# R Visualisation

Irish Centre for High End Computing (ICHEC)

March 19, 2013

# Visualization Techniques Outline

- graphics: scatter, bar, pie, 3D, box plots, graphics devices
- practical
- lattice, ggplot2, RGL graphics

# Charting Data

- Useful to clean workspace

```
rm(list = ls(all = TRUE))
```

- Load HSAUR2 data

```
install.packages("HSAUR2")
library("HSAUR2")
data("Forbes2000")

data("Forbes2000", package = "HSAUR")

# Save default plot settings
def.par <- par(no.readonly = TRUE)
```

# plot() Parameters

- x, y
  - x and y data to be plotted
- type
  - line type, “p” for points, “l” for lines, “b” for both, etc.
- xlim
  - the x limits (x1, x2) of the plot. Note that x1 can be greater than x2 and leads to a reversed axis.
- log
  - Character string which contains ‘”x”’ if the x axis is to be logarithmic
- main
  - Main title for the plot.
- sub
  - Subtitle for the plot.
- xlab, ylab
  - Labels for the x- and y-axes, respectively.
- ann
  - logical value indicating whether the default annotation (title and x and y axis labels) should appear on the plot.

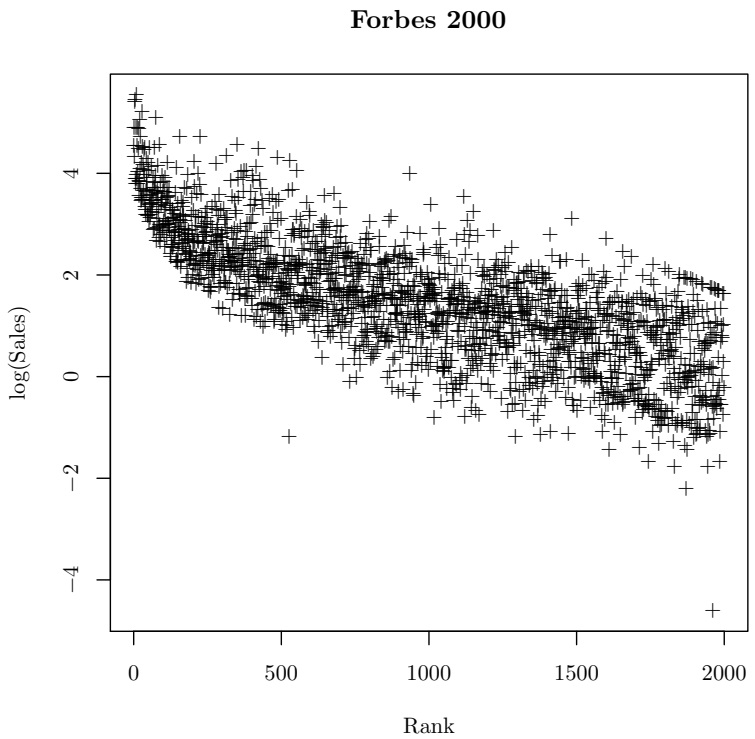
# plot() Parameters

- axes
  - Controls whether axes will be plotted on the chart.
- frame.plot
  - logical indicating whether a box should be drawn around the plot.
- panel.first
  - an expression to be evaluated after the plot axes are set up but before any plotting takes place. This can be useful for drawing background grids or scatterplot smooths.
- panel.last
  - an expression to be evaluated after plotting has taken place.
- asp
  - the y/x aspect ratio, see 'plot.window'.
- pch
  - specify symbols to mark the points
  - e.g. "+", ".", "-", "o"
- lwd
  - line width, default=1, 2 is twice as wide.
- add
  - Should this plot be added to the existing plots on the device?

# Scatter Plots

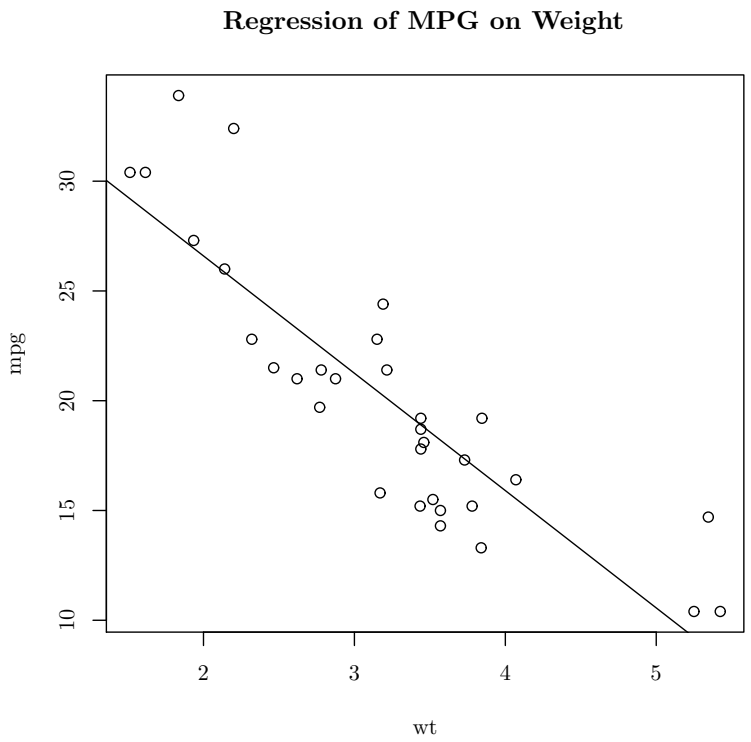
```
plot(Forbes2000$rank,
     log(Forbes2000$sales),
     xlab="log(Rank)",
     ylab="Sales",
     main="Forbes 2000",
     pch = "+"
)
```

```
identify(Forbes2000$rank,
         log(Forbes2000$sales),
         Forbes2000$country)
```



# Scatter Plots

```
attach(mtcars)

plot(wt, mpg)

abline(lm(mpg~wt))

title(
 "Regression of MPG on Weight"
     )
```




Note: Formulas. Tilde is used to show relationship between the response variables (left) and the stimulus variables (right). The plus sign expresses a linear relationship between variables.

```
sample.formula <- y~x1+x2
```

# Graphics Devices

ICHEC
Irish Centre for High-End Computing

- Saving a png file.

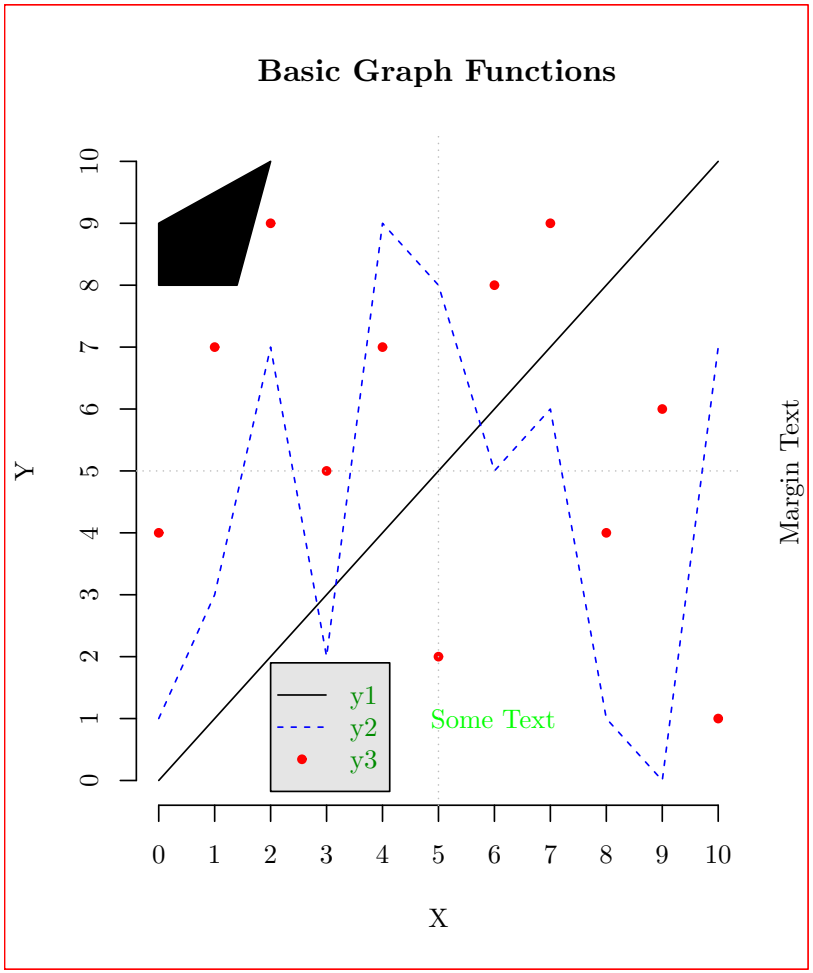
```
png("barplot.png", bg="transparent", width=900, height=700)
par(mar=c(10,6,4,2) + 0.1) #  'c(bottom, left, top, right)'
par(bg="white")
bp <- barplot(
 table(Forbes2000$category), xaxt="n", xlab="", ylim=c(0, 350))
mylabels <- sort(unique(Forbes2000$category))
text(bp, par ("usr")[3], labels=mylabels, srt=35, offset=1,
 adj=1, xpd=TRUE)
dev.off()
```

- Other file formats.

```
pdf("mygraph.pdf")            # pdf file
win.metafile("mygraph.wmf") # windows metafile
png("mygraph.png")            # png file
jpeg("mygraph.jpg")           # jpeg file
bmp("mygraph.bmp")            # bmp file
postscript("mygraph.ps")      # postscript file
pictex("mygraph.tex")         # LaTeX/PicTeX file
```

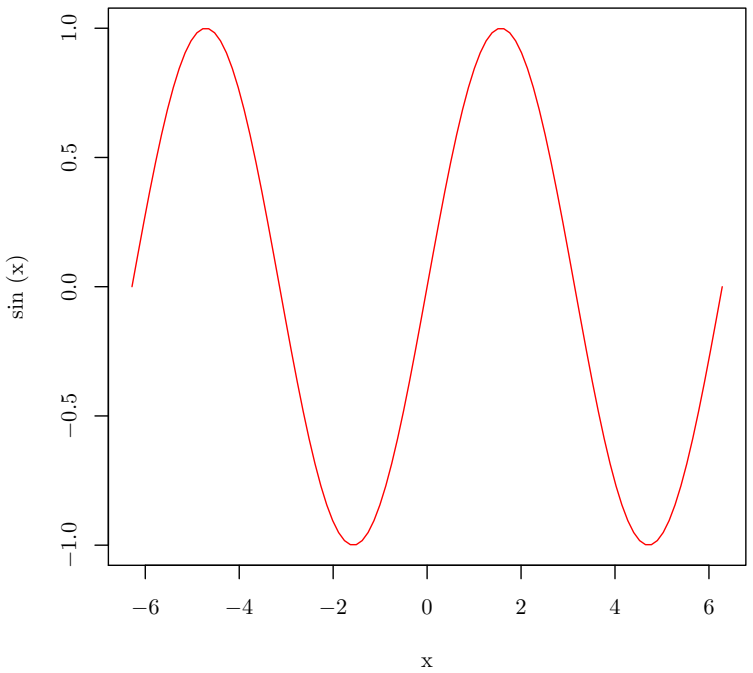
# Basic Graphics Functions

```
x <- 0:10
y1 <- 0:10
y2 <- c(1,3,7,2,9,8,5,6,1,0,7)
y3 <- c(4,7,9,5,7,2,8,9,4,6,1)
plot(x, y1, type="l", lty=1,
 lwd=1, col="black", main="",
 xlab="X", ylab="Y",
 axes = FALSE);

axis(1, x)
axis(2, x)
title(main="Basic Graph Functions")
lines(x,y2, lty=2,col="blue")
points(x,y3, lty=1, pch=20, col="red",
 bg="red")

text(6,1,labels="Some Text",col="green")
abline(h=5, col="grey",lty=3)
abline(v=5, col="grey",lty=3)
polygon(x=c(0,1.4,2,0), y=c(8,8,10,9),
 col="black")

legend(x=2, y=1.9, legend=c("y1", "y2",
 "y3"), col = c("black","blue","red"),
 text.col = "green4", lty = c(1, 2, -1),
 pch = c(-1, -1, 20),
 merge = TRUE, bg = "gray90")

box(which="outer",lty="solid",col="red")
mtext("Margin Text", side=4, line=1)
```



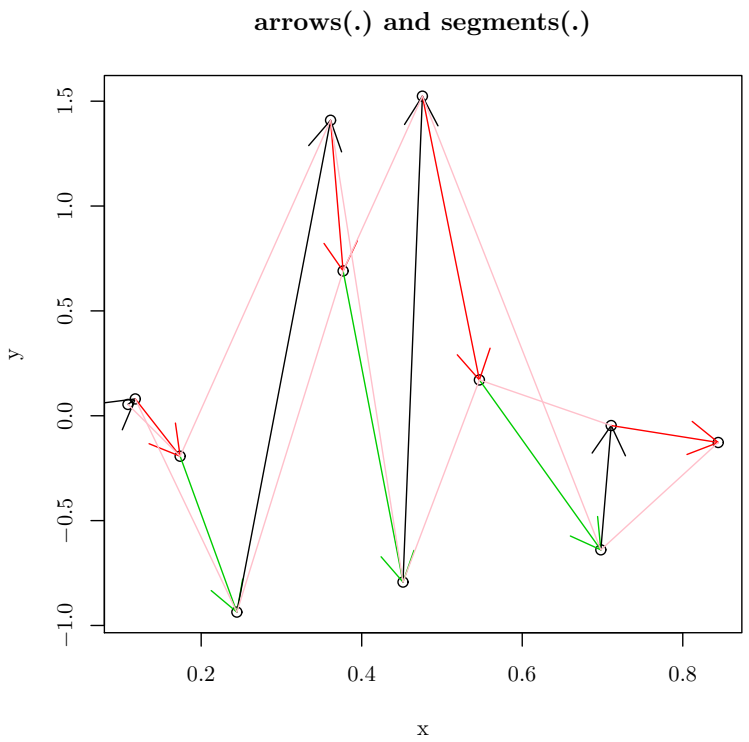
# Curve Plot

```
curve(expr=sin,
      from=-2*pi,
      to=2*pi,
      col="red")

chippy <- function(x)
  sin(cos(x)*exp(-x/2))
curve(chippy, -8, 7, n=2001)
```



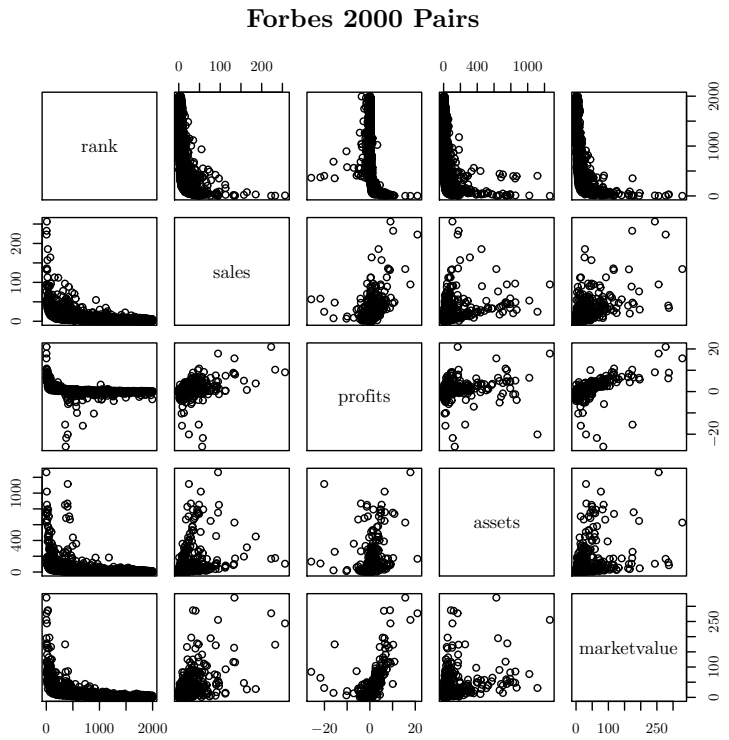
# Arrows and Segments

```
x <- stats::runif(12)
y <- stats::rnorm(12)

i <- order(x,y)
x <- x[i]
y <- y[i]

plot(x, y,
 main=
 "arrows(.) and segments(.)")

# one shorter than data
s <- seq(length(x)-1)

# draw arrows from pt to pt
arrows(x0=x[s], y0=y[s],
       x1=x[s+1], y1=y[s+1],
       col= 1:3)

s <- s[-length(s)]
segments(x[s], y[s], x[s+2],
  y[s+2], col= "pink")
```



