Feuille de Travaux Dirigés n° 0
Les graphiques avec le logiciel R

Les exemples de cette feuille de travaux dirigés sont tirés de l'aide du logiciel R.

Table des matières

1 Introduction et contenu de ce T.D. 3
1.1 Le graphique de Charles Minard . . . 3
1.2 Importance des représentations graphiques . . . 3
1.3 Les grandes familles de fonctions graphiques . . . 6

2 Graphe de Cleveland 7

3 Diagramme en barres 9

4 Histogramme 15

5 Boîte à moustaches 18
5.1 boxplot d'une formule . . . 18
5.2 boxplot d'une matrice . . . 20

6 Graphique circulaire 22

7 Tableaux de contingence 26
7.1 balloonplot . . . 26
7.2 assocplot . . . 27
7.3 mosaicplot . . . 28
7.4 splineplot . . . 32
7.5 cdplot (Conditional Density Plots) . . . 36

8 Représentation graphique de variables qualitatives 40

9 Représentation graphique d'une matrice 43

10 Simplepanel plots 46

11 Jitter 48

12 Courbe 48

13 Loess (régression non-paramétrique) 52

14 Représentation graphique de l'estimation de la fonction de densité 55

15 Radar plots 63

16 Steam and leaf 71

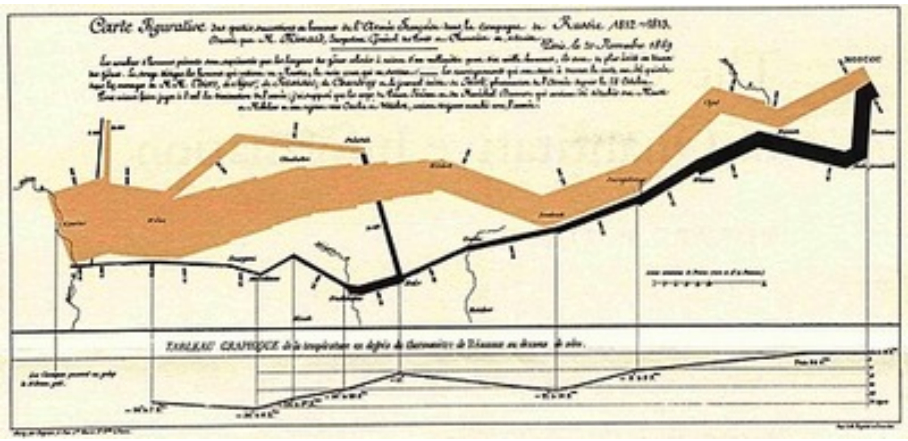
