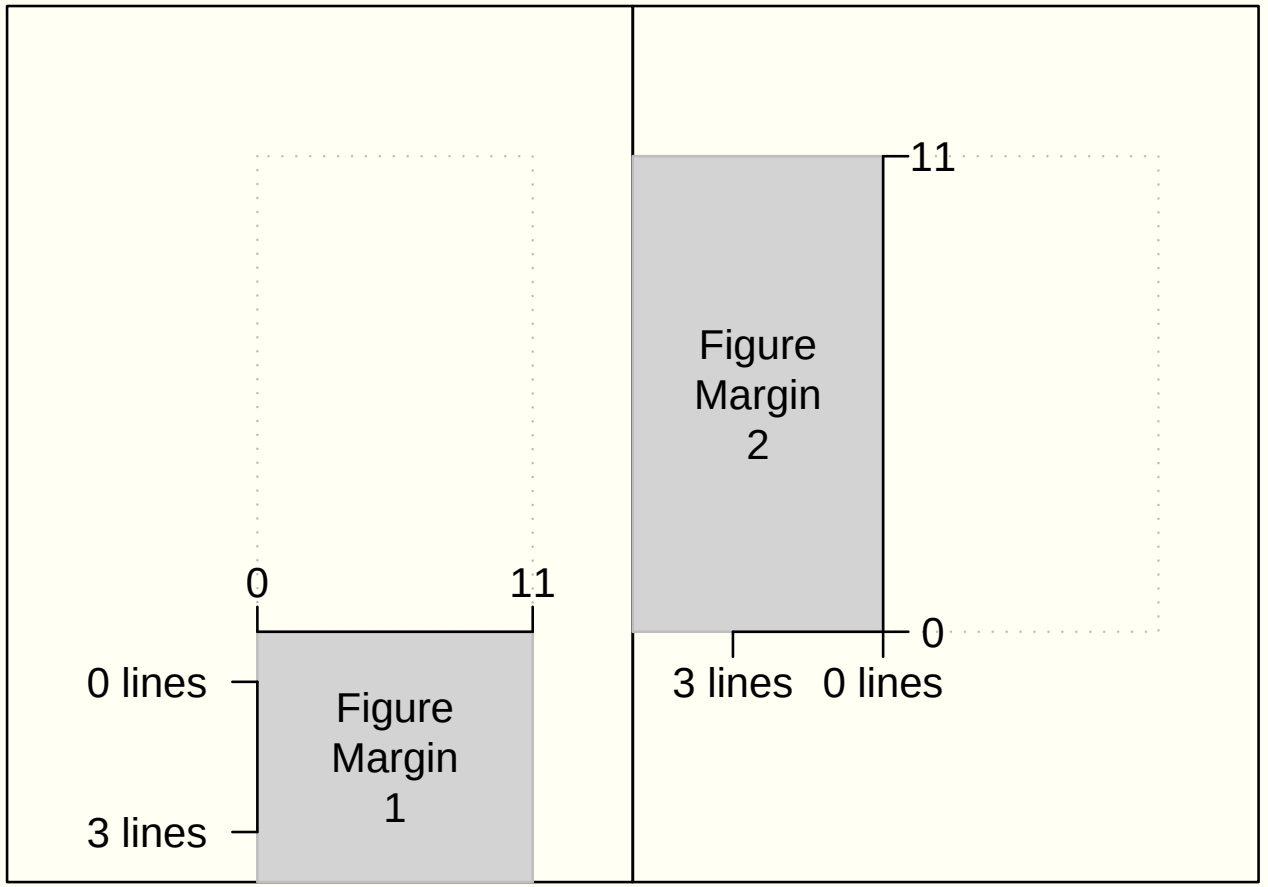

```
par(mar=c(5.1, 4.1, 4.1, 2.1), mai=)

par(pty="m", pin=, plt=)
```