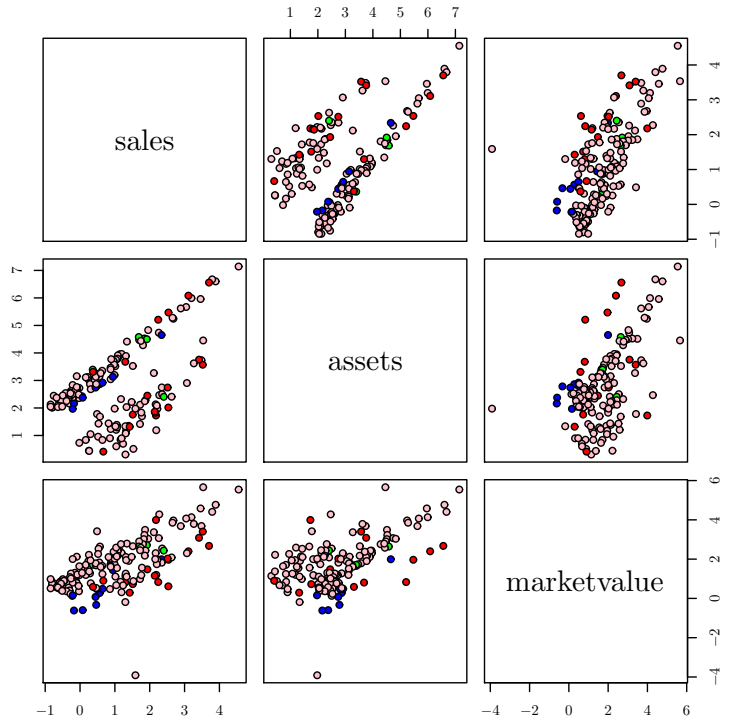
# Pairs

```
pairs(Forbes2000[
    c("rank", "sales",
      "profits", "assets",
      "marketvalue")
                  ],
       main="Forbes 2000 Pairs"
      )
```



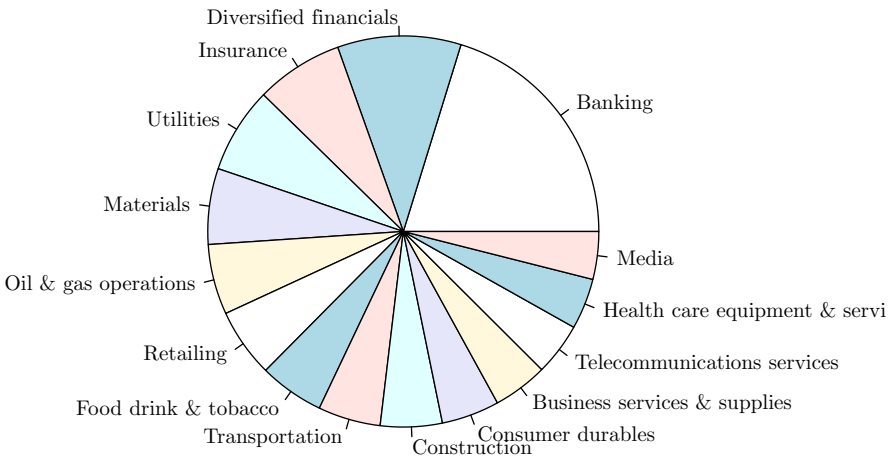
# Pairs

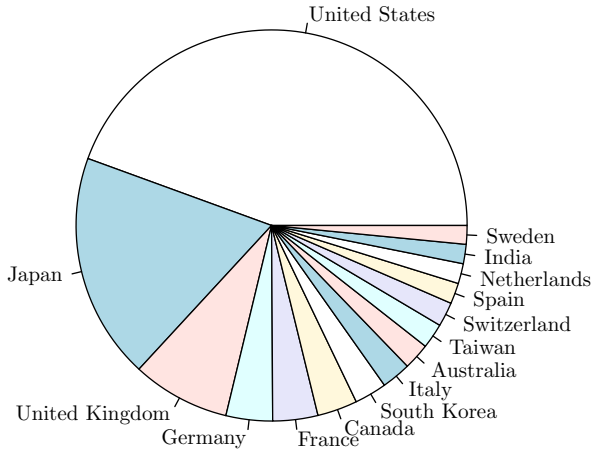
```
Forbes2000.fourCountries <-
 subset(Forbes2000,country %in% c(
  "United States","Germany",
  "India","Ireland"))

Forbes2000.fourCategories <-
 subset(Forbes2000.fourCountries,
  category %in% c("Banking",
   "Chemicals", "Construction",
   "Software & services"))

Forbes2000.fourCategories <-
  droplevels(Forbes2000.fourCategories)

pairs(log(
 Forbes2000.fourCategories[
 c("sales", "assets", "marketvalue")]),
 pch = 21,
 bg = c("red", "blue", "green",
  "pink") [unclass(
  Forbes2000.fourCategories$country)])
```



# Pie Charts

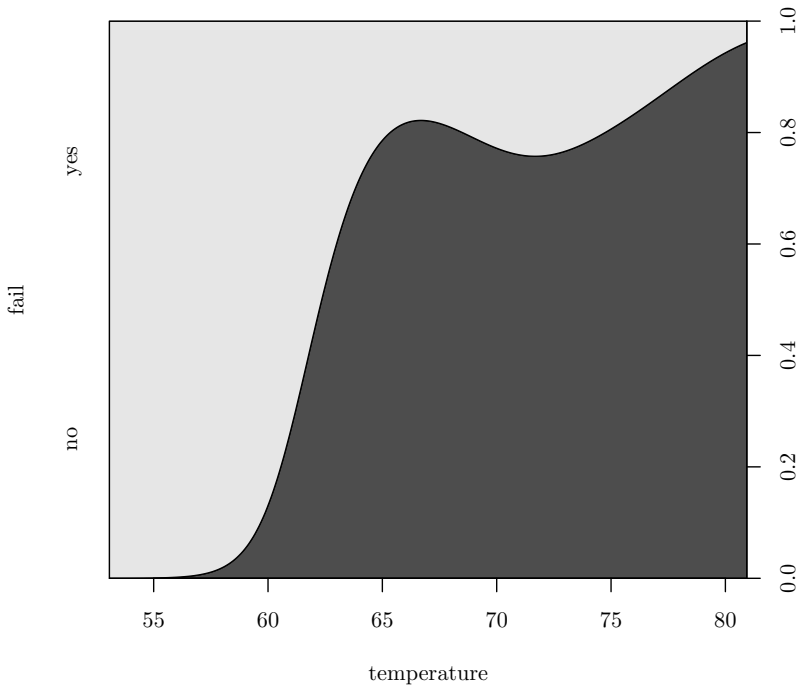
```
pie(sort(table(Forbes2000$category), decreasing = TRUE)[0:15])

pie(sort(table(Forbes2000$country), decreasing = TRUE)[0:15])
```



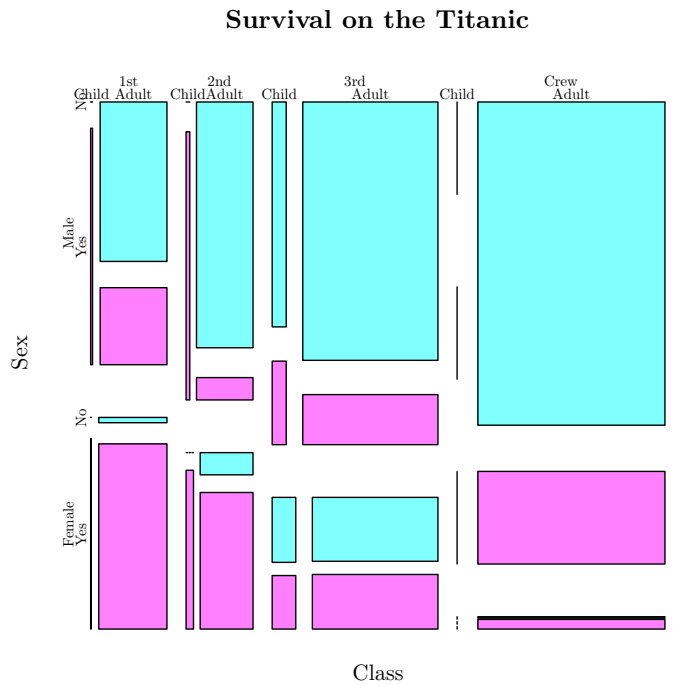
# cdplot

```
## NASA space shuttle o-ring failures
fail <- factor(
 c(2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 2, 1,
   2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1),
         levels = 1:2, labels = c("no", "yes"))
temperature <-
 c(53, 57, 58, 63, 66, 67, 67, 67,
   68, 69, 70, 70, 70, 70, 72, 73,
   75, 75, 76, 76, 78, 79, 81)

cdplot(fail ~ temperature)
```



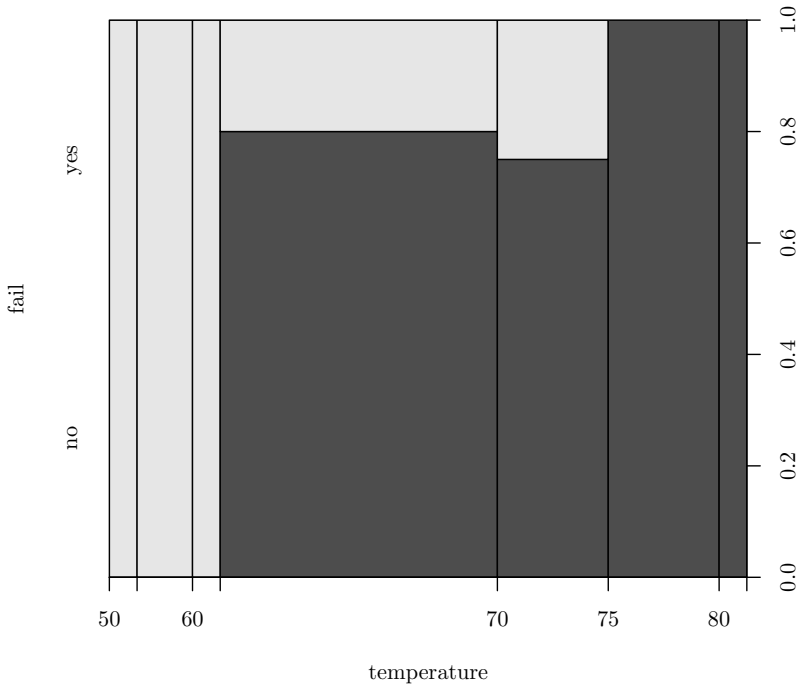
# mosaicplot

```
mosaicplot(Titanic,
 main = "Survival on the Titanic",
 color = cm.colors(2))
```



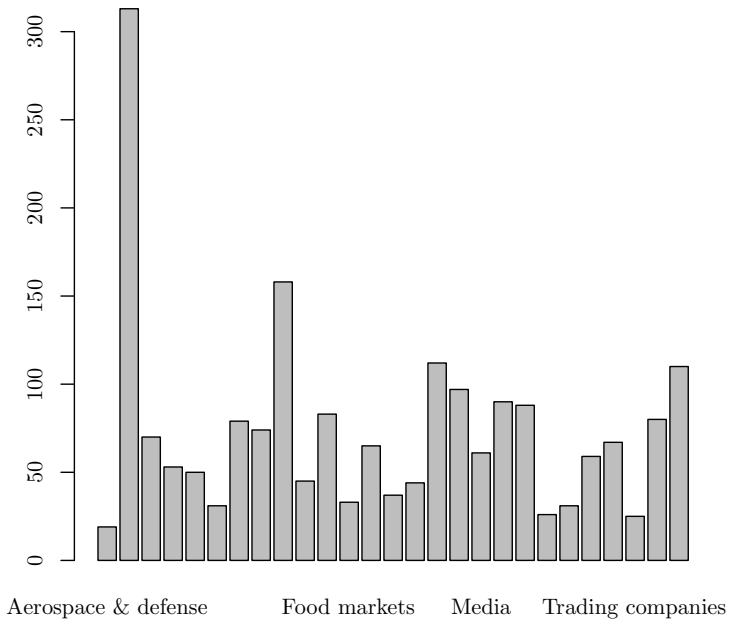
# spineplot

```
## NASA space shuttle o-ring failures
fail <- factor(
 c(2, 2, 2, 2, 1, 1, 1, 1, 1, 1,
   2, 1, 2, 1, 1, 1, 1, 2, 1, 1,
   1, 1, 1),
              levels = c(1, 2),
              labels = c("no", "yes"))
temperature <-
 c(53, 57, 58, 63, 66, 67, 67, 67,
   68, 69, 70, 70, 70, 70, 72, 73,
   75, 75, 76, 76, 78, 79, 81)

## (dependence on a numerical variable)
(spineplot(fail ~ temperature))
```



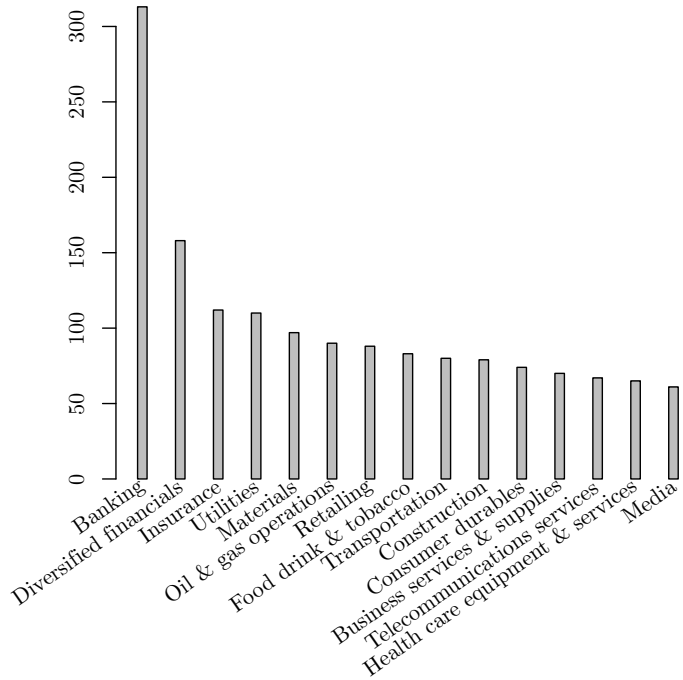
# Bar Plots

```
barplot(height, width = 1, space = NULL,
        names.arg = NULL,
        legend.text = NULL,
        beside = FALSE, horiz = FALSE,
        density = NULL, angle = 45,
        col = NULL, border = par("fg"),
        main = NULL, sub = NULL,
        xlab = NULL, ylab = NULL,
        xlim = NULL, ylim = NULL,
        xpd = TRUE, log = "",
        axes = TRUE, axisnames = TRUE,
        cex.axis = par("cex.axis"),
        cex.names = par("cex.axis"),
        inside = TRUE, plot = TRUE,
        axis.lty = 0, offset = 0,
        add = FALSE, args.legend = NULL,
        ...)

# Example
barplot(
  table(
   Forbes2000$category
       ))
```



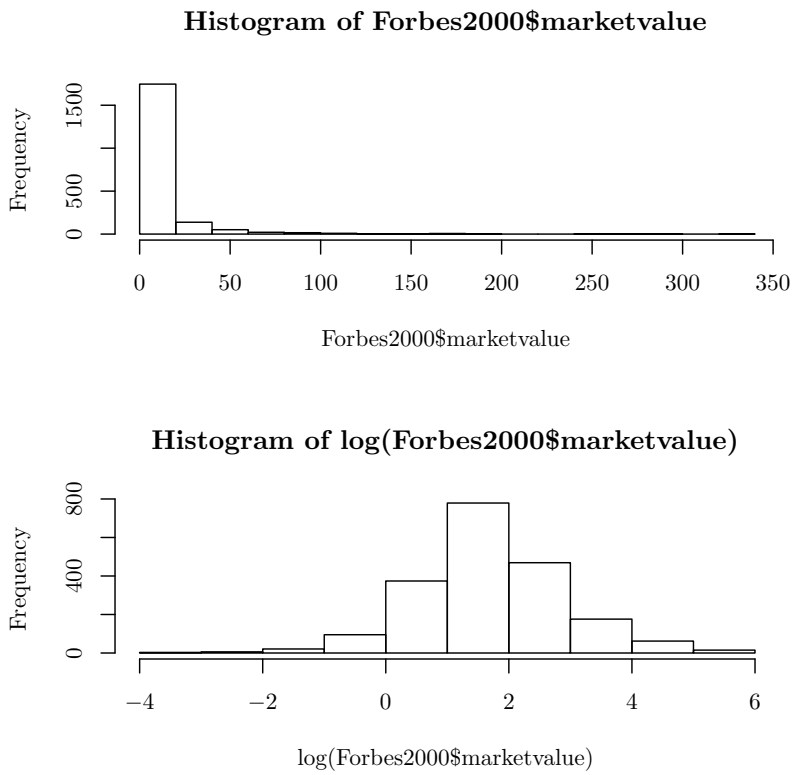
# Bar Plots

```
lD <- sort(
  table(Forbes2000$category),
  decreasing = TRUE)[0:15]
labels <- names(lD)

# store original settings
def.par <- par()
# 'c(bottom, left, top, right)'
par(mar=c(8,6,4,2) + 0.1)

bp <- barplot(lD,
     xaxt="n", # no x-axis text
     xlab="",  # no x-axis label
     space=rep(c(3),c(15)),
     width=rep(c(1.5),c(15)))

text(x=bp, y=par ("usr")[3],
     labels=labels,
     srt=35, # string rotation
     adj=c(1,1), # right-justified text
     xpd=TRUE) # clipping to fig region

# restore settings
par(def.par)
```