17 Conditioning plots 71

18 Persp 75

1 Introduction et contenu de ce T.D.

1.1 Le graphique de Charles Minard

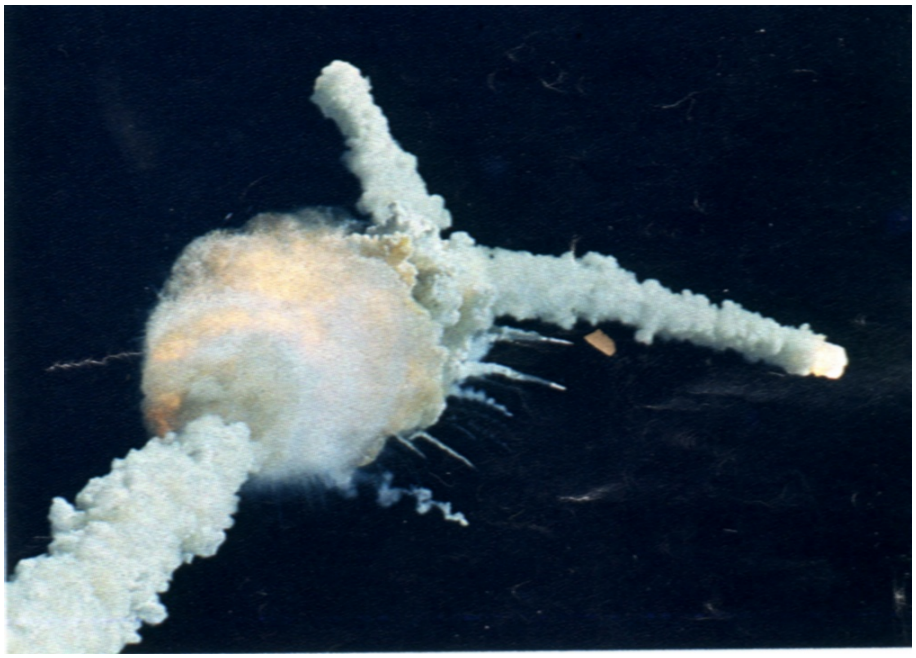
Citation : *" Un bon croquis vaut mieux qu'un long discours,"* d'après Napoléon.
Par une ironie de l'histoire, le graphique statistique considéré par beaucoup comme étant le meilleur jamais produit illustre la désastreuse campagne de Russie conduite par Napoléon en 1812.
Ce graphique est de l'ingénieur français Charles Minard (1781-1870). Il représente le nombre de survivants de l'armée par l'épaisseur des bandes sur la carte de la campagne, à l'aller et au retour. La température pendant la retraite est indiquée au bas de la figure.



Carte figurative des pertes successives en hommes de l'armée française dans la campagne de Russie 1812-1813.
Voir par exemple : http ://www.math.yorku.ca/SCS/Gallery/re-minard.html.

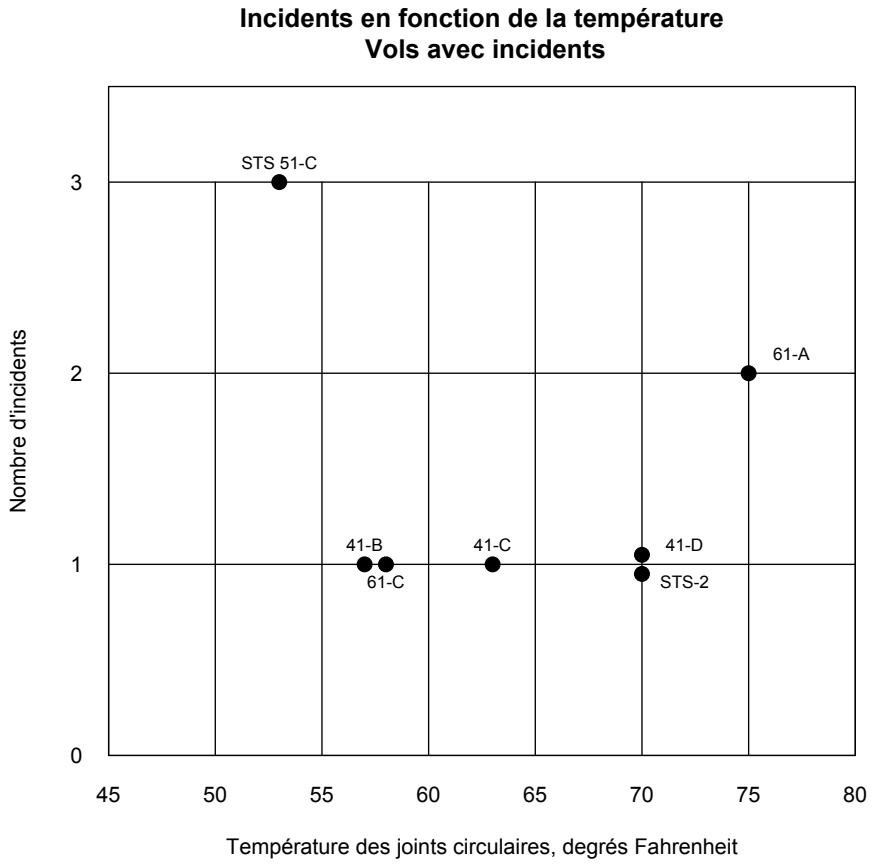
1.2 Importance des représentations graphiques

Exemple : explosion de la navette spatiale Challenger
Un mauvais graphique peut avoir des conséquences catastrophiques.