User Coordinates

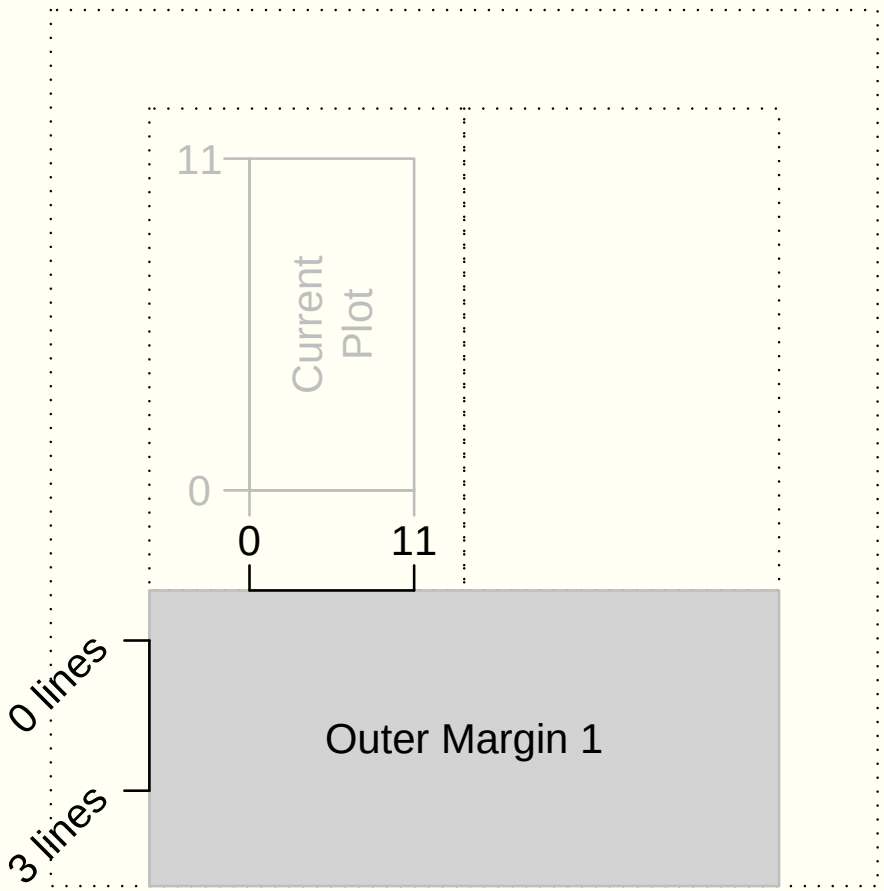


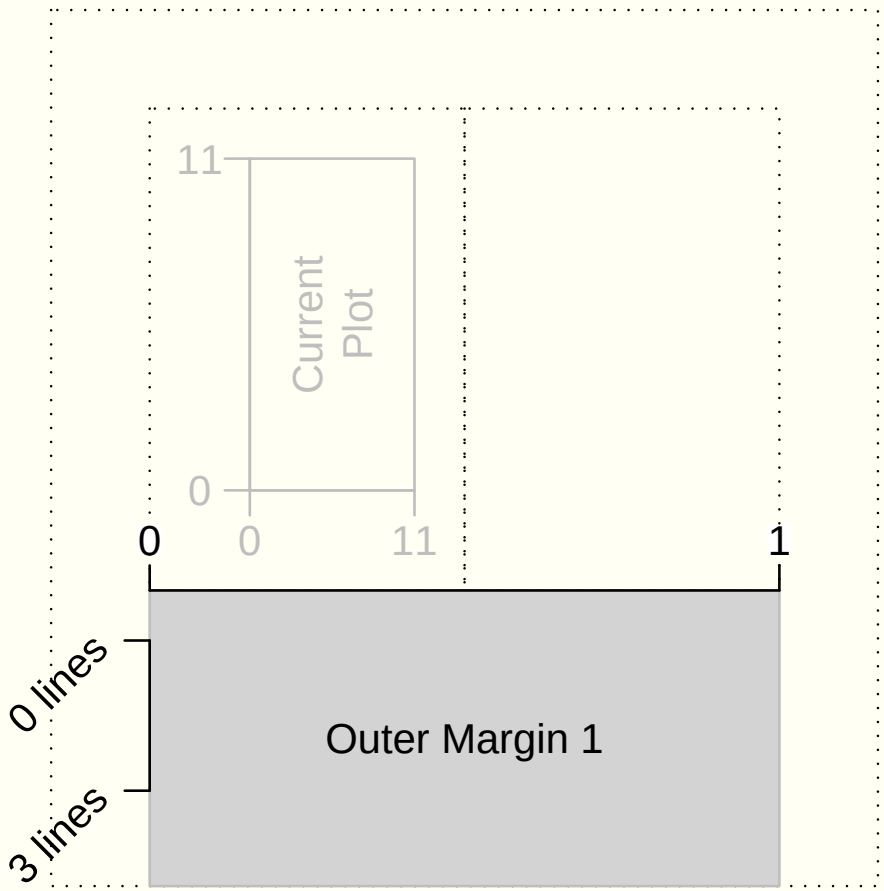

```
<plot.function>(..., xlim=, ylim=)
par(xaxs="r", yaxs="r")
```

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

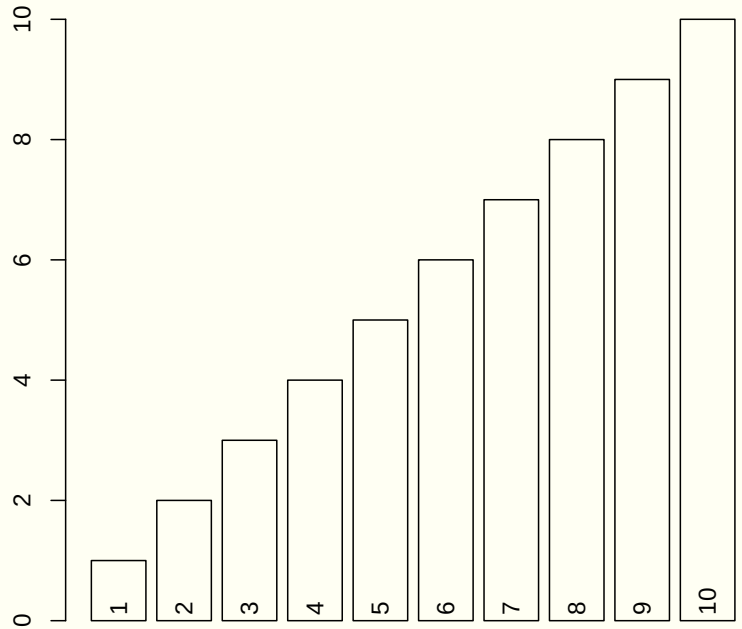
- Permanent settings
  `par(<param>=)`
- Temporary settings
  `<plot.function>(..., <param>=)`

`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

Adding to Existing Plots

- User Coordinates are not always obvious

```
> midpts <- barplot(1:5, density = -1)
> text(midpts, 0.1, 1:5, srt = 90, adj = 0)
```