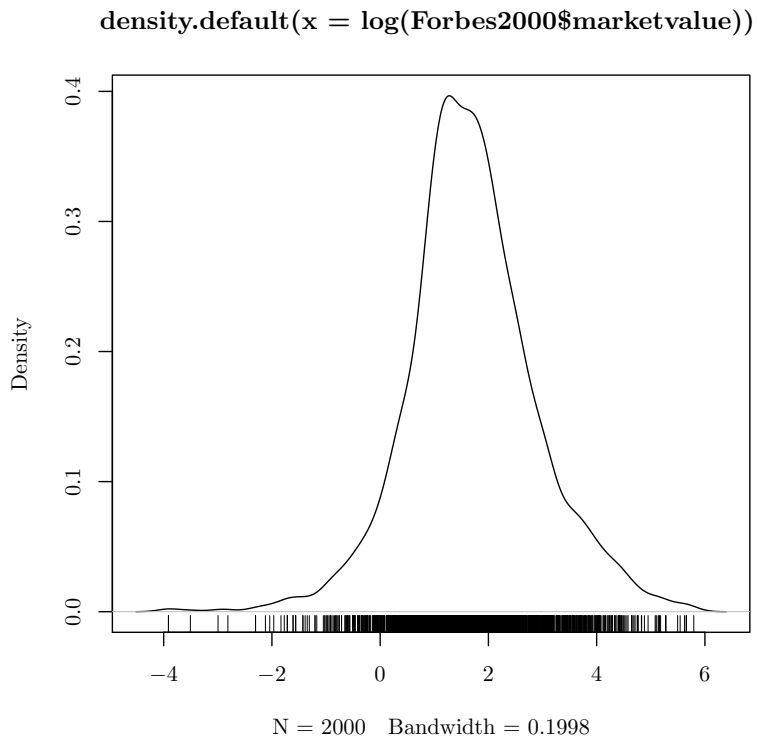
# Histograms

```
# store original settings
def.par <- par()

layout(matrix(1:2, nrow = 2))
hist(Forbes2000$marketvalue)
hist(
 log(
  Forbes2000$marketvalue
))

# restore settings
par(def.par)
```



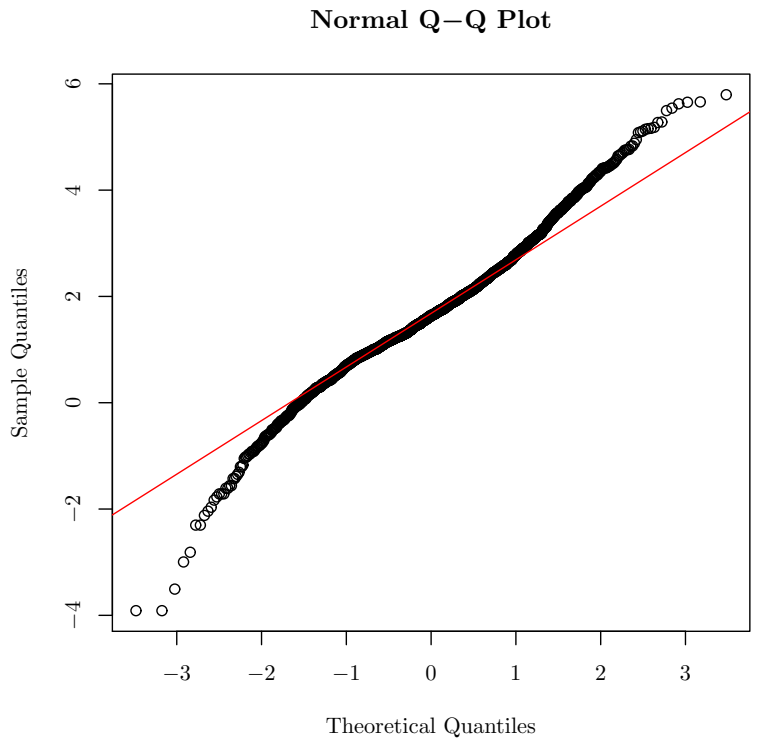
# Density

```
plot(
 density(
  log(
   Forbes2000$marketvalue
)))

rug(
 log(
  Forbes2000$marketvalue
))
```



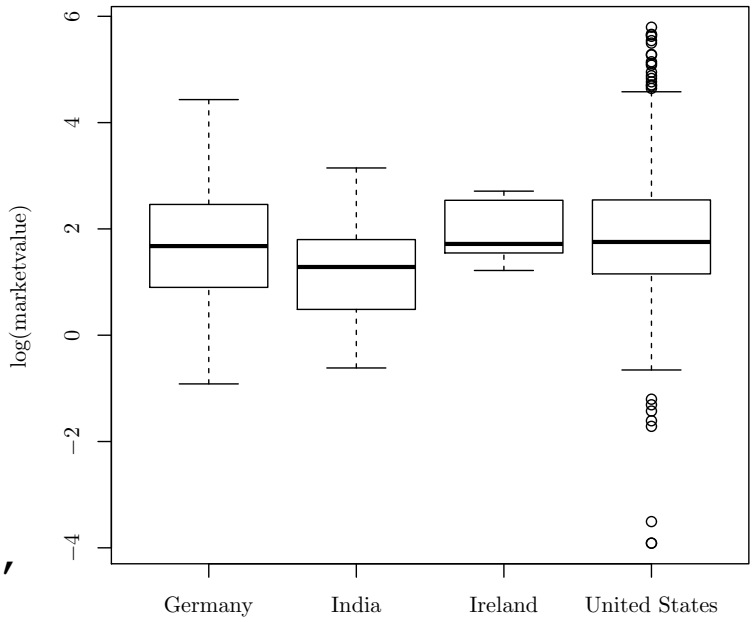
# QQ Plot

```
qqnorm(
 log(
  Forbes2000$marketvalue
))

qqline(
  log(
   Forbes2000$marketvalue),
   col=2
)
```



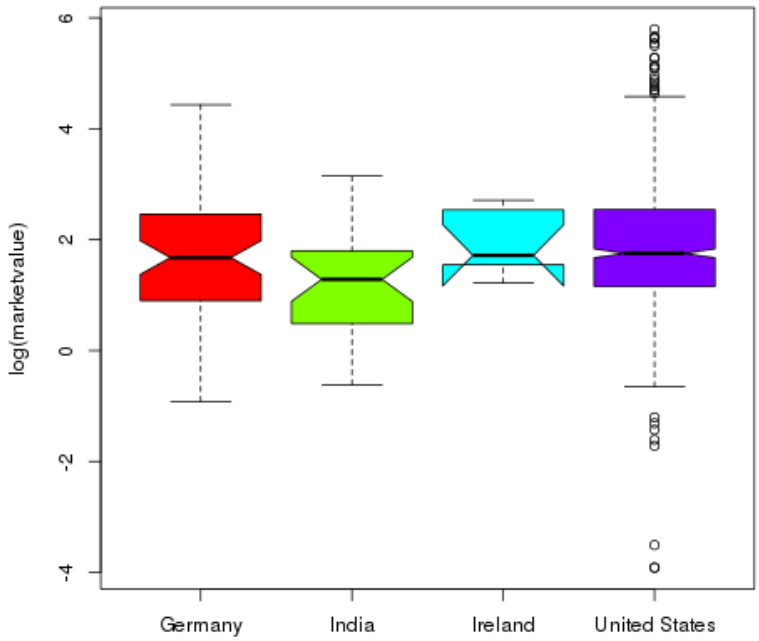
# Box Plots

```
Forbes2000.fourCountries <-
 subset(Forbes2000,
  country %in% c(
 "United States", "Germany",
 "India", "Ireland"))

Forbes2000.fourCountries <-
 droplevels(
  Forbes2000.fourCountries)

boxplot(
 log(marketvalue) ~ country,
 data = Forbes2000.fourCountries,
 ylab = "log(marketvalue)",
 varwidth = FALSE
)
```



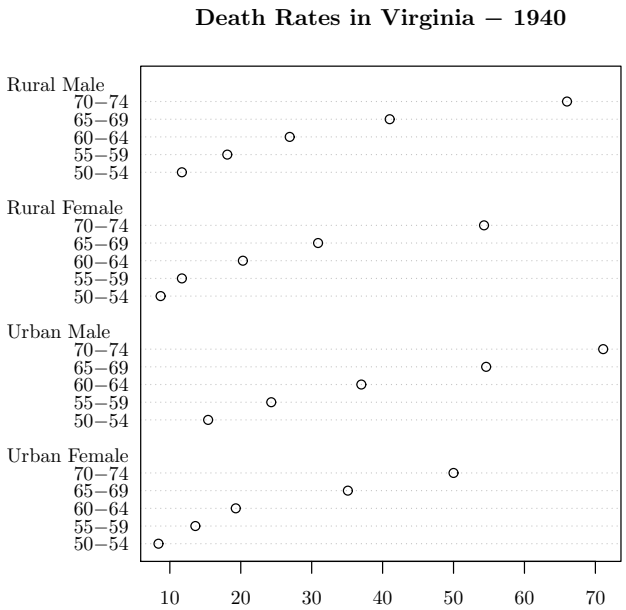
# More Box Plots

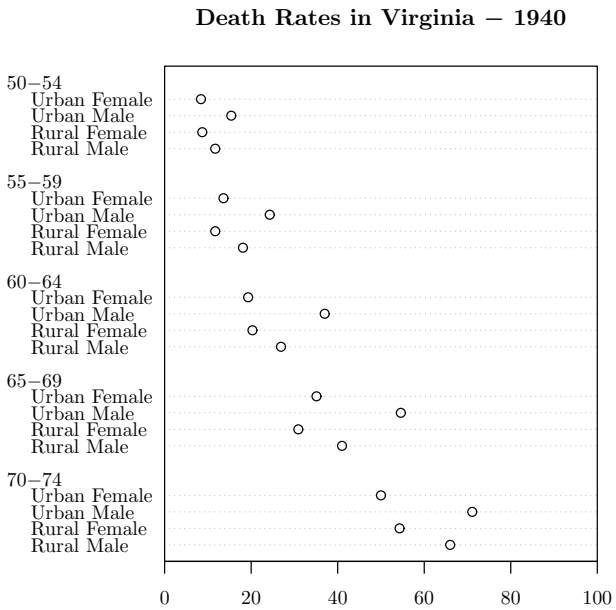
```
Forbes2000.fourCountries <-
 subset(Forbes2000,
  country %in% c(
 "United States", "Germany",
 "India", "Ireland"))

Forbes2000.fourCountries <-
 droplevels(
  Forbes2000.fourCountries)

boxplot(
 log(marketvalue) ~ country,
 data = Forbes2000.fourCountries
 ylab = "log(marketvalue)",
 notch=TRUE, col=rainbow(4),
 varwidth = FALSE
)
```



# Dot Chart

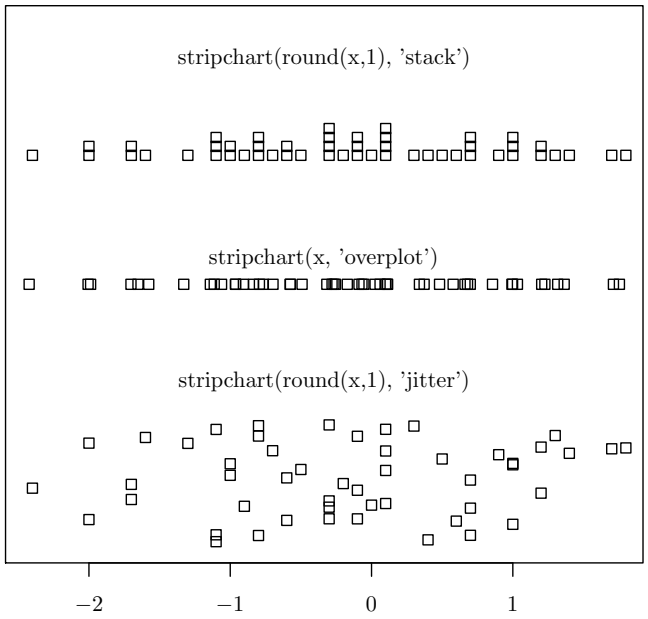
```
dotchart(VADeaths, main = "Death Rates in Virginia - 1940")

op <- par(xaxs="i") # 0 -- 100%
dotchart(t(VADeaths), xlim = c(0,100),
       main = "Death Rates in Virginia - 1940")
par(op)
```



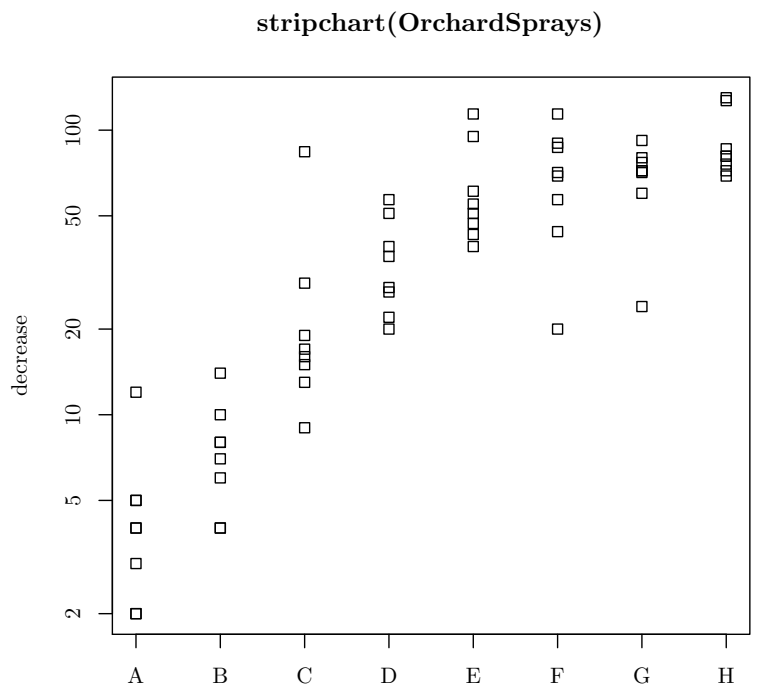
# Strip Chart

```
x <- stats::rnorm(50)
xr <- round(x, 1)

stripchart(x)
m <- mean(par("usr")[1:2])
text(m, 1.04, "stripchart(x,
 \'overplot\')")

stripchart(xr, method = "stack",
 add = TRUE, at = 1.2)
text(m, 1.35,
  "stripchart(round(x,1),
  \'stack\')")

stripchart(xr, method = "jitter",
  add = TRUE, at = 0.7)
text(m, 0.85,
  "stripchart(round(x,1),
  \'jitter\')")
```



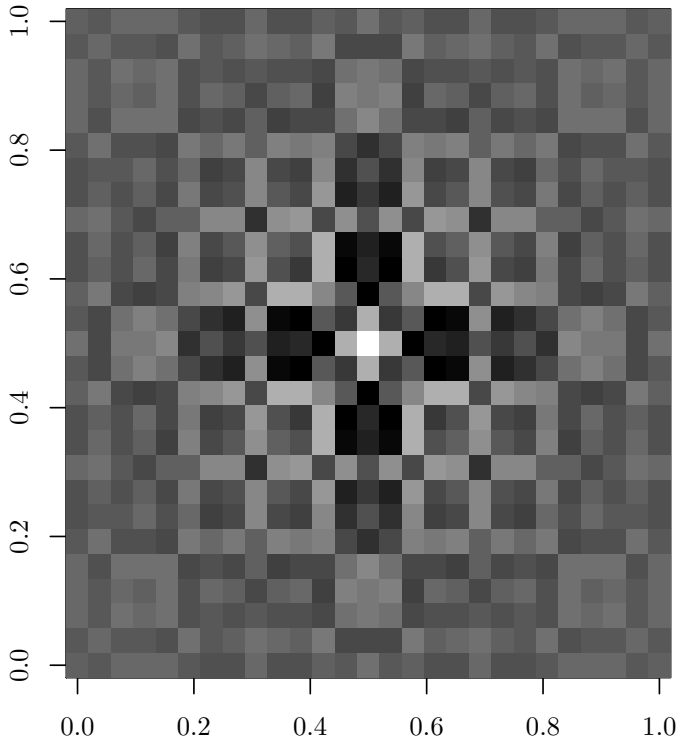
# Strip Chart

```
stripchart(decrease~treatment,
 main =
  "stripchart(OrchardSprays)",
 vertical = TRUE, log = "y",
 data = OrchardSprays
)
```