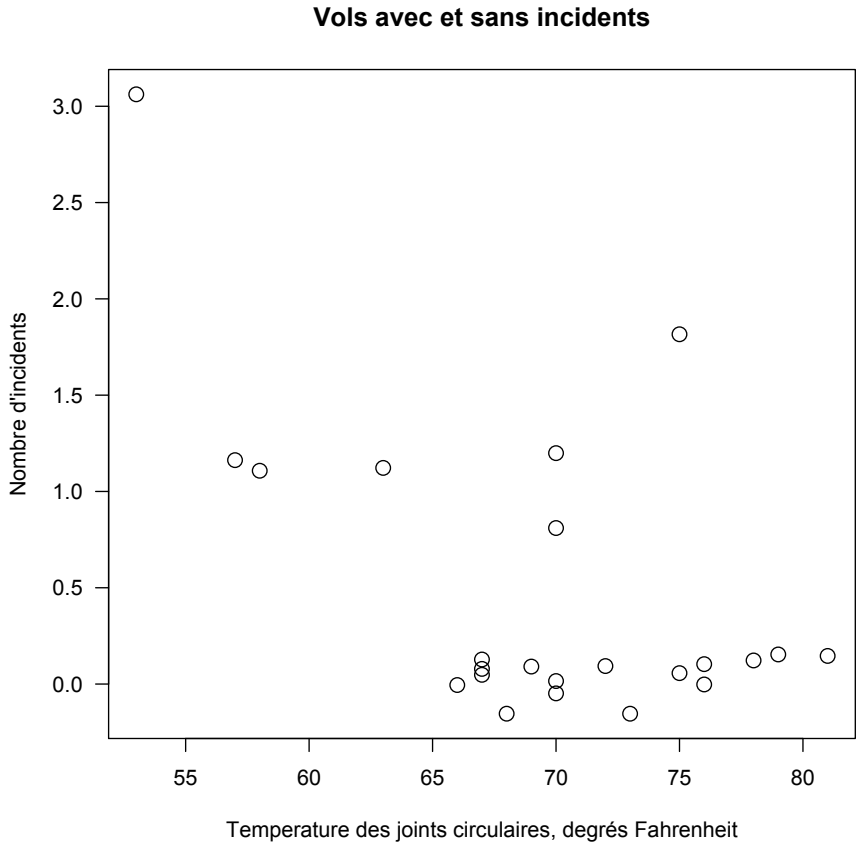
Un autre exemple : on peut (aussi) faire de mauvais graphiques sous R

```
> plot(orf[orf$Failure > 0 & orf$Temperature != 70, ], pch = 19, xlim = c(45,80),
+ ylim = c(0,3.5), cex = 1.5, las = 1,
+ xlab = "Température des joints circulaires, degrés Fahrenheit",
+ ylab = "Nombre d'incidents", xaxs = "i", yaxs = "i", xaxt = "n", yaxt = "n",
+ main = "Incidents en fonction de la température %\n
Vols avec incidents")
> points(c(70,70), c(0.95,1.05), pch = 19, cex = 1.5)
> axis(1, at = seq(45,80,by = 5), tick = FALSE)
> axis(2, at = 0:3, las = 1, tick = FALSE)
> abline(h = 0:3)
> for( i in seq(45,80,by = 5)) segments(i,0,i,3)
> text(53, 3, "STS 51-C", pos = 3, cex = 0.8)
> text(77, 2, "61-A", pos = 3, cex = 0.8)
> text(57, 1, "41-B", pos = 3, cex = 0.8)
> text(58, 1, "61-C", pos = 1, cex = 0.8)
> text(63, 1, "41-C", pos = 3, cex = 0.8)
> text(72, 1, "41-D", pos = 3, cex = 0.8)
> text(72, 1, "STS-2", pos = 1, cex = 0.8)
```