Adding to Existing Plots

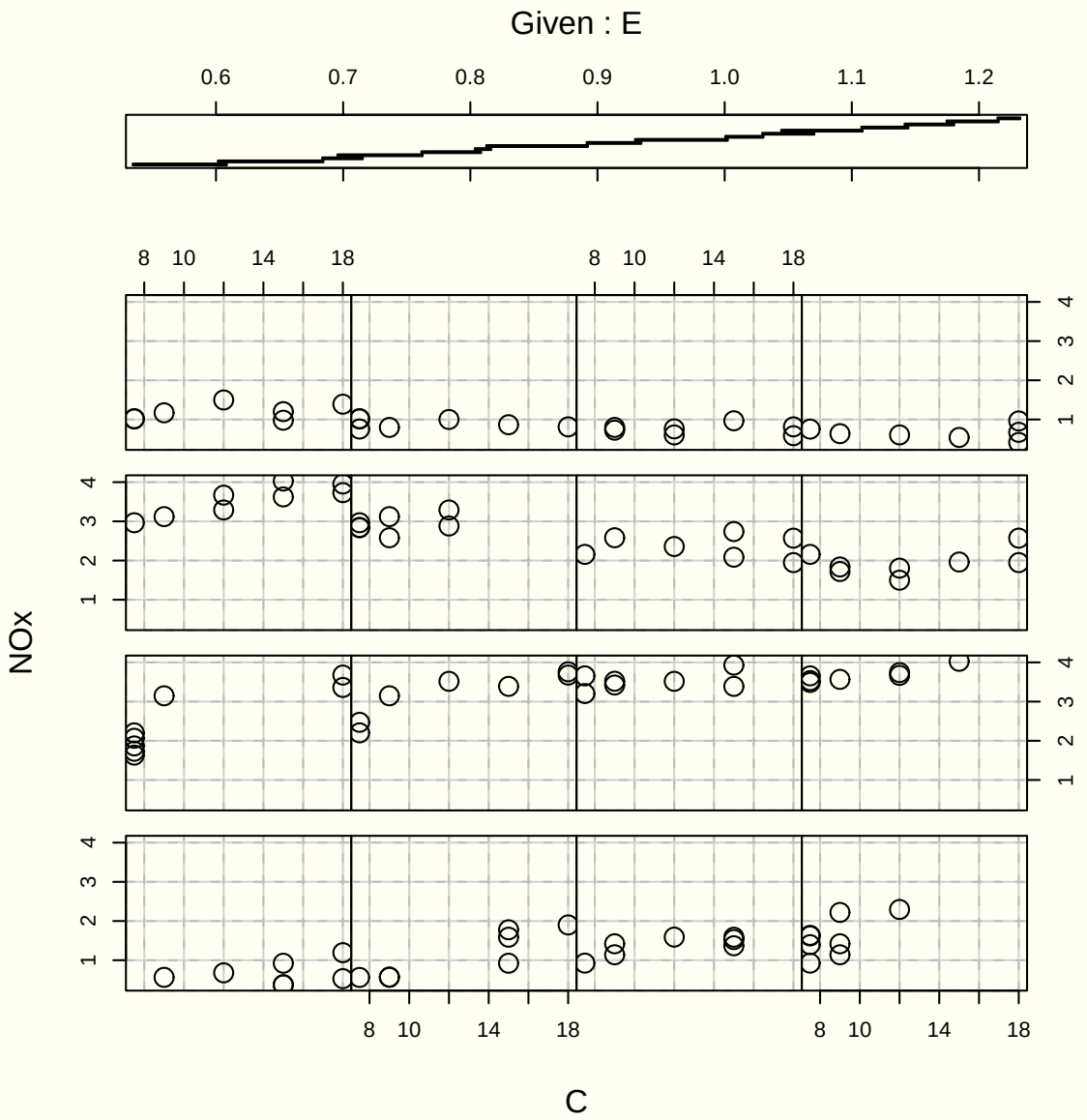
- Querying the current region and coordinate settings

```
> plot(0:1, 1:2)
> par("usr")
[1] -0.04  1.04  0.96  2.04
> par("uin")
[1] 6.603704 4.659259
```

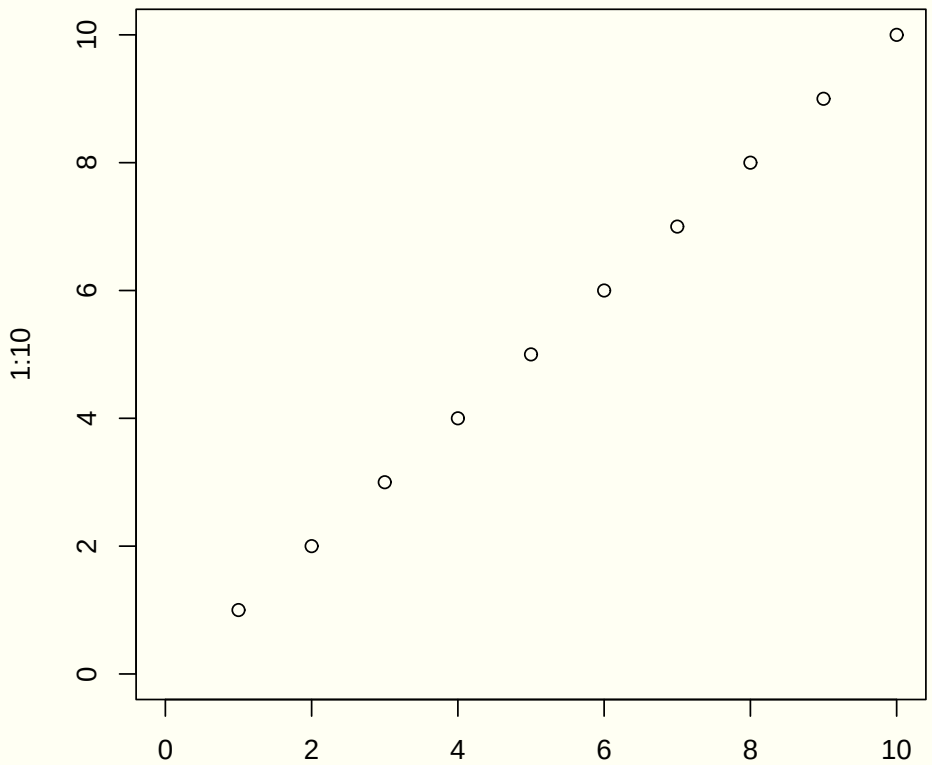
- Some complex plots revert region and coordinate settings (panel functions)

```
> E.intervals <- co.intervals(ethanol$E, 16, 0.25)
> coplot(NOx ~ C | E, given.values = E.intervals, data = ethanol,
+        panel = function(x, y, col, pch) {
+                  points(x, y);
+                  axis(1, tck=1, lty=2, labels=F);
+                  axis(2, tck=1, lty=2, labels=F) })
```