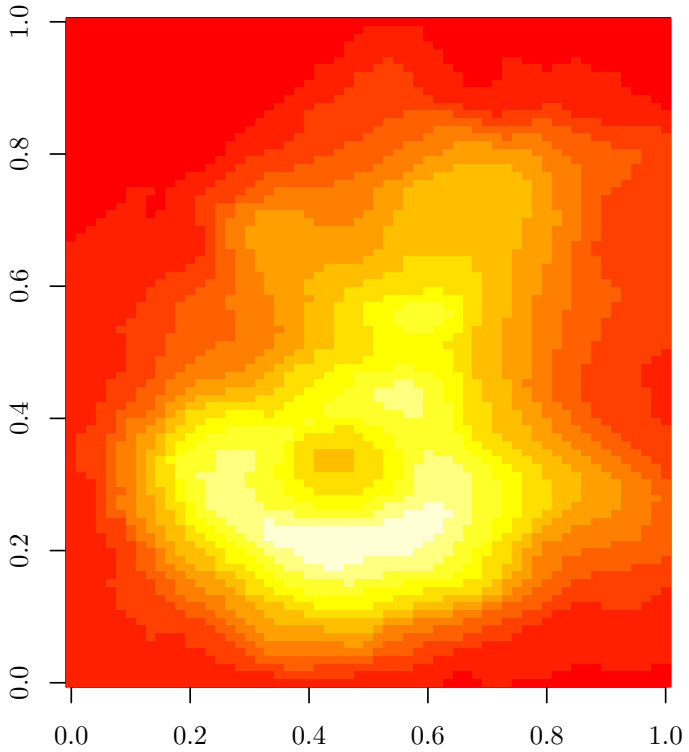
# Image

```
# For colours
require(grDevices)

x <- y <-
 seq(-4*pi,4*pi, len=27)

r <- sqrt(outer(x^2, y^2, "+"))

image(
 z = z <- cos(r^2)*exp(-r/6),
 col=gray((0:32)/32))
```



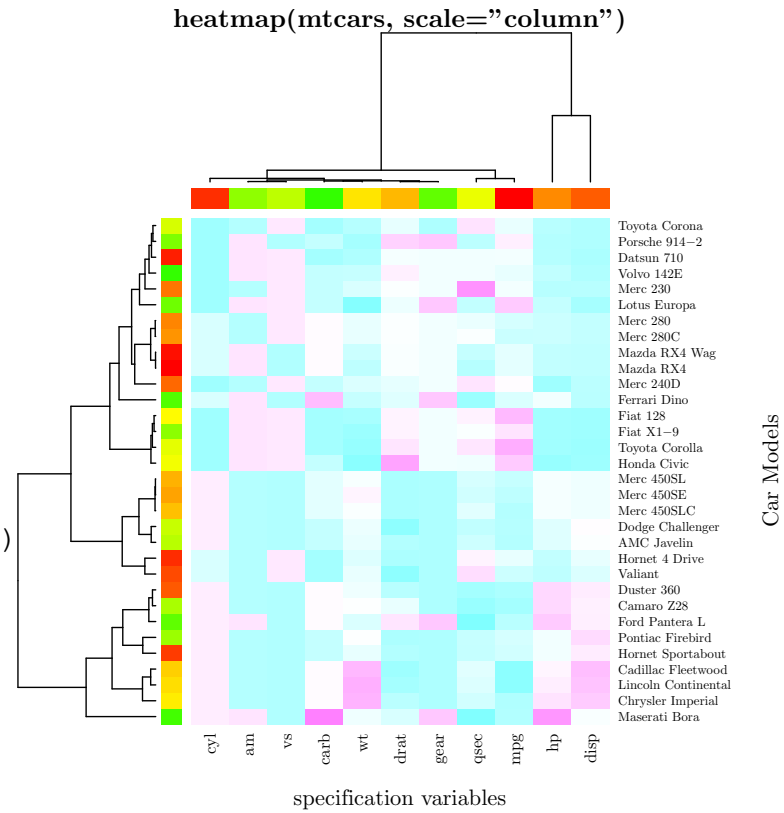
# Image

```
# Volcano data visualized as
# matrix. Need to transpose
# and flip matrix horizontally.
image(
 t(volcano)[ncol(volcano):1,])
```



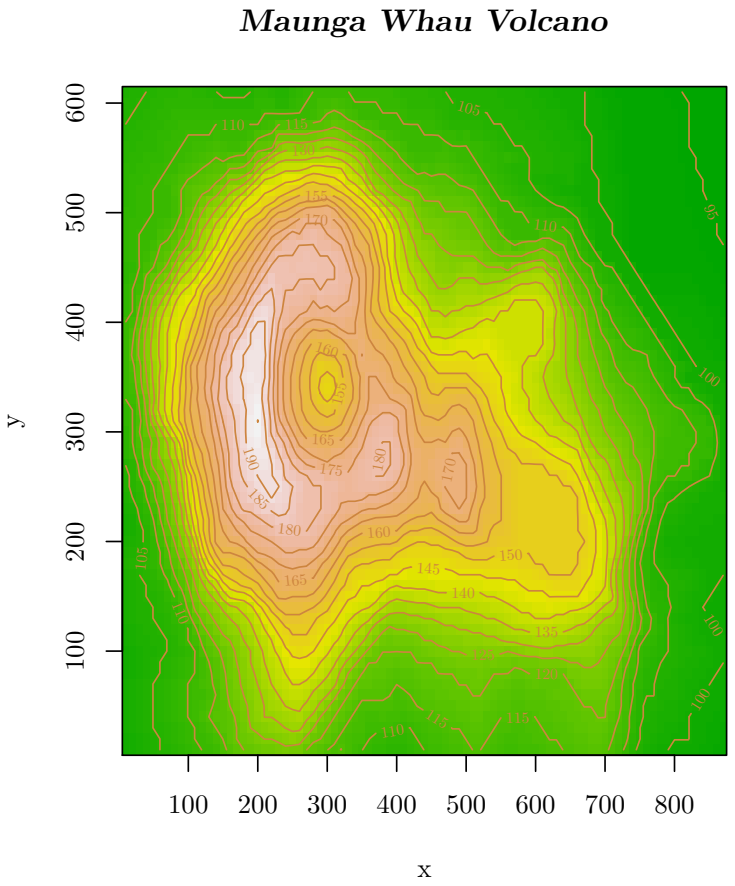
# heatmap

```
require(graphics);
require(grDevices)
x  <- as.matrix(mtcars)
rc <- rainbow(nrow(x), start=0, end=.3)
cc <- rainbow(ncol(x), start=0, end=.3)
hv <- heatmap(x,
 col = cm.colors(256),
 scale="column",
 RowSideColors = rc,
 ColSideColors = cc,
 margins=c(5,10),
 xlab = "specification variables",
 ylab= "Car Models",
 main = "heatmap(mtcars, scale=\"column\")")
```



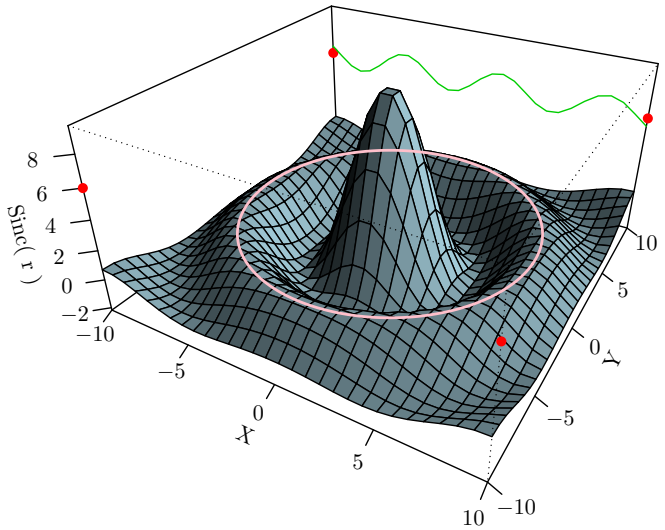
# contour

```
# A prettier display of the
# volcano
x <- 10*(1:nrow(volcano))
y <- 10*(1:ncol(volcano))
image(x, y, volcano,
 col = terrain.colors(100),
 axes = FALSE)

contour(x, y, volcano,
  levels = seq(90, 200, by=5),
  add = TRUE, col = "peru")

axis(1, at = seq(100, 800,
 by = 100))

axis(2, at = seq(100, 600,
 by = 100))
box()

title(main =
 "Maunga Whau Volcano",
 font.main = 4)
```



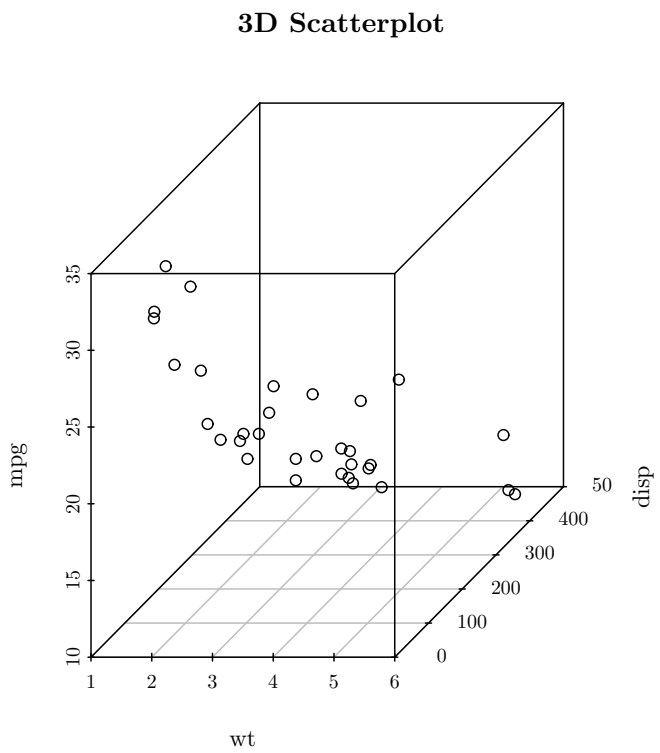
# persp and trans3d

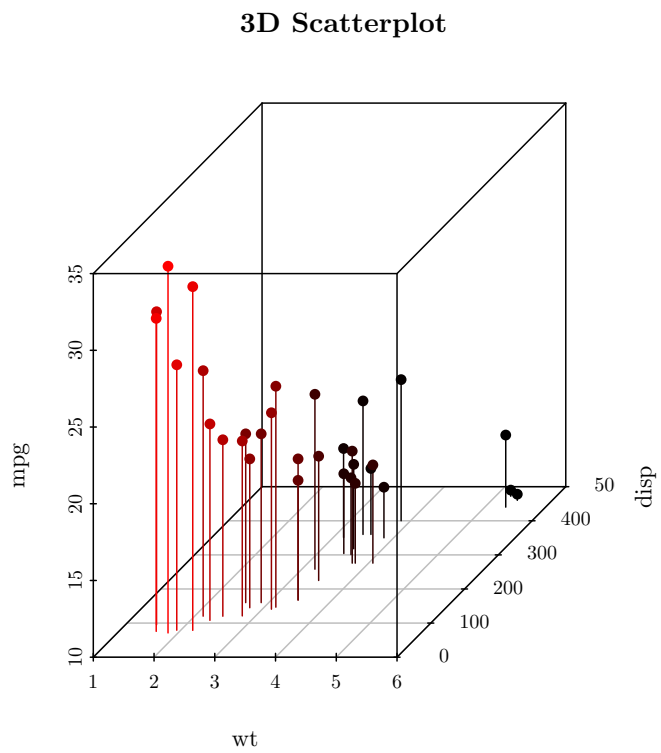
```
x <- seq(-10, 10, length= 30)
y <- x
f <- function(x,y) {
      r <- sqrt(x^2+y^2);
      10 * sin(r)/r}
z <- outer(x, y, f)
z[is.na(z)] <- 1
op <- par(bg = "transparent")

persp(x, y, z, theta = 30, phi = 30,
      expand = 0.5, col = "lightblue")
persp(x, y, z, theta = 30, phi = 30,
      expand = 0.5, col = "lightblue",
      ltheta = 120, shade = 0.75,
      ticktype = "detailed",
      xlab = "X", ylab = "Y",
      zlab = "Sinc( r )") -> res

xE <- c(-10,10); xy <- expand.grid(xE, xE)
points(trans3d(xy[,1], xy[,2], 6,
   pmat = res), col = 2, pch =16)
lines (trans3d(x, y=10, z= 6 + sin(x),
  pmat = res), col = 3)

phi <- seq(0, 2*pi, len = 201)
r1 <- 7.725 # radius of 2nd maximum
xr <- r1 * cos(phi)
yr <- r1 * sin(phi)
lines(trans3d(xr,yr, f(xr,yr), res),
 col = "pink", lwd = 2)
```



# 3D Scatterplots

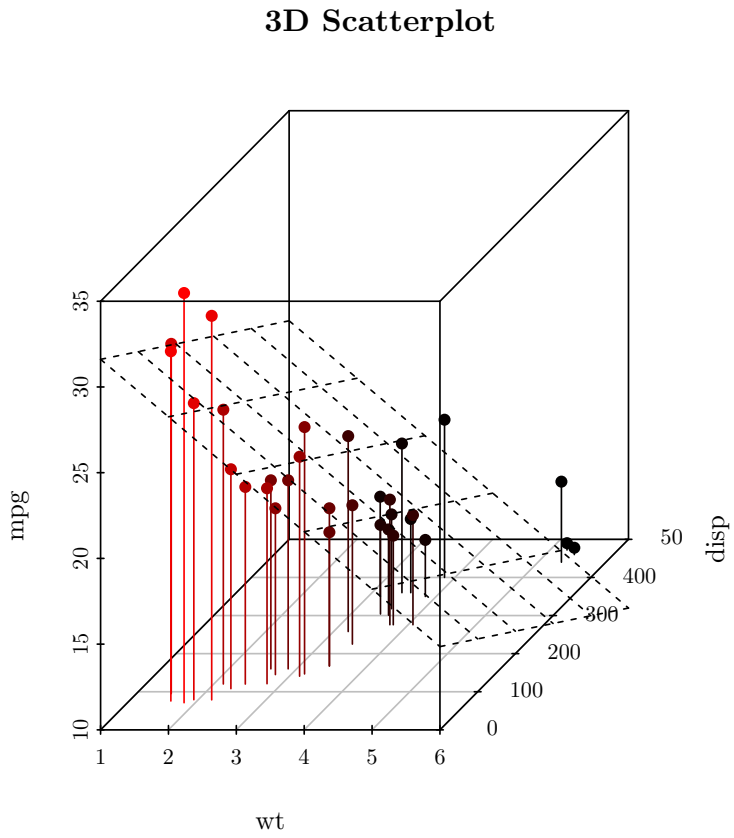
```
library(scatterplot3d)
attach(mtcars)

scatterplot3d(wt,disp,mpg, main="3D Scatterplot")

scatterplot3d(wt,disp,mpg, pch=16, highlight.3d=TRUE, type="h",
  main="3D Scatterplot")
```



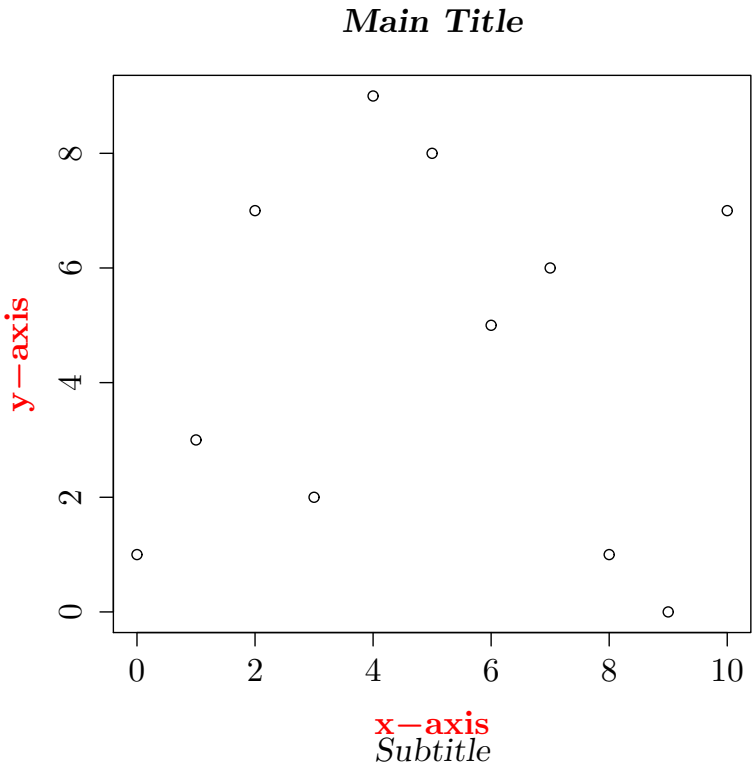
# 3D Scatterplots

```
library(scatterplot3d)
attach(mtcars)

s3d <- scatterplot3d(
 wt, disp, mpg,
 pch=16,
 highlight.3d=TRUE,
 type="h",
 main="3D Scatterplot"
)

fit <- lm(mpg ~ wt+disp)

s3d$plane3d(fit)
```