Les options par défaut des fonctions graphiques de R sont étudiées pour donner de bons résultats.

```
plot(orf$Temperature, jitter(orf$Failure), cex = 1.5,
las = 1, xlab = "Temperature des joints circulaires, degrés Fahrenheit",
+ ylab = "Nombre d'incidents",
main = "Vols avec et sans incidents")
```



1.3 Les grandes familles de fonctions graphiques

`R` est un très bon environnement pour produire de façon reproductible des graphiques statistiques de haute qualité. Nous pouvons classer les fonctions graphiques en plusieurs catégories :

- Les fonctions liées format de sortie des graphiques.
- Les fonction permettant d'interagir avec les graphiques.
- Les fonctions graphiques de bas niveau pour retoucher un graphique existant.
- Les fonctions graphiques de haut niveau.

Les fonctions de format de sortie des graphiques

- Il existe de nombreuses fonctions pour ouvrir un nouveau dispositif graphique (*e.g.* `pdf()`, `jpeg()`, `postscript()`, `x11()`, `png()`, `gnome()`, `quartz()`, `xfig()`, `bitmap()`, `pictex()`).
- Elles ne sont pas toutes disponibles pour tous les systèmes d'exploitation.
- Pour en savoir plus voir `?Devices`.
- Le dispositif utilisé par défaut est donné par : `getOption("device")`.

Exemple d'utilisation pour sauvegarder un graphique dans un fichier au format PDF :

```
>pdf("monfichier.pdf")
>plot(0)
>dev.off()
```

Les fonctions interactives
Ces fonctions permettent de retoucher "à la main" un graphique, tout en conservant le résultat pour sa reproductibilité ultérieure.

- `locator()` permet de récupérer les coordonnées des points lorsque nous cliquons sur le dispositif écran.
- `identify()` permet d'identifier les points sur le dispositif écran. Donne le rang des points dans le jeu de données.

Les fonctions graphiques de bas niveau
Ces fonctions permettent de retoucher un graphique déjà existant (*e.g.* `points()`, `abline()`, `arrows()`, `lines()`, `seqments()`, `polygon()`, `rect()`, `box()`, `axis()`, `title()`, `rug()`, `grid()`, `legend()`, `text()`, `mtext()`).
Exemple d'ajout d'une légende :

```
> t3var <- read.table("http://pbil.univ-lyon1.fr/R/donnees/t3var.txt", h = T)
> plot(t3var$tai,t3var$poi,pch=ifelse(t3var$sex=="h",1,19))
> legend("topleft", inset = 0.01, c("Homme","Femme"), pch = c(1,19))
```

Les fonctions graphiques de haut niveau
Ce sont celles que nous utilisons le plus souvent parce qu'elles donnent un graphique complet. Elles sont très nombreuses (*e.g.* `plot()`, `hist()`, `dotchart()`, `barplot()`, `hist()`, `boxplot()`, `pie()`, `balloonplot()`, `assocplot()`, `mosaicplot()`, `splineplot()`, `cdplot()`, `pairs()`, `curve()`, `stars()`, `coplot()`, `persp()`, `symbols()`, `contour()`, `image()`).
Nous allons envisager ci-après quelques fonctions graphiques de haut niveau très utilisés en analyse exploratoire des données.