Adding to Existing Plots



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="")
```

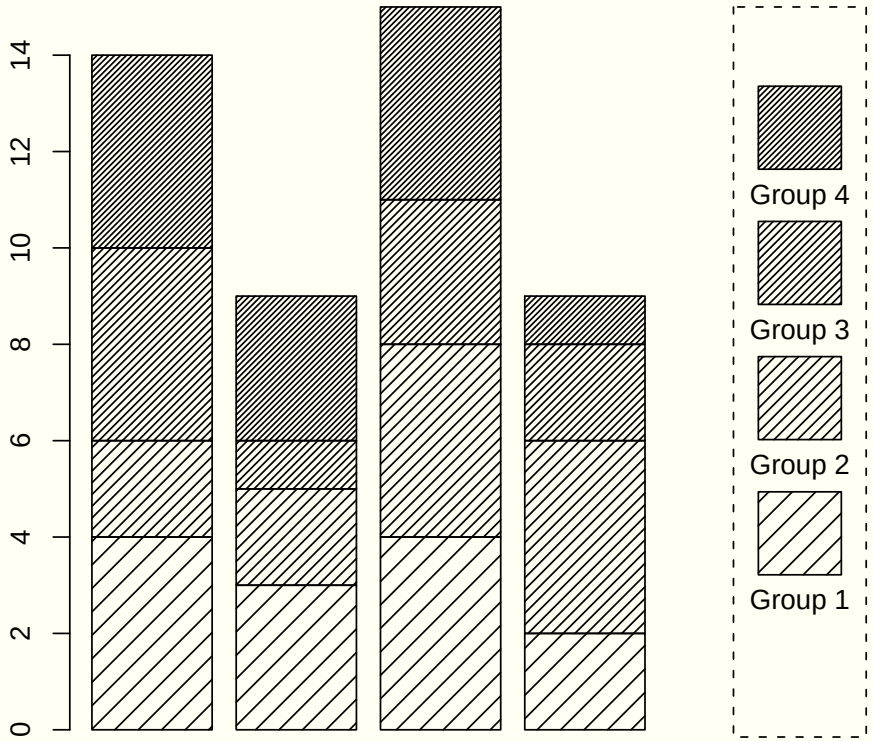
- Draw data symbols in plot region

```
> 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. .

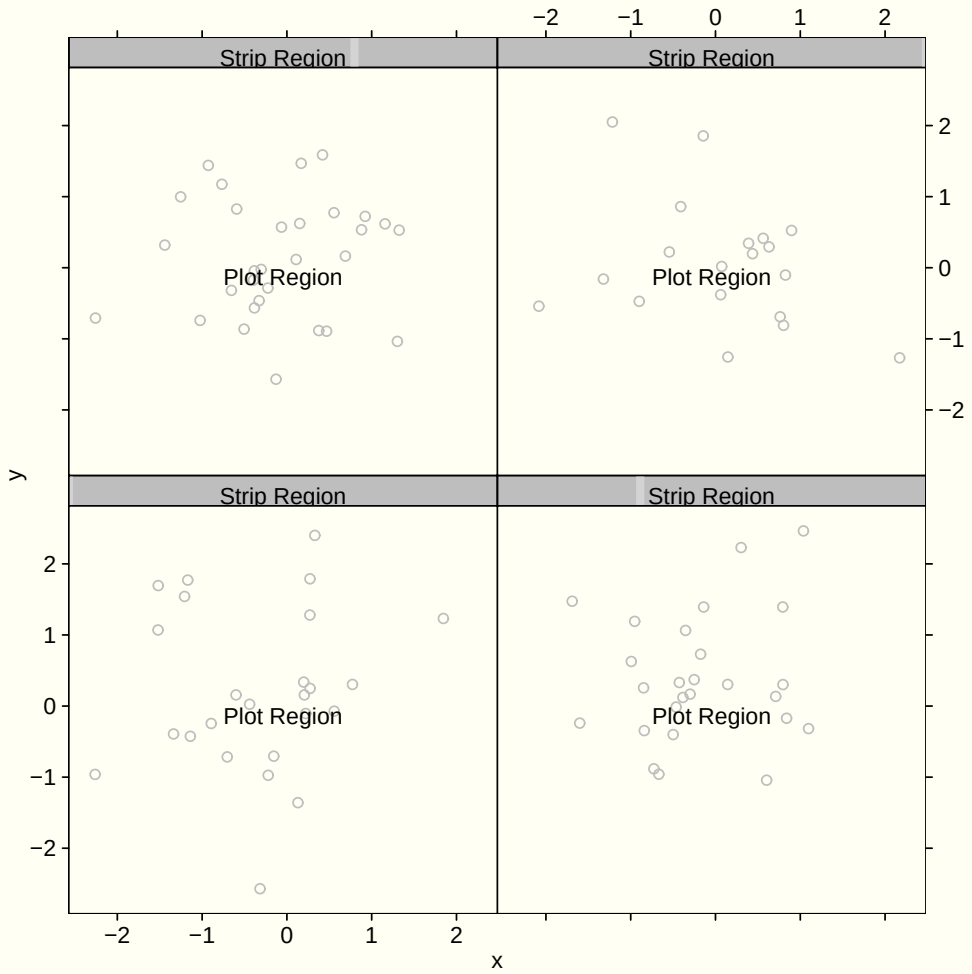
```
box(lty=2)
for (i in 1:4)
  polygon(c(0.5 - size[1]/2, 0.5 - size[1]/2,
            0.5 + size[1]/2, 0.5 + size[1]/2),
          c(i, i + size[2], i + size[2], i),
          angle=45, density=i*10)
  text(0.5, i-0.2, paste("Group", i))
```

Traditional Trellis Graphics

- plots with a design principles based on human perception experiments
  - text is horizontal
  - colours and symbols have easily-distinguishable defaults
  - “banking” of plots

- “multipanel conditioning”, where multiple plots of x versus y are produced for different levels of a grouping variable g.