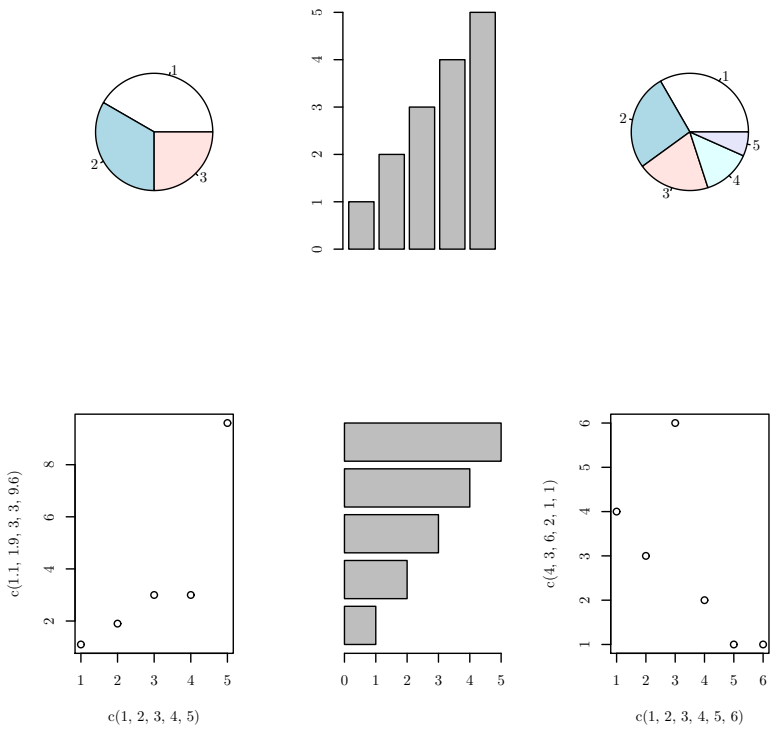
# Customizing Charts

```
# store original settings
opar <- par()

par(col.lab="red")
par(cex=1, cex.axis=1.5,
    cex.lab=1.5, cex.main=1.5,
    cex.sub=1.5)

par(font.axis=1, font.lab=2,
    font.sub=3, font.main=4)

x <- 0:10
y <- c(1,3,7,2,9,8,5,6,1,0,7)

plot(x, y,
  xlab="x-axis", ylab="y-axis",
  main="Main Title",
  sub="Subtitle")

# restore original settings
par(opar)
```



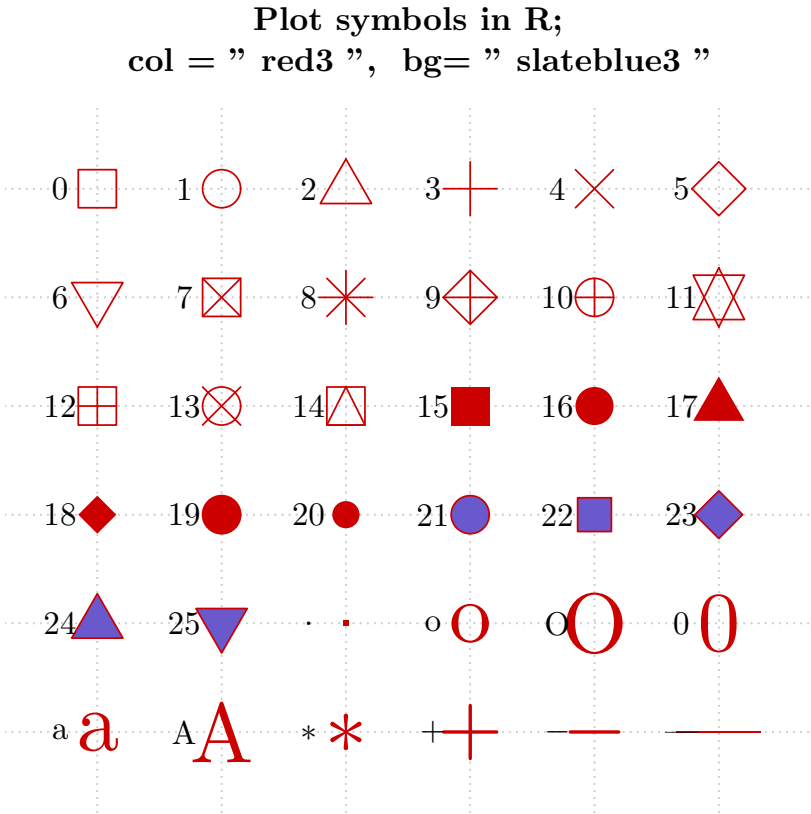
# Multi Plots

```
png("multiplot.png",
 bg="transparent",
 width=500,
 height=300)

par(mfcol=c(2,3))
pie(c(5,4,3))
plot(x=c(1,2,3,4,5),
 y=c(1.1, 1.9, 3, 3, 9.6))

barplot(c(1,2,3,4,5))

barplot(c(1,2,3,4,5),
 horiz=TRUE)

pie(c(5,4,3,2,1))

plot(c(1,2,3,4,5,6),
     c(4,3,6,2,1,1))

dev.off()
```



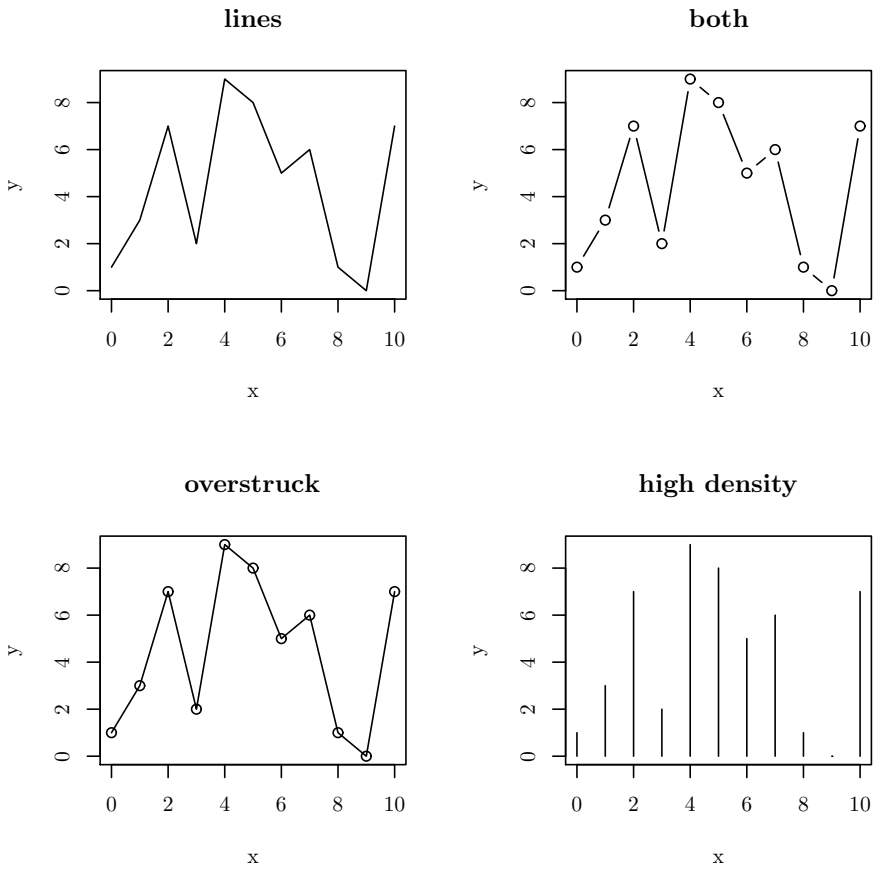
# Symbols

```
plot(x,
     y,
     pch=0,
     xlab="X",
     ylab="Y"
)
```



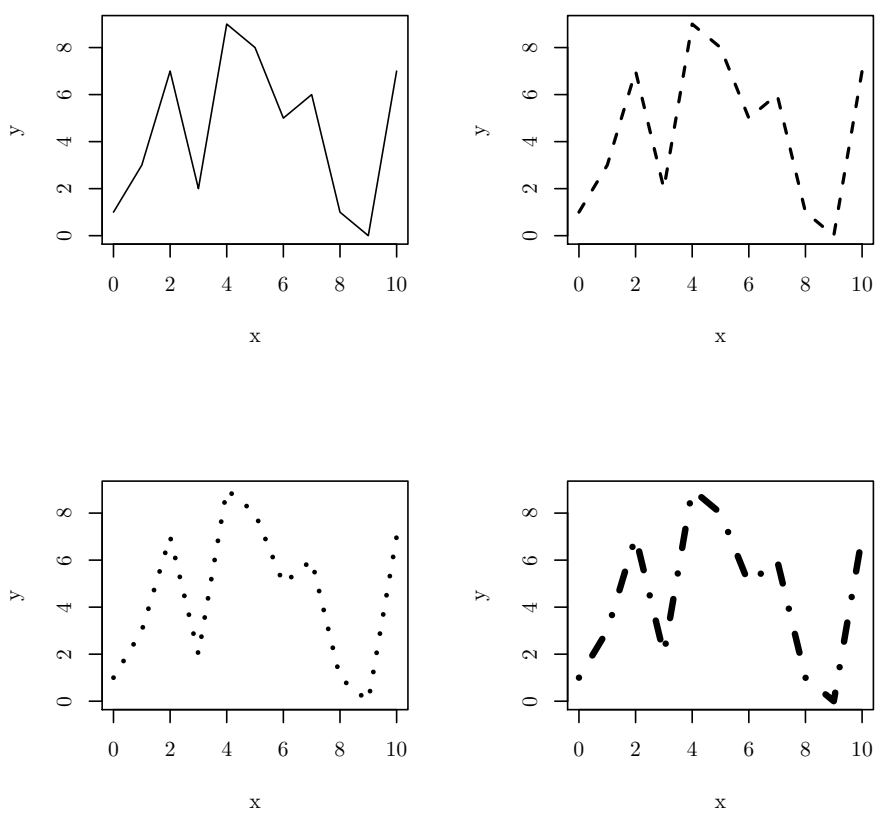
# Line Types

```
x = 0:10;
y <- c(1,3,7,2,9,8,5,6,1,0,7);

par(mfrow=c(2,2))
plot(x,y, type="l");
title("lines")

plot(x,y, type="b");
title("both")

plot(x,y, type="o");
title("overstruck")

plot(x,y, type="h");
title("high density")
```



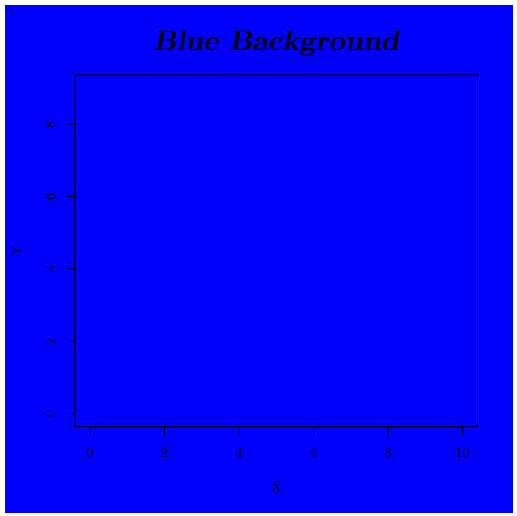
# Line Types

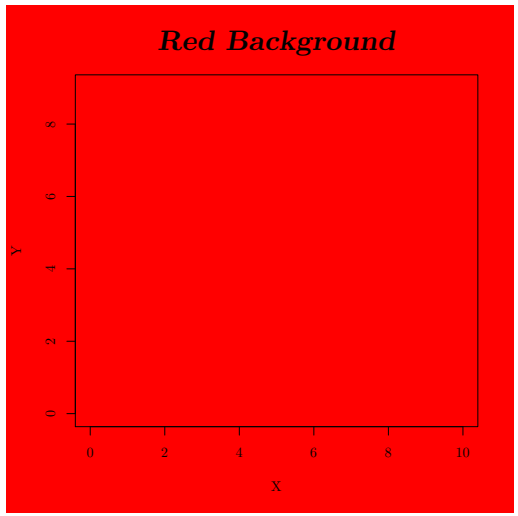
```
x = 0:10;
y <- c(1,3,7,2,9,8,5,6,1,0,7);

par(mfrow=c(2,2))
plot(x,y, type="l",
 lty=1, lwd=1)

plot(x,y, type="l",
 lty=2, lwd=2)

plot(x,y, type="l",
 lty=3, lwd=3)

plot(x,y, type="l",
 lty=4, lwd=4)
```



# Background Colour

```
par(def.par)

par("bg")
[1] "transparent"

par(bg="blue")
par("bg")
[1] "blue"

plot(x, y, type="n", xlab="X", ylab="Y")
title("Blue Background", cex.main = 2, font.main= 4, col.main= "black")
```

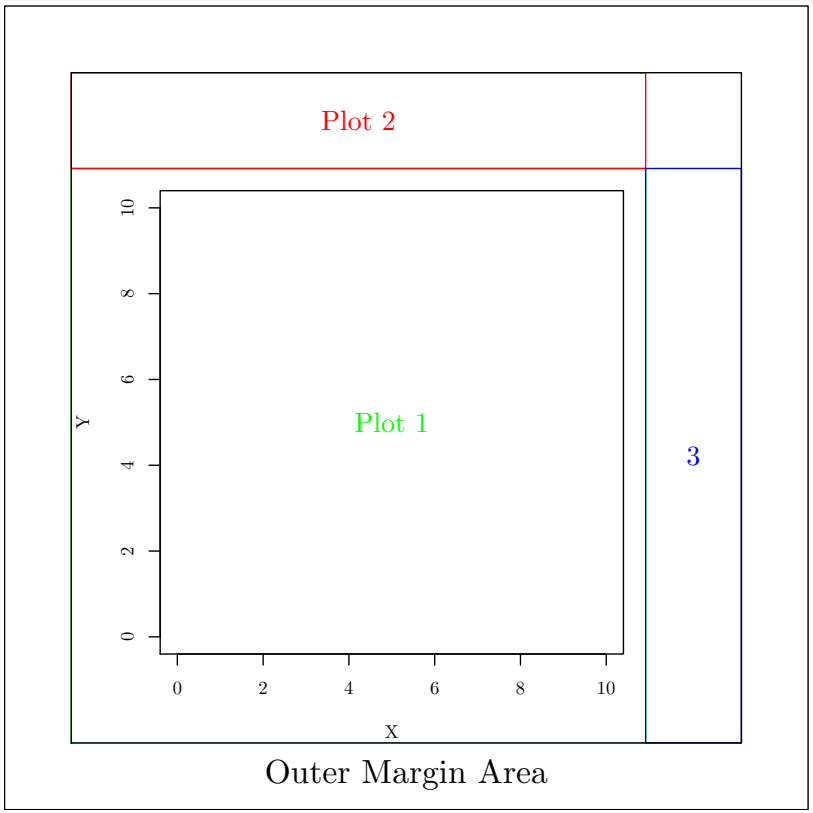


# Margins

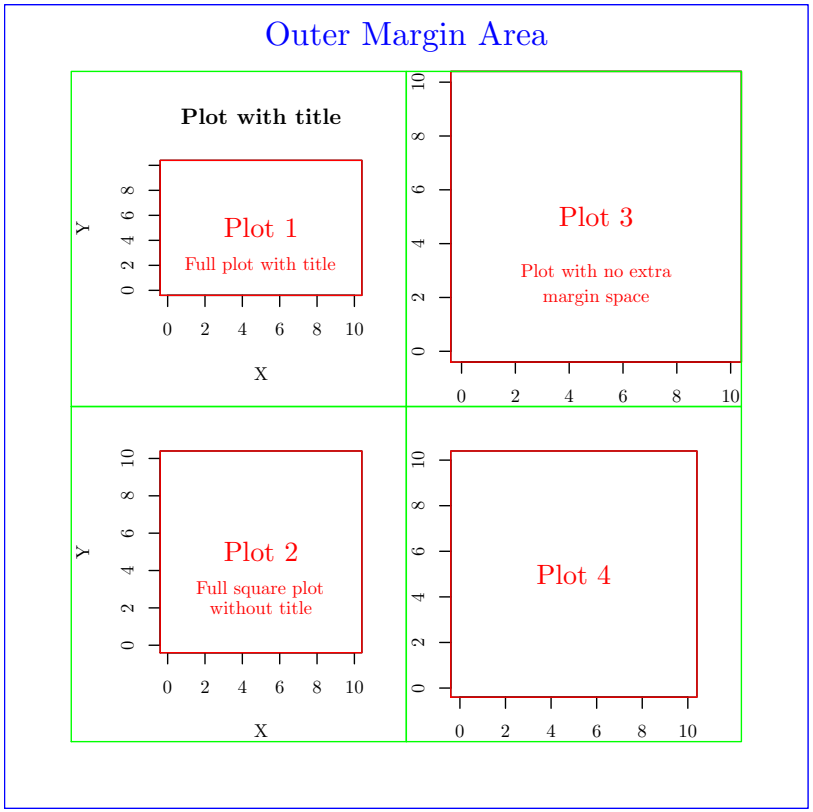
```
par(oma=c(3,3,3,3))
nf <- layout(matrix(c(2,0,1,3),2,2,
 byrow=TRUE), widths=c(6,1),
 heights=c(1,6), TRUE)

# First plot, goes in position one, so
#  lower left. This is our large plot.
par(mar=c(4,4,1,1))
plot(0:10, 0:10, type="n", xlab="X",
 ylab="Y", col="green")
box("figure", col="green")
text(5,5,"Plot 1", col="green", cex=1.5)

# Second plot, top left
par(mar=c(1,1,1,1))
plot(0:10, 0:10, type="n", xlab="X",
 ylab="Y", axes=FALSE, col="green")
box("figure", col="red")
text(5,5,"Plot 2", col="red", cex=1.5)

# Third plot, bottom right.
par(mar=c(1,1,1,1))
plot(0:10, 0:10, type="n", xlab="X",
 ylab="Y", axes=FALSE, col="green")
box("figure", col="blue")
text(5,5,"3", col="blue", cex=1.5)
```

```
# Label the outer margin area to show it
#   is still there
mtext("Outer Margin Area", side=1, line=1,
 cex=1.5, col="black", outer=TRUE)
box("inner", col="black")

box("outer", col="black")
```