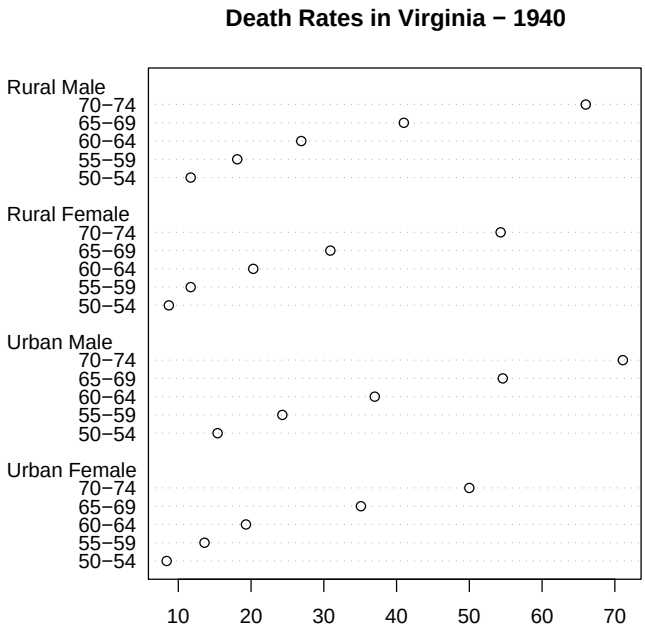
Nous allons maintenant nous intéresser à différents types de représentations graphiques adaptées à la nature des variables (quantitatives ou qualitatives) que nous souhaitons représenter.

```
> library(graphics)
```

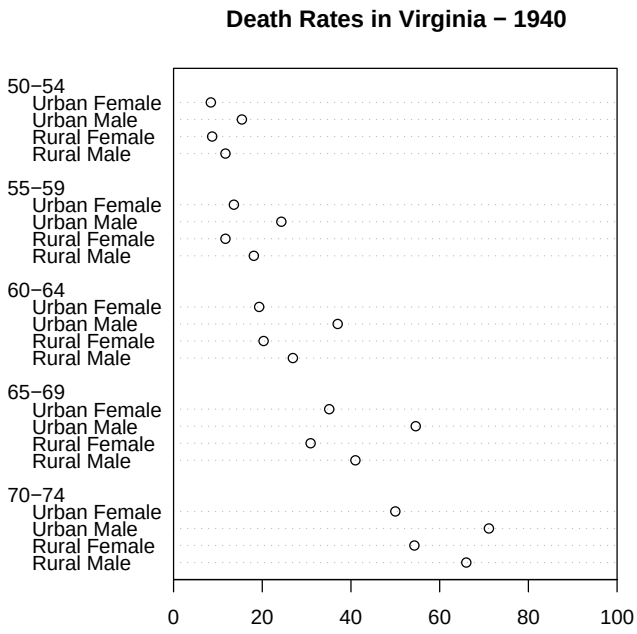
2 Graphe de Cleveland

Ce sont des graphes superposés ligne par ligne et colonne par colonne.

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

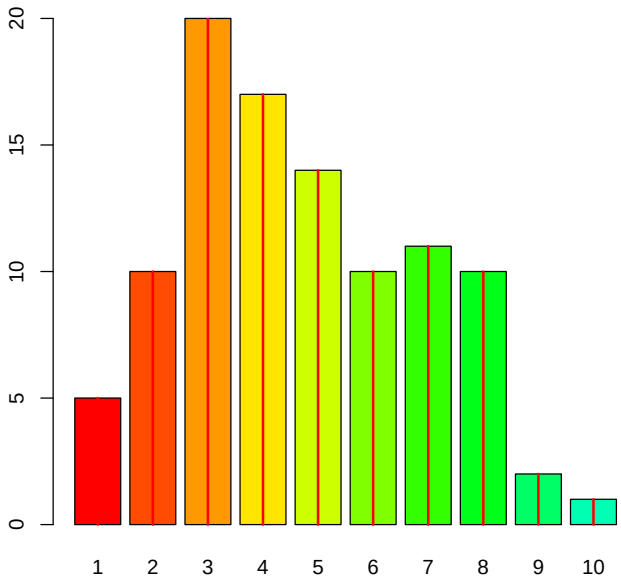



```
> op <- par(xaxs = "i")
> dotchart(t(VADeaths), main = "Death Rates in Virginia - 1940",
+     xlim = c(0, 100))
> par(op)
```