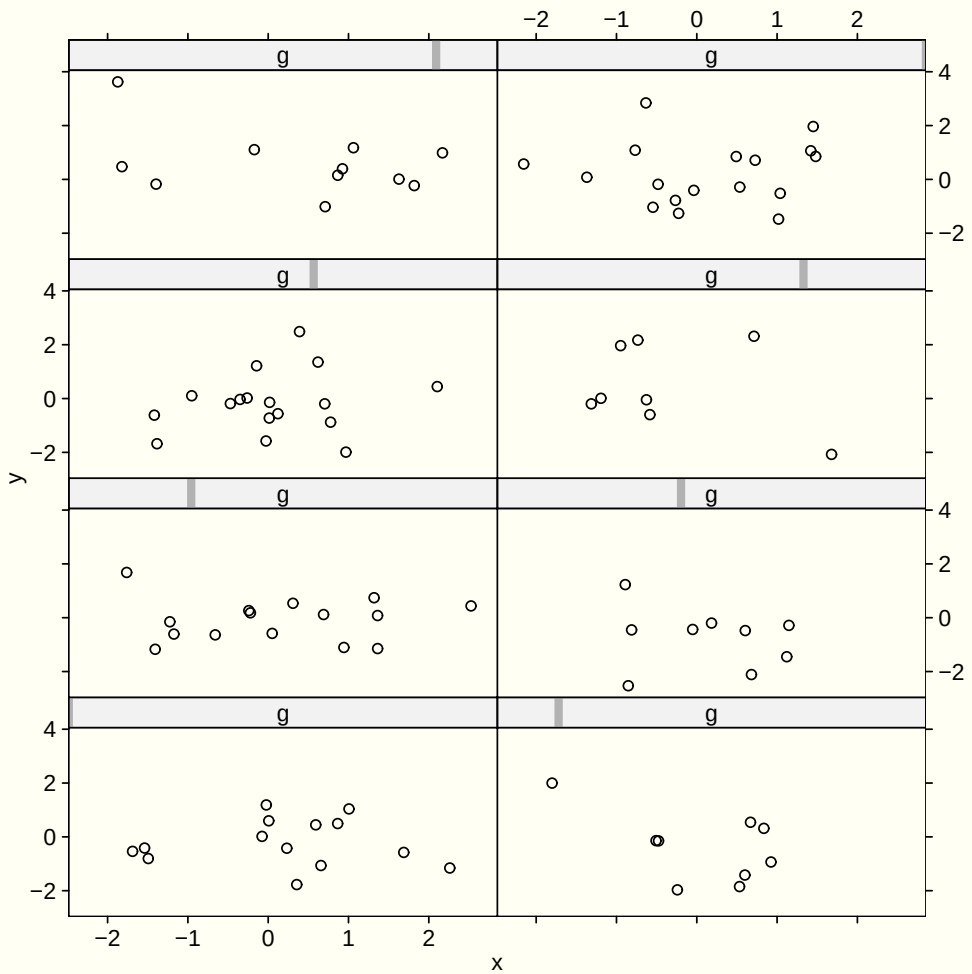
Traditional Trellis Graphics



Trellis Graphics Regions and Coordinate Systems

- Controlling number of columns and rows of plots.

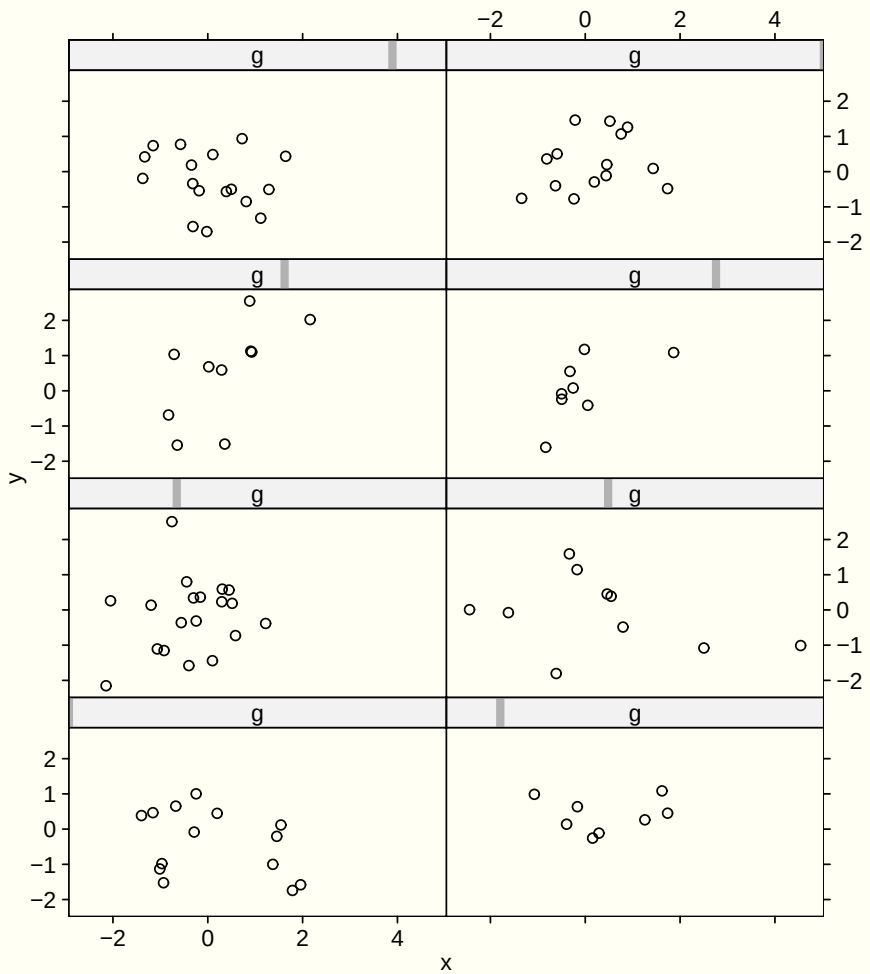
```
xyplot(y ~ x | g, layout=c(2, 4))
```



Trellis Graphics Regions and Coordinate Systems

- Controlling the aspect ratio of each plot.