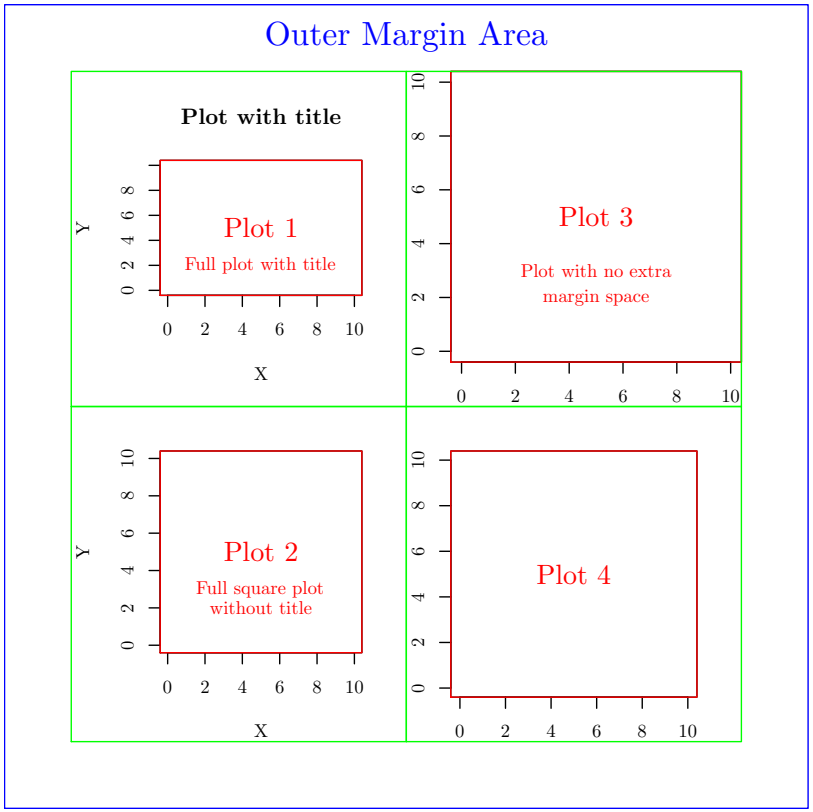
# Multiplot

```
par(mfcol=c(2,2))
par(mar=c(5,4,4,2)); par(oma=c(3,3,3,3))

# Main Plot
plot(x, y, main="Plot with title",
 type="n", xlab="X", ylab="Y")
text(5,5, "Plot 1", col="red", cex=1.5)
text(5,2, "Full plot with title",
 col="red", cex=1)
box("plot", col="red")
box("figure", col="green")

# Plot 1
plot(x, y, main="Plot with title",
 type="n", xlab="X", ylab="Y")
text(5,5, "Plot 1", col="red", cex=1.5)
text(5,2, "Full plot with title",
 col="red", cex=1)
box("plot", col="red")
box("figure", col="green")

# Plot 2: Altering the margins
par(mar=c(4,4,2,2))
plot(x, y, type="n", xlab="X", ylab="Y")
text(5,5, "Plot 2", col="red", cex=1.5)
text(5,3, "Full square plot", col="red",
 cex=1)
text(5,2, "without title", col="red",
 cex=1)
box("plot", col="red")
box("figure", col="green")
```



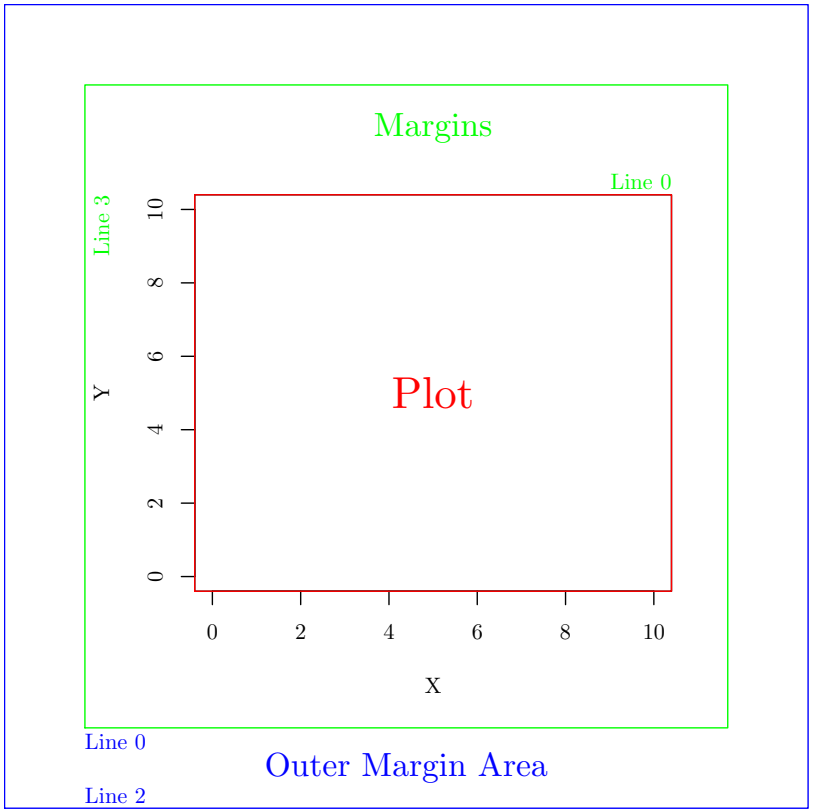
# Multiplot

```
# Plot 3: Altering the margins
par(mar=c(2,2,0,0))
plot(x, y, type="n", xlab="X", ylab="Y",
 axes=)
text(5,5, "Plot 3", col="red", cex=1.5)
text(5,3, "Plot with no extra", col="red",
 cex=1)
text(5,2, "margin space", col="red",
 cex=1)
box("plot", col="red")
box("figure", col="green")

# Plot 4: Altering the margins
par(mar=c(2,2,2,2))

#  Notice that the X and Y labels are
#   gone, they didn't fit into the space
#   provided for the graphic so they
#   simply were covered.
plot(0:10, 0:10, type="n", xlab="X",
 ylab="Y")
text(5,5, "Plot 4", col="red", cex=1.5)
box("plot", col="red")
box("figure", col="green")

# Label the outer margin area so we can
# see where it is
mtext("Outer Margin Area", side=3, line=1,
 cex=1.5, col="blue", outer=TRUE)
box("outer", col="blue")
```



# Margin Text

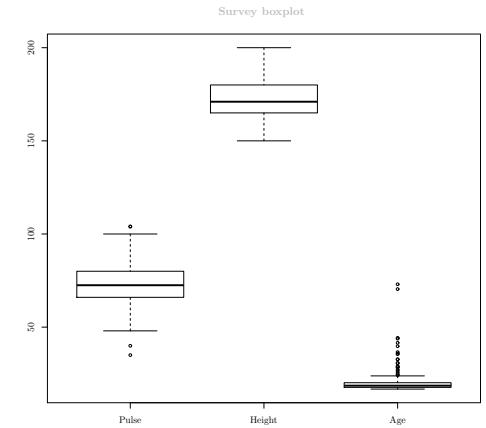
```
par(mar=c(5,4,4,2) + 0.1)
par(oma=c(3,3,3,3))

plot(x, y, type="n", xlab="X", ylab="Y")
text(5,5, "Plot", col="red", cex=2)
box("plot", col="red")

mtext("Margins", side=3, line=2,
  cex=2, col="green")
mtext("Line 0", side=3, line=0, adj=1.0,
  cex=1, col="green")

mtext("Line 3", side=2, line=3, adj=1.0,
  cex=1, col="green")
box("figure",col="green")

mtext("Outer Margin Area", side=1, line=1,
  cex=2, col="blue", outer=TRUE)
mtext("Line 0", side=1, line=0, adj=0.0,
  cex=1, col="blue", outer=TRUE)
mtext("Line 2", side=1, line=2, adj=0.0,
  cex=1, col="blue", outer=TRUE)
box("outer", col="blue")
```

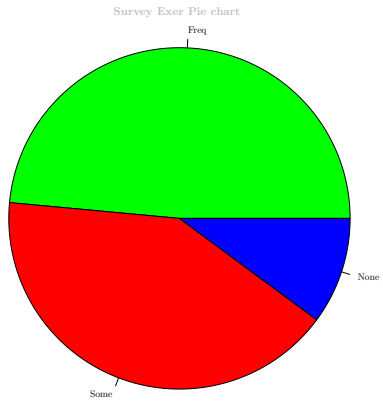


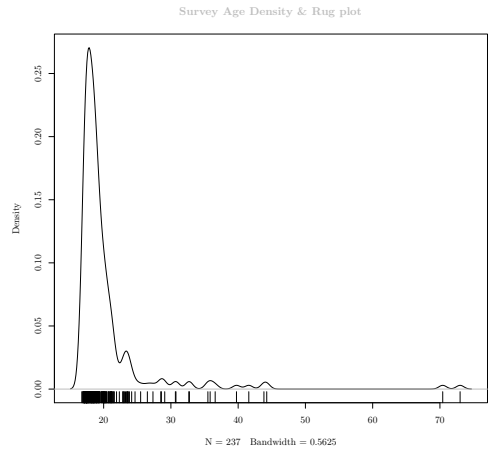
# Practical

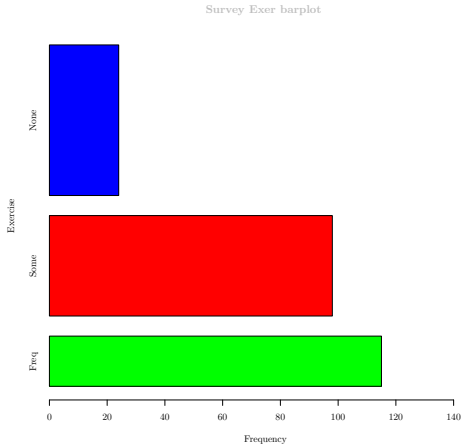
Can you recreate the following diagram?

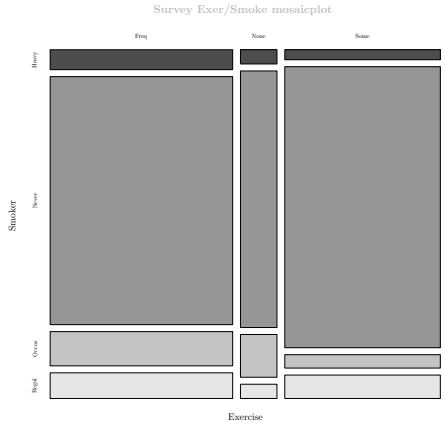
```
install.packages("MASS")
library("MASS"); names(survey)
```

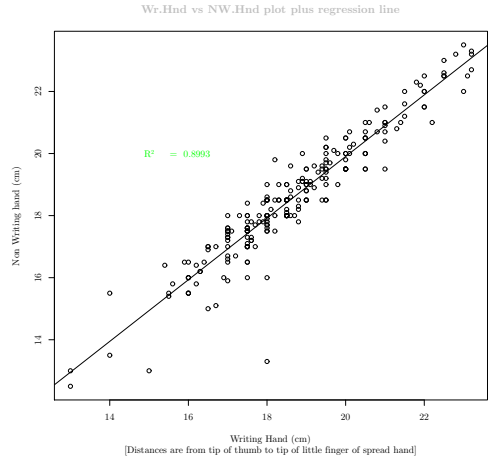


# Lattice Graphics

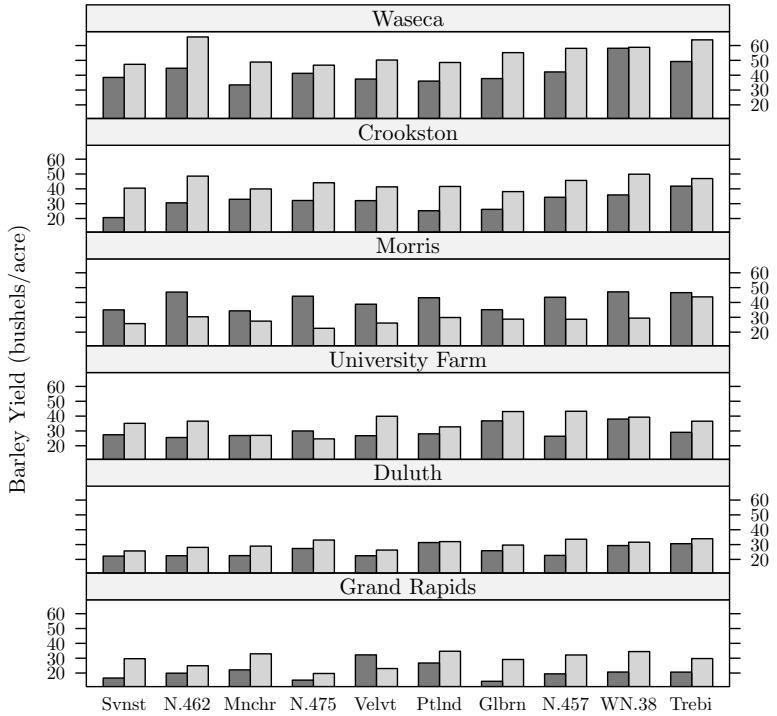
ICHEC
Irish Centre for High-End Computing

- Lattice package is an implementation of the Trellis graphics.
- Different from standard graphics.
- Easier to plot multiple graphs on the same page or superimpose graphs.
- Produce clean, readable output by default.

| General | Trellis |
|---|---|
| barplot | barchart |
| dotchart | dotplot |
| hist | histogram |
| density | densityplot |
| stripchart | stripplot |
| qqnorm | qqmath |
| xplot | xyplot |
| qqplot | qq |
| pairs | splom |
| image | levelplot |
| contour | contourplot |
| persp | cloud, wireframe |