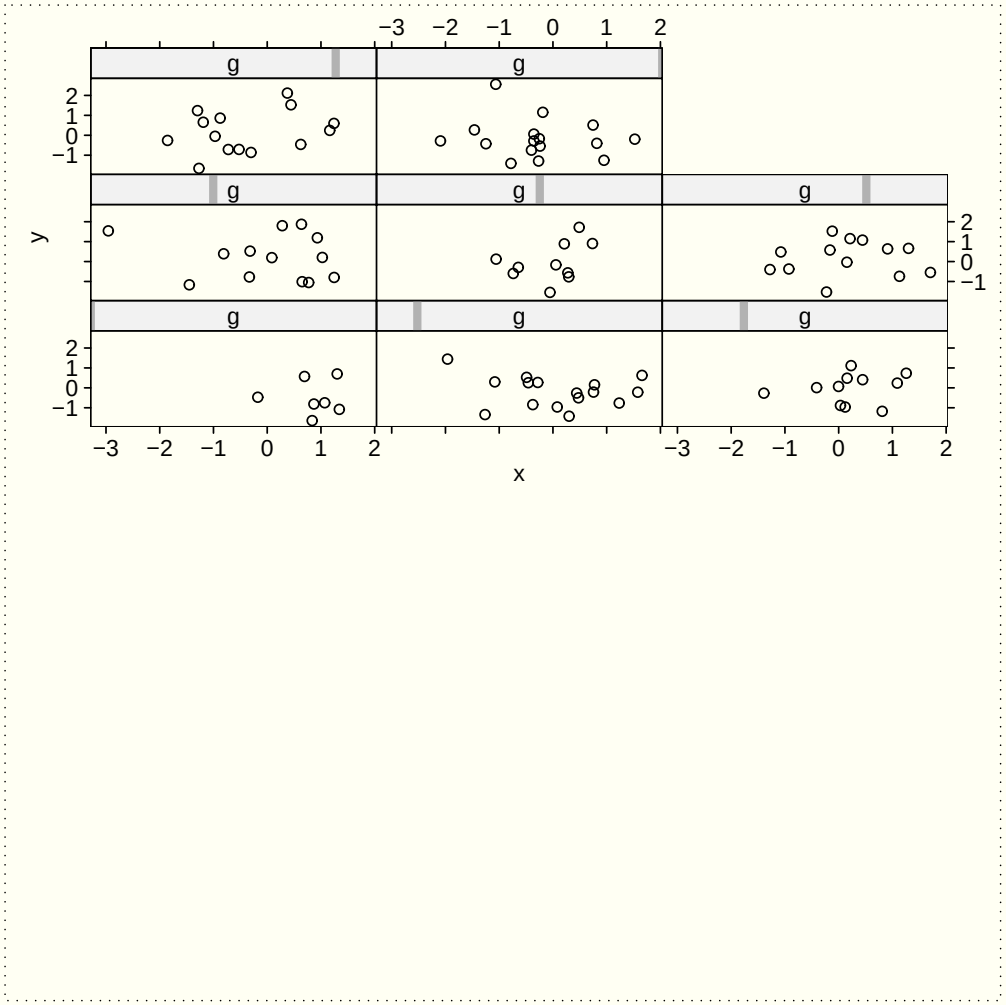
```
xyplot(y ~ x | g, aspect=0.5)
```



Trellis Graphics Regions and Coordinate Systems

- Controlling the position of the entire plot.

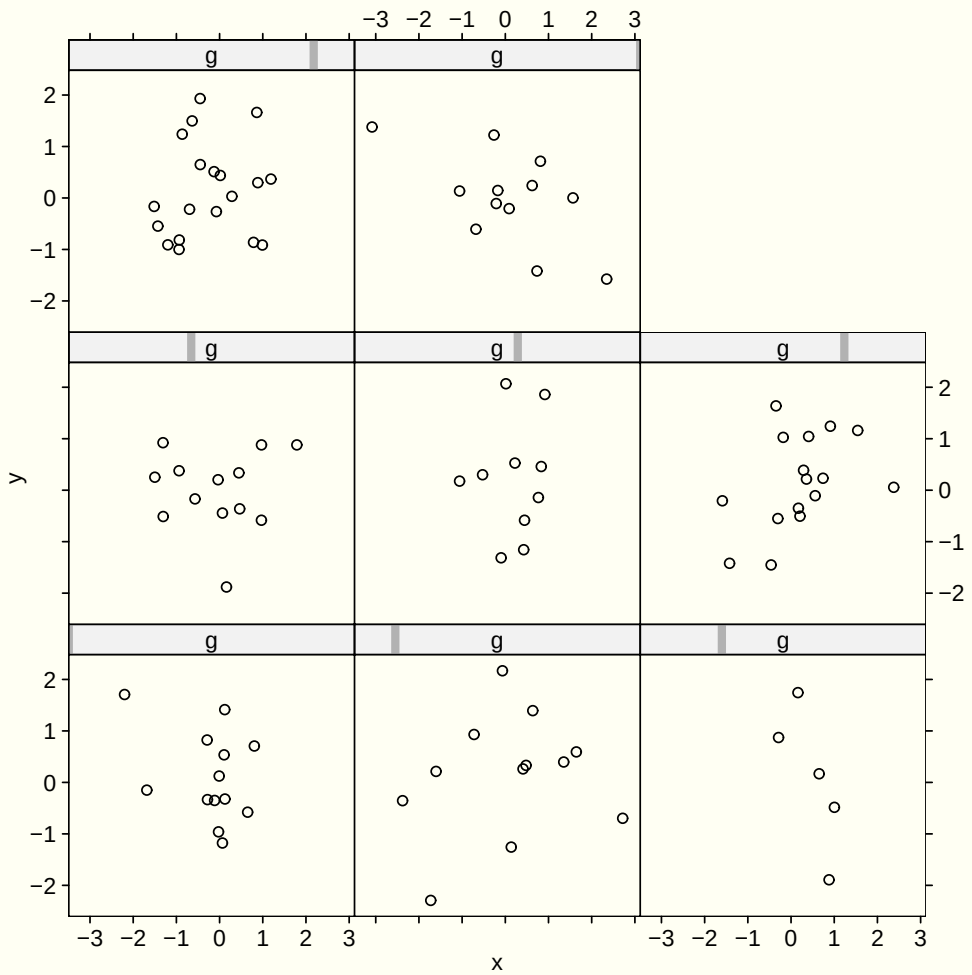
```
myplot <- xyplot(y ~ x | g)
print.trellis(myplot, position=c(0, 0.5, 1, 1))
```



Trellis Graphics Regions and Coordinate Systems

- Controlling the scales on the axes of the plot regions.

```
xyplot(y ~ x | g, scales=list(limits=c(-4, 4)))
```



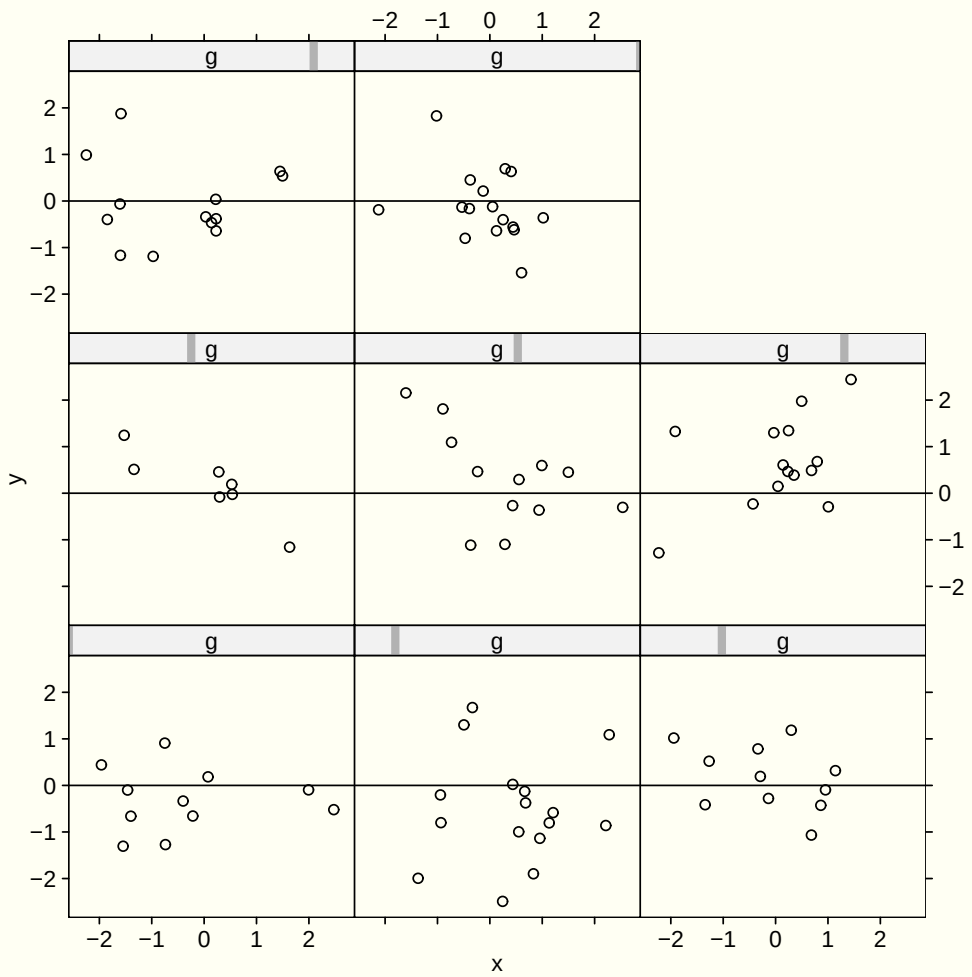
Customising Trellis Plots

- The `panel` argument.

```
xyplot(y ~ x | g,
       panel=function(x, y, ...) {
         panel.xyplot(x, y, ...);
         abline(0, 0)
       })
```

The arguments to the panel function depend on the trellis function; for trellis function **<name>**, look at the default panel function **panel.<name>**