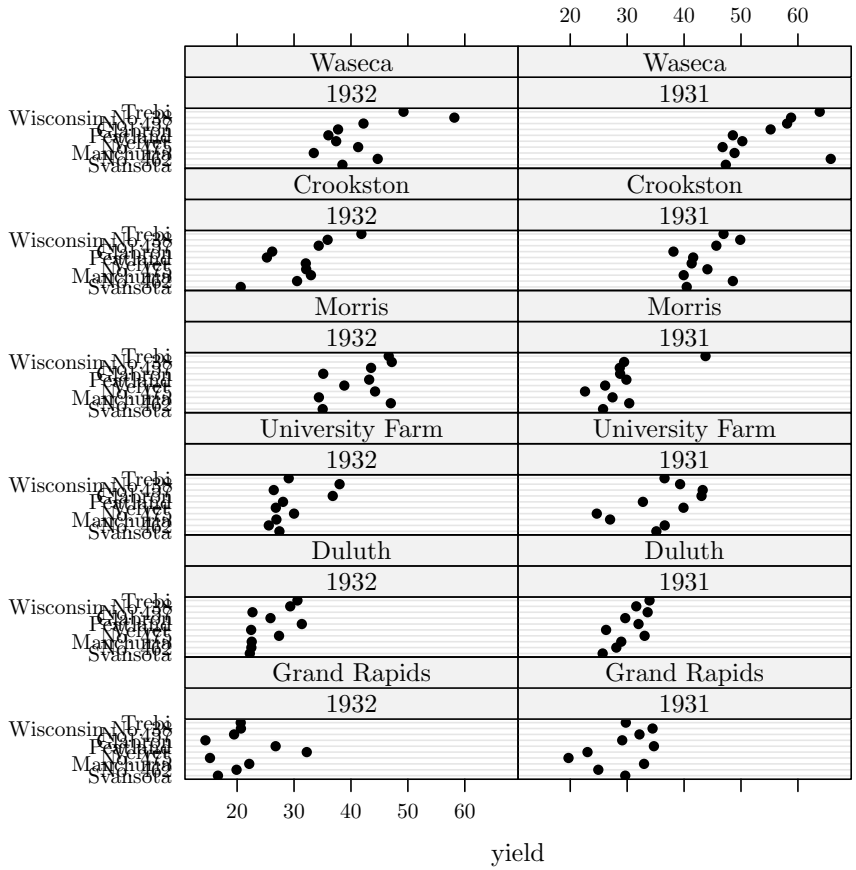
# Univariate: Barchart

```
library(lattice)
require(stats)

barchart(
 # y~x|z
 yield ~ variety | site,
 data = barley,
 groups = year,
 layout = c(1,6),
 ylab =
 "Barley Yield (bushels/acre)",
 scales = list(
  x = list(abbreviate = TRUE,
  minlength = 5)))
```



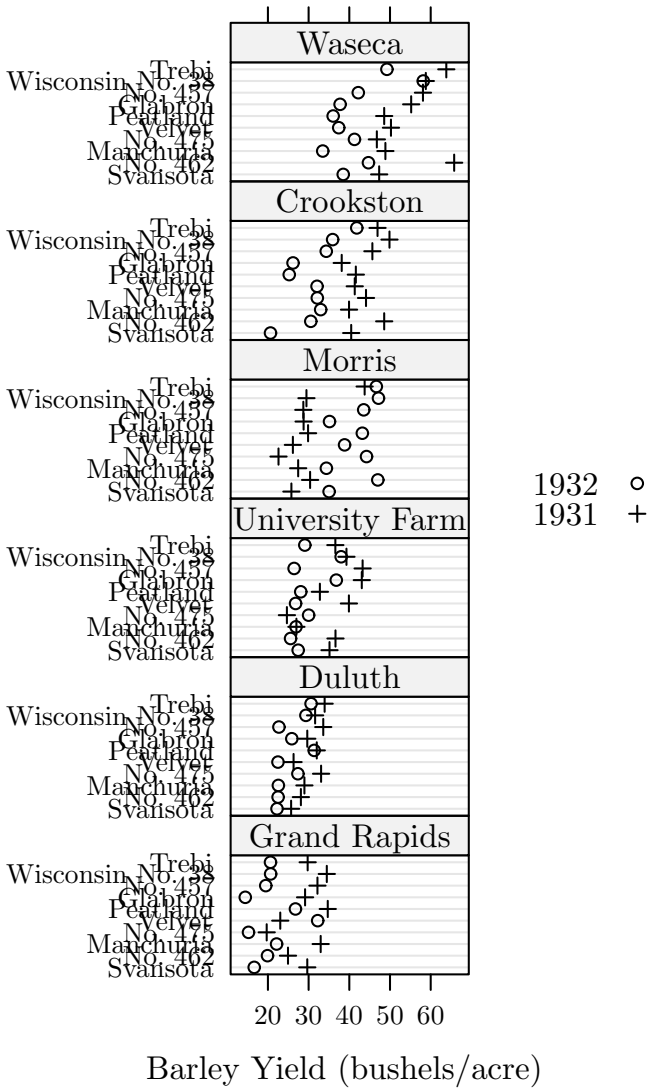
# Univariate: dotplot

```
dotplot(
 variety ~ yield | year * site,
 data=barley)
```



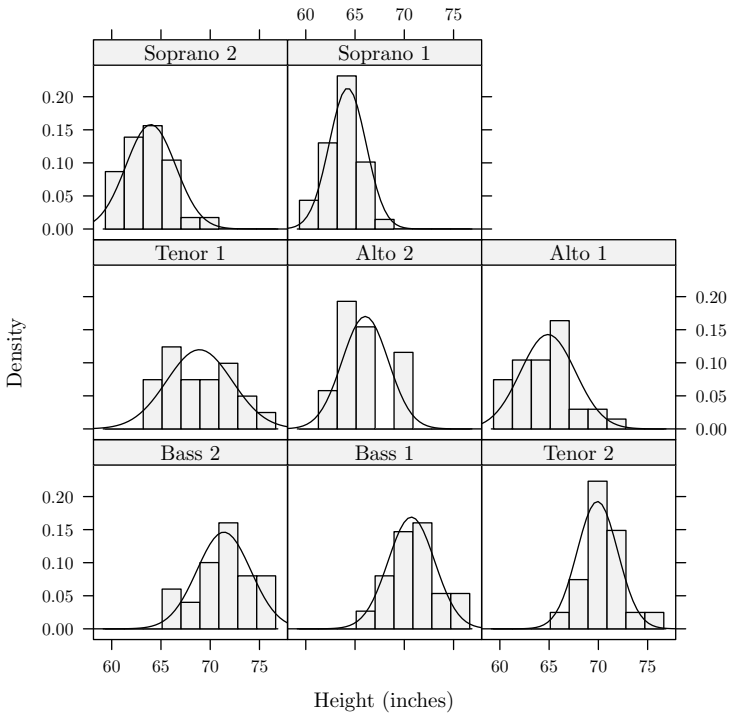
# Univariate: dotplot

```
dotplot(variety ~ yield | site,
  data = barley,
  groups = year,
  key = simpleKey(
   levels(barley$year),
   space = "right"),
  xlab = "Barley Yield (bushels/acre) ",
  ylab=NULL,
  aspect=0.5,
  layout = c(1,6))
```



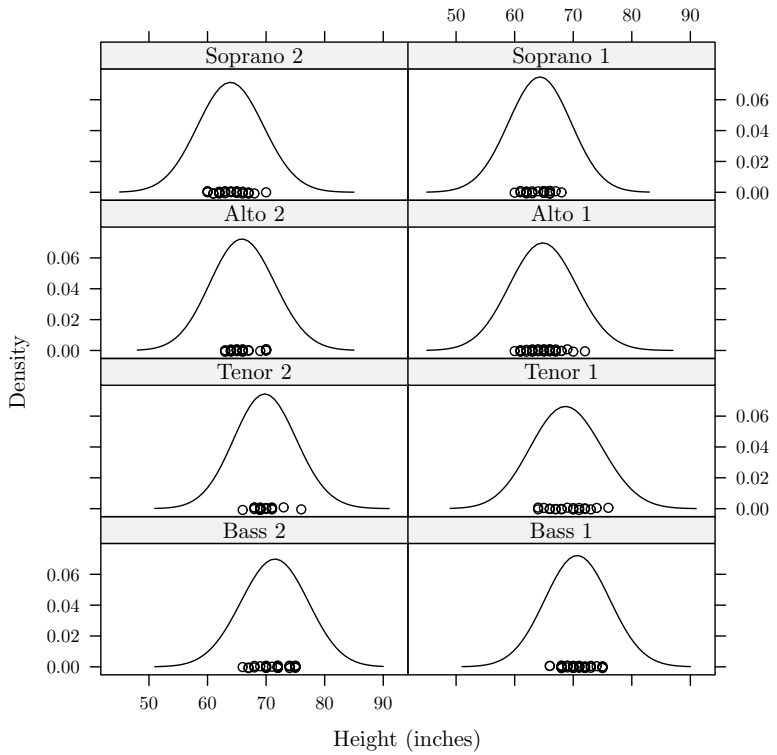
# Univariate: histogram

```
histogram( ~ height | voice.part,
     data = singer,
     xlab = "Height (inches)",
     type = "density",
     panel = function(x, ...) {
         panel.histogram(x, ...)
         panel.mathdensity(dmath = dnorm,
                 col = "black",
                args = list(mean=mean(x),
                sd = sd(x)))
     } )
```



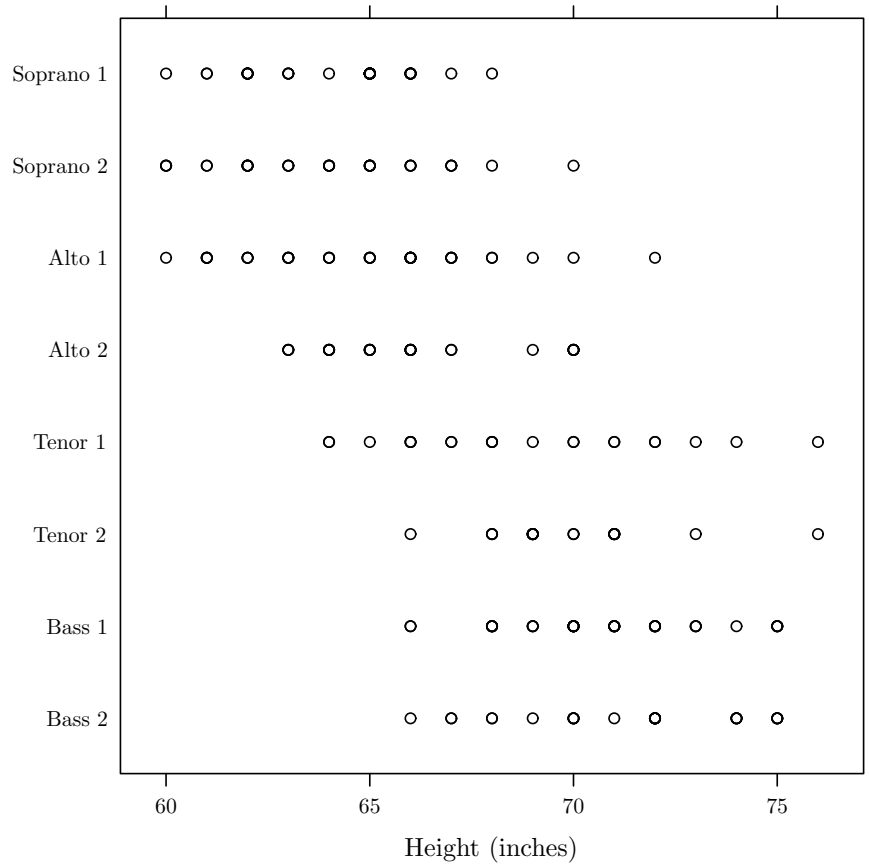
# Univariate: densityplot

```
densityplot( ~ height | voice.part,
       data = singer,
       layout = c(2, 4),
       xlab = "Height (inches)",
       bw = 5)
```



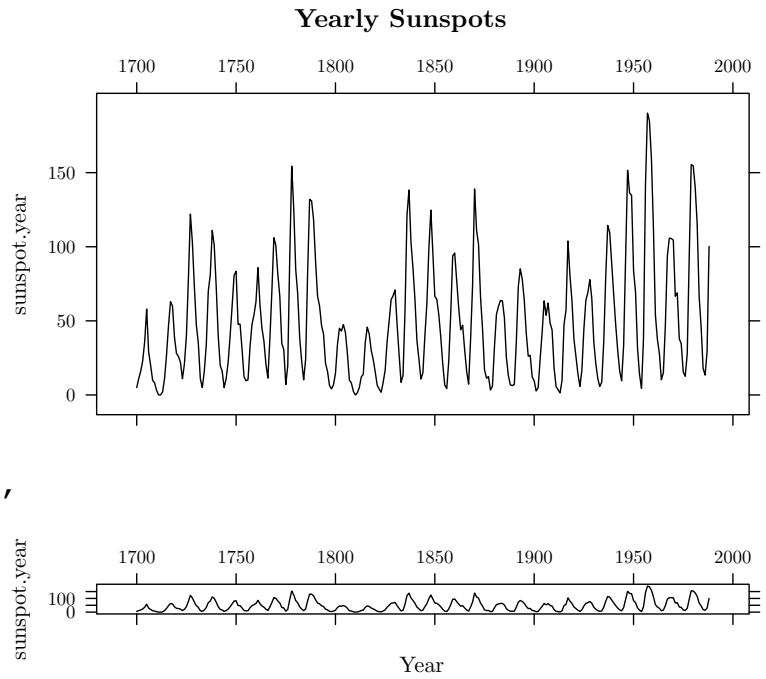
# Univariate: stripplot

```
stripplot(voice.part ~ height,
  data=singer,
  xlab="Height (inches)"
)
```



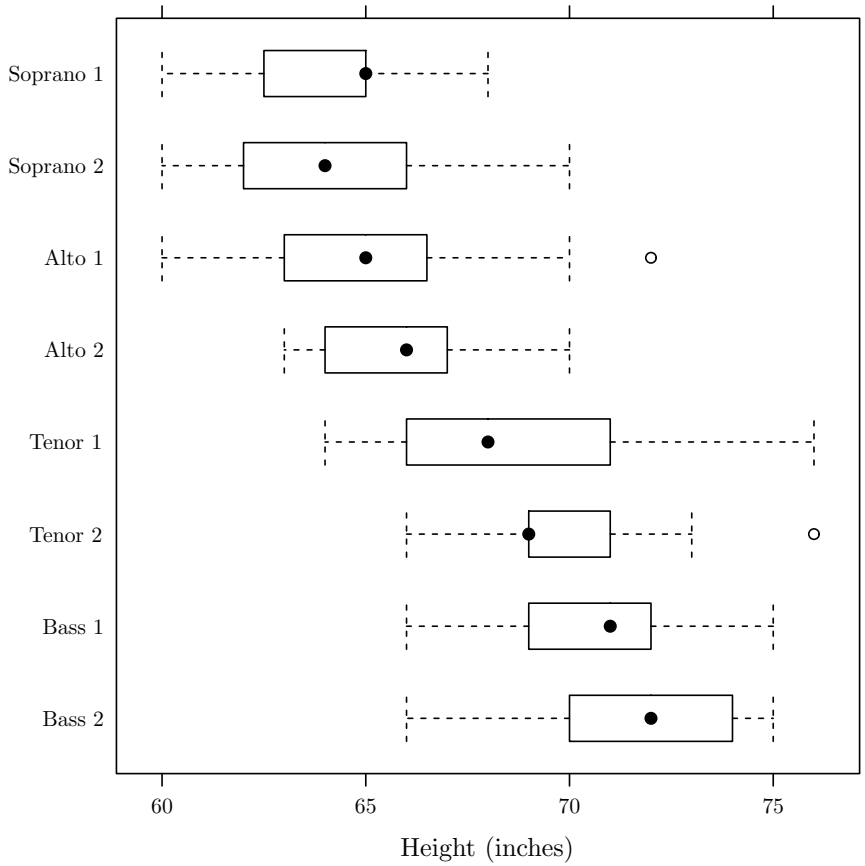
# Bivariate: xyplot

```
trellis.par.set(background =
  list(col = "transparent"))

plot <- xyplot(sunspot.year ~ 1700:1988,
          xlab = "",
          type = "l",
          scales = list(
            x = list(alternating = 2)
                  ),
          main = "Yearly Sunspots")

print(plot, position = c(0, .3, 1, .9),
  more = TRUE)

print(update(plot, aspect = "xy", main = "",
  xlab = "Year"),
  position = c(0, 0, 1, .3)
)
```



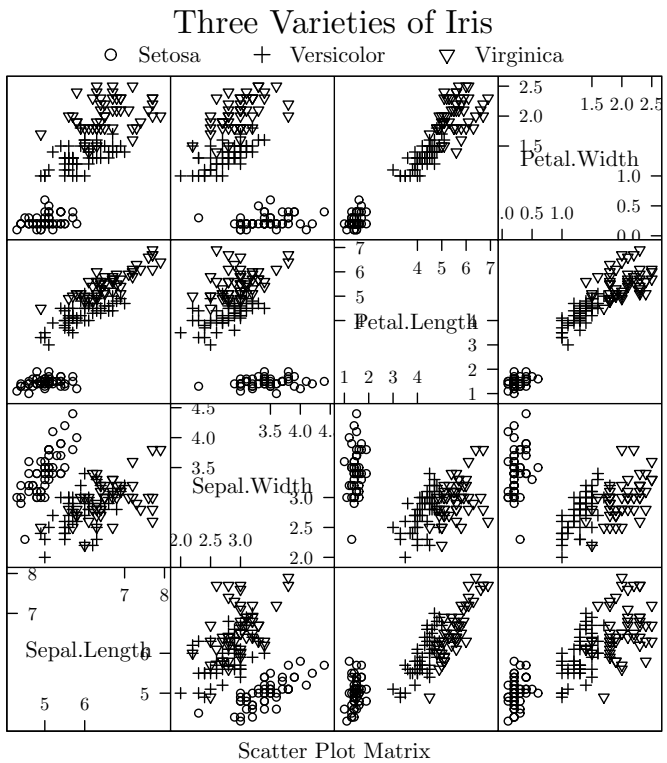
# Bivariate: bwplot

```
bwplot(voice.part ~ height,
    data=singer,
    xlab="Height (inches)"
)
```



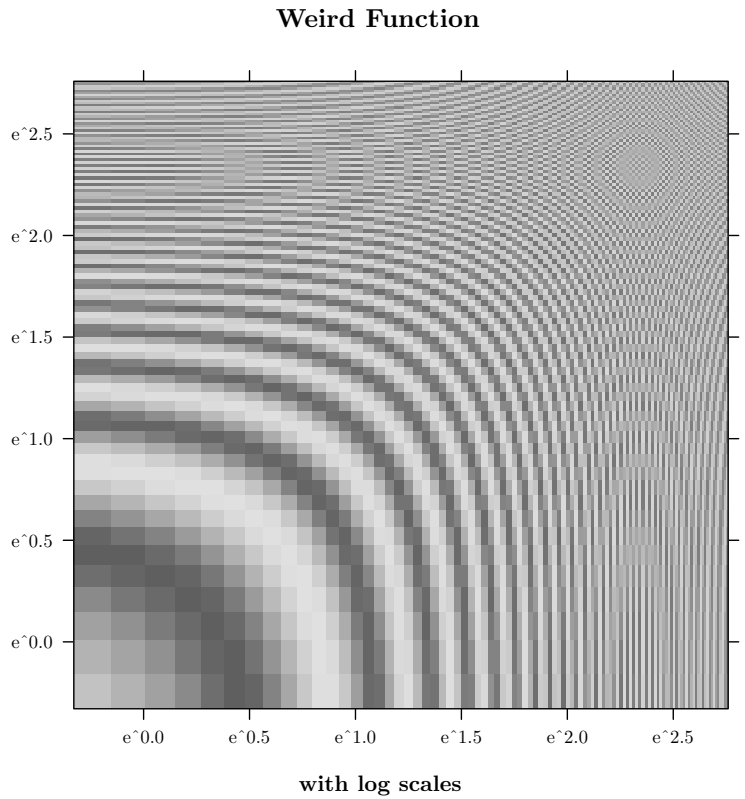
# Bivariate: splom

```
super.sym <- trellis.par.get(
  "superpose.symbol"
)

# Sepal.Length Sepal.Width
# Petal.Length Petal.Width
splom(~iris[1:4],
    groups = Species,
    data = iris,
    panel = panel.superpose,
    key = list(
     title = "Three Varieties of Iris",
     columns = 3,
     points = list(
      pch = super.sym$pch[1:3],
      col = super.sym$col[1:3]),
     text = list(c("Setosa",
       "Versicolor", "Virginica"))
     )
)
```



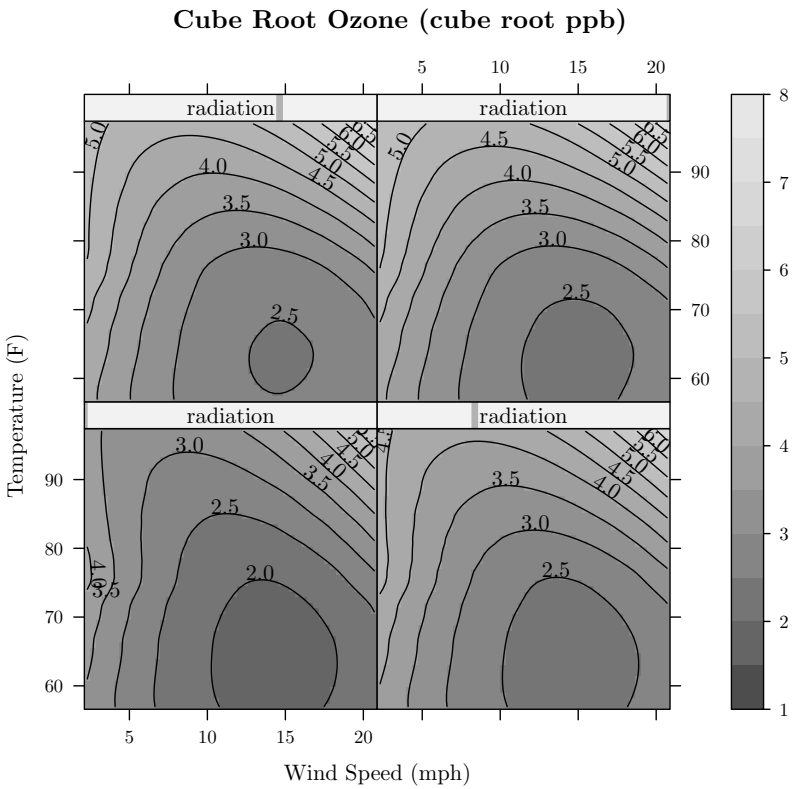
# Trivariate: levelplot

```
x <- seq(pi/4, 5 * pi, length.out = 100)
y <- seq(pi/4, 5 * pi, length.out = 100)
r <- as.vector(sqrt(outer(x^2, y^2, "+")))
grid <- expand.grid(x=x, y=y)
grid$z <- cos(r^2) * exp(-r/(pi^3))

levelplot(z~x*y, grid, cuts = 50,
     scales=list(log="e"), xlab="",
     ylab="", main="Weird Function",
     sub="with log scales",
     colorkey = FALSE, region = TRUE)
```



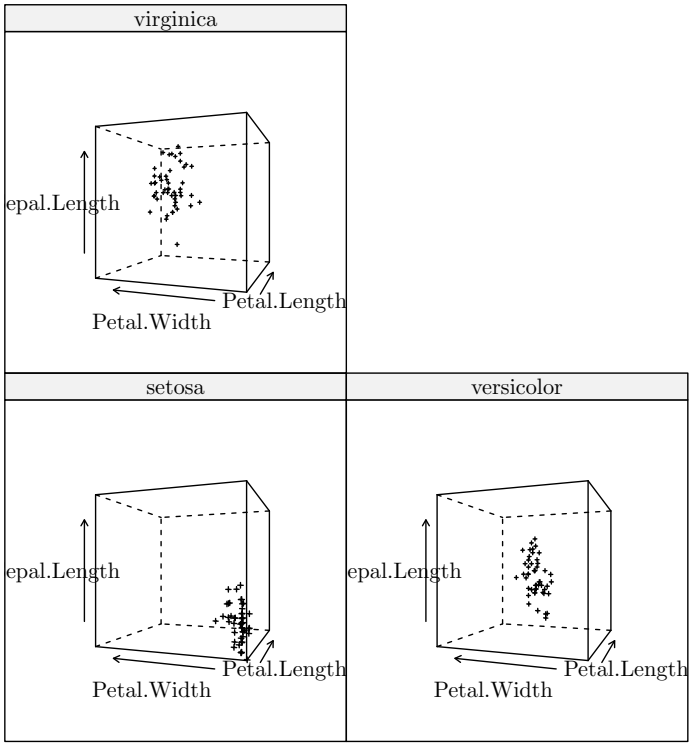
# Trivariate: contourplot

```
attach(environmental)
ozo.m <- loess((ozone^(1/3)) ~
      wind * temperature * radiation,
  parametric = c("radiation", "wind"),
                  span = 1, degree = 2)

w.marginal <- seq(min(wind),
          max(wind), length = 50)

t.marginal <- seq(min(temperature),
          max(temperature), length = 50)

r.marginal <- seq(min(radiation),
          max(radiation), length = 4)

wtr.marginal <- list(wind = w.marginal,
                 temperature = t.marginal,
                 radiation = r.marginal)

grid <- expand.grid(wtr.marginal)
grid[, "fit"] <- c(predict(ozo.m, grid))

contourplot(
     fit ~ wind * temperature | radiation,
     data = grid,
     cuts = 10, region = TRUE,
     xlab = "Wind Speed (mph)",
     ylab = "Temperature (F)",
     main = "Cube Root Ozone (cube root ppb)"
)
```



# Trivariate: cloud

```
cloud(Sepal.Length ~ Petal.Length *
      Petal.Width | Species, data = iris,
  screen = list(x = -90, y = 70),
  distance = 0.4,
  zoom = 0.6
)
```



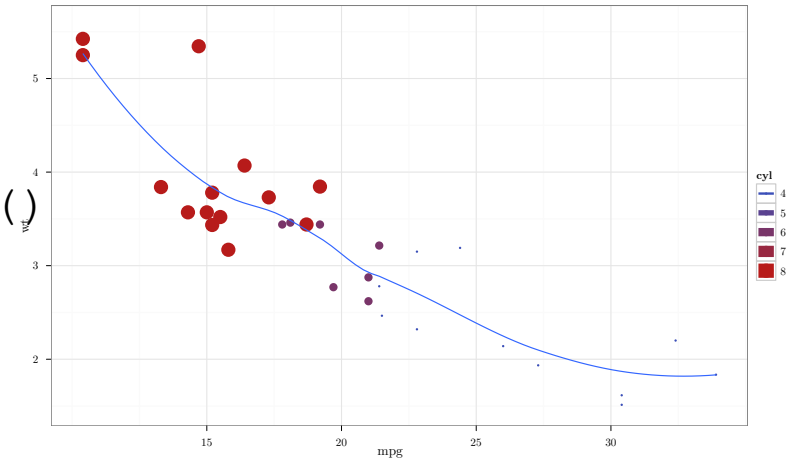
# ggplot2: Background

ICHEC
Irish Centre for High-End Computing

- Unlike other R graphics packages, it has a deep underlying grammar based on the Grammar of Graphics.
- Easy to learn, hassle-free plots

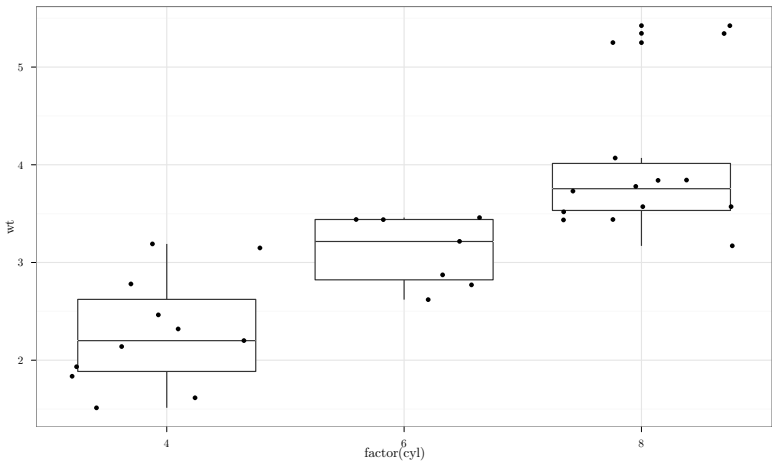
# ggplot2

```
library(ggplot2)
attach(mtcars)

ltheme <- theme_set(theme_bw())

qplot(mpg,
  wt,
  data=mtcars,
  geom=c("point", "smooth"),
  colour=cyl,
  size=cyl)
```



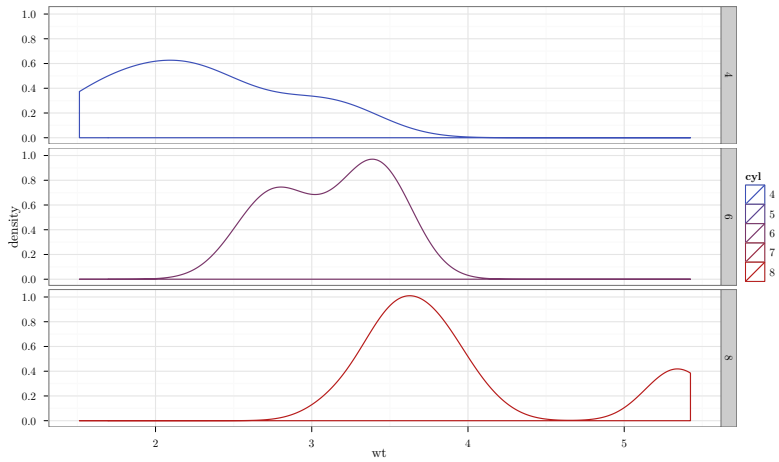
# ggplot2

```
qplot(factor(cyl),
 wt,
 geom=c("boxplot",
        "jitter"))
```



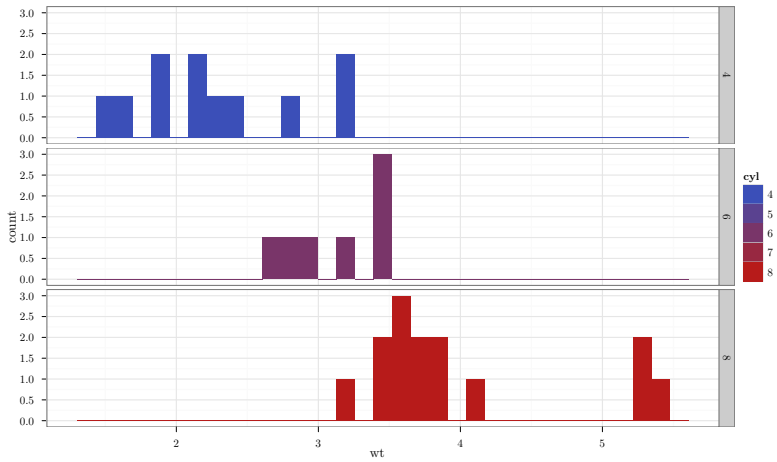
# ggplot2

```
qplot(wt,
 data=mtcars,
 geom="density",
 facets=cyl~.,
 colour=cyl)
```



# ggplot2

```
qplot(wt,
 data=mtcars,
 geom="histogram",
 facets=cyl~.,
 fill=cyl)
```



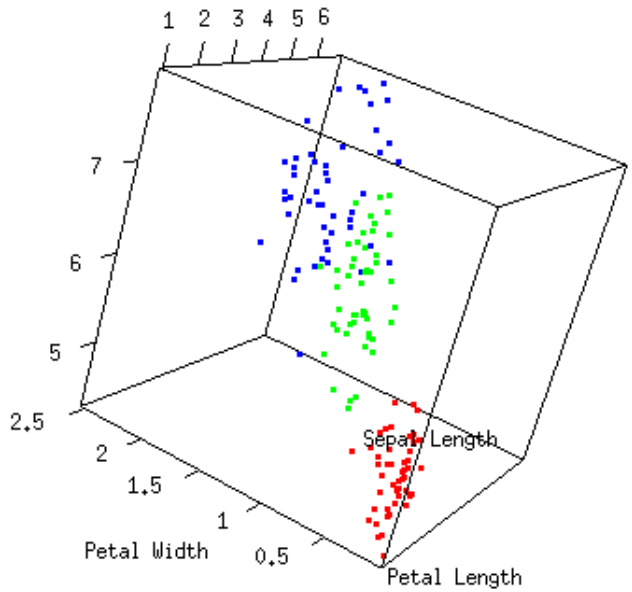
# RGL: Background

ICHEC
Irish Centre for High-End Computing

- Though R is generally used for static '2D' plots, interactive 3D plotting functionalities are provided by additional packages.
- RGL is an OpenGL-based vizualisation device for R.
- Real-time interactive navigation.
- Fast, convenient way to visualize 3D clouds of dots.

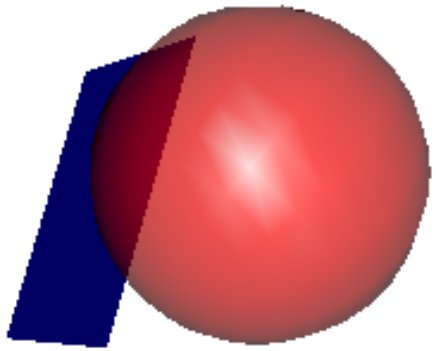
# RGL: dot cloud

```
library(rgl)
x <- iris$Petal.Length
y <- iris$Petal.Width
z <- iris$Sepal.Length
s <- iris$Species
plot3d(x, y, z,
  col=rainbow(3)[unclass(s)],
  xlab="Petal Length",
  ylab="Petal Width",
  zlab="Sepal Length")
# use mouse to navigate
```



# RGL: simple shapes

```
# points3d(x, y, z,  ...)
# lines3d(x, y, z,  ...)
# segments3d(x, y, z, ...)
# triangles3d(x, y, z, ...)
# quads3d(x, y, z, ...)
# spheres3d(x, y, z, radius,
quads3d(c(1,1,5,5),
        c(1,5,5,1),
        c(0,2,2,0),
        col="blue")
spheres3d(8, 3, 1, 2,
        col="red",
        alpha=0.7)
```



# RGL: surfaces

```
x <- 1:nrow(volcano)
y <- 1:ncol(volcano)
rgl.surface(x, y, volcano)
# use mouse to navigate
```