Customising Trellis Plots

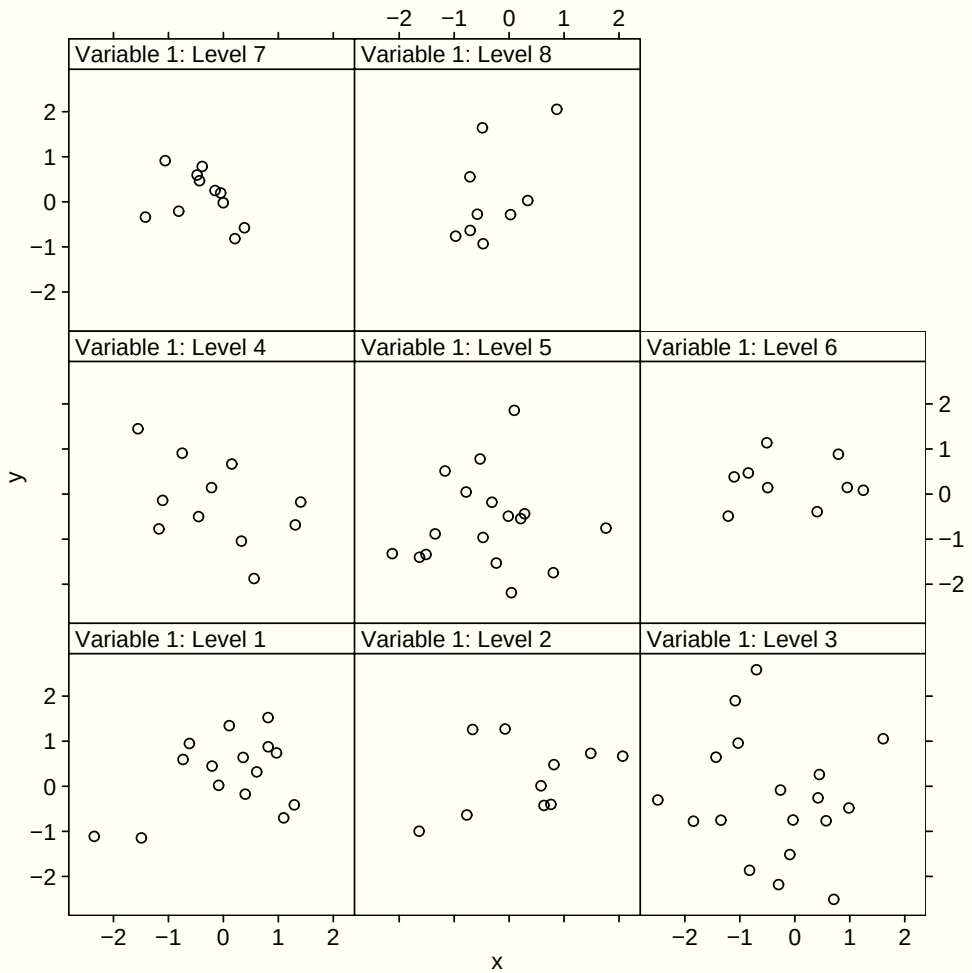


Customising Trellis Plots

- The `strip` argument.

```
xyplot(y ~ x | g,
       strip=function(which.given, which.panel,
                      var.name, factor.levels,
                      shingle.intervals,
                      par.strip.text,
                      strip.names, style) {
         text(0, 0.5,
              paste("Variable ", which.given, ": Level ",
                    which.panel[which.given], sep=""), adj=0)
       })
```

Customising Trellis Plots

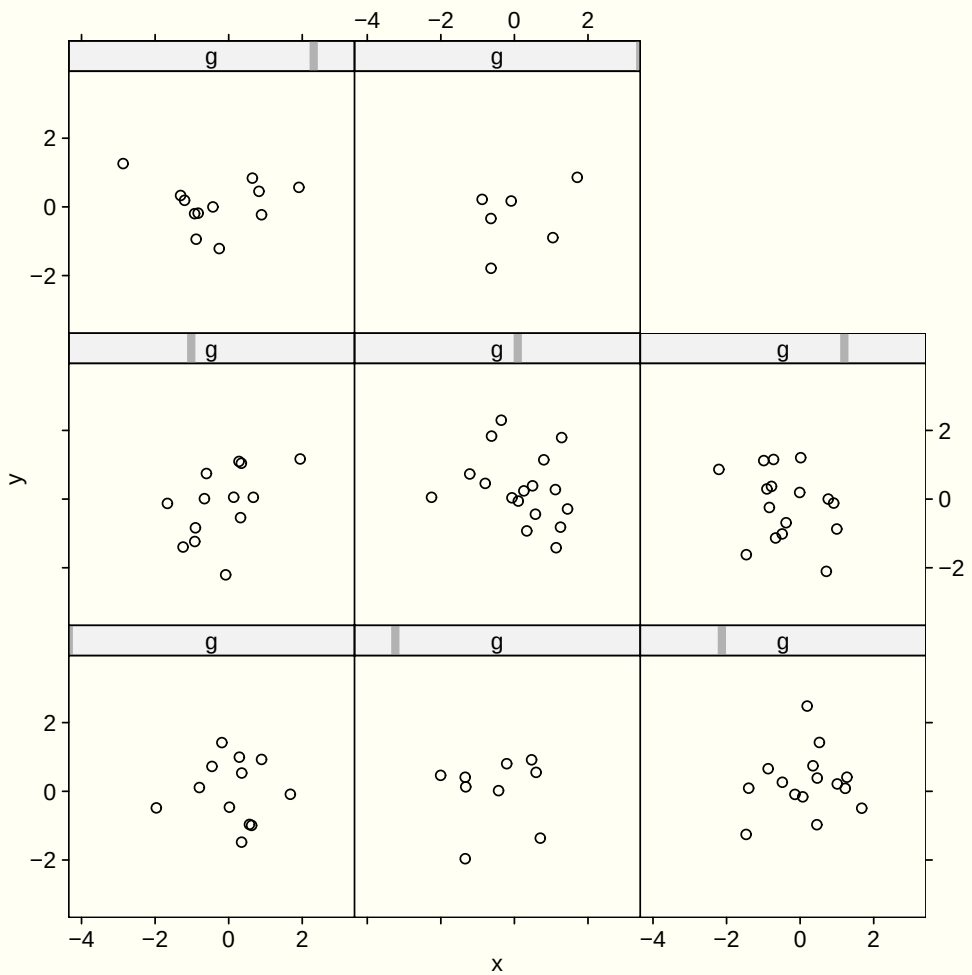


Customising Trellis Plots

- The `prepanel` argument.

```
xyplot(y ~x|g,
       prepanel=function(x ,y) {
         list(xlim=range(x)+c(-1,1),
              ylim=range(y)+c(-1,1))
       })
```

Customising Trellis Plots



Trellis Graphical Parameters

- Temporary settings:

```
> xyplot(y ~ x | g, pch=16)
```

- Permanent settings:

```
> trellis.par.get("plot.symbol")
$cex:
[1] 0.8

$col:
[1] 2

$font:
[1] 1

$pch:
[1] 1
> ps <- trellis.par.get("plot.symbol")
> ps$pch=16
> trellis.par.set("plot.symbol", ps)
```

CD Contents

`base.pdf`	"Traditional S Graphics" notes
`base.S`	S code for examples in base.pdf
`slide.pdf`	"Traditional S Graphics" slides
`ex.pdf`	"Traditional S Graphics" exercises
`ex.S`	S code for creating data for exercises
`model.pdf`	"Traditional S Graphics" answers to exercises
`model.S`	S code for answers to exercises