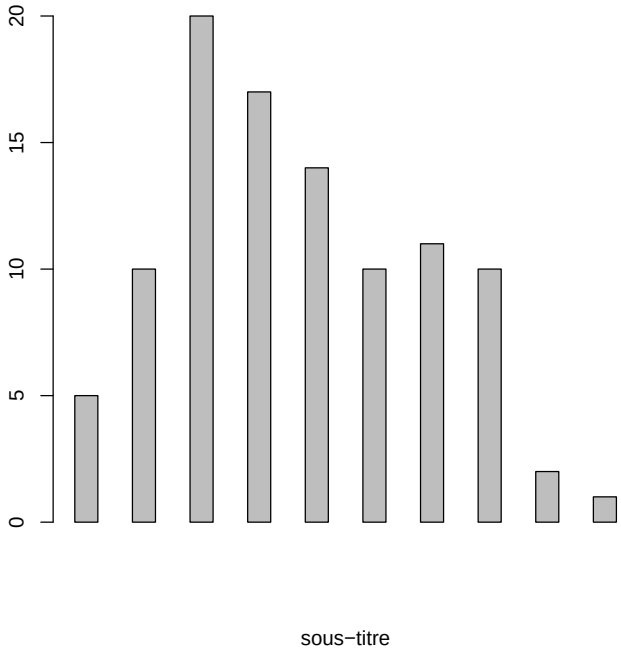


3 Diagramme en barres

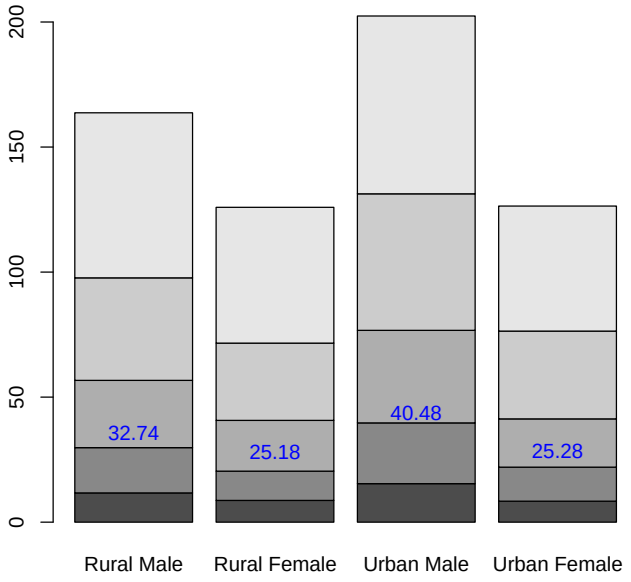
```
> require(grDevices)
> tN <- table(Ni <- stats::rpois(100, lambda = 5))

> r <- barplot(tN, col = rainbow(20))
> lines(r, tN, type = "h", col = "red", lwd = 2)
```



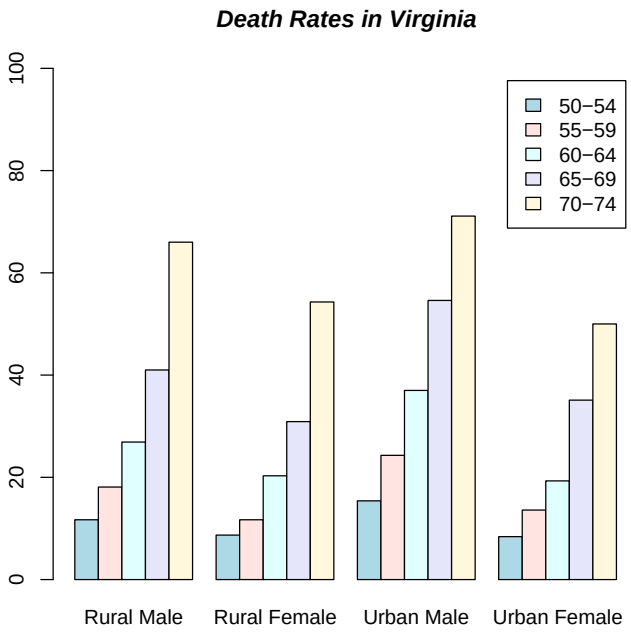

```
> barplot(tN, space = 1.5, axisnames = FALSE, sub = "sous-titre")
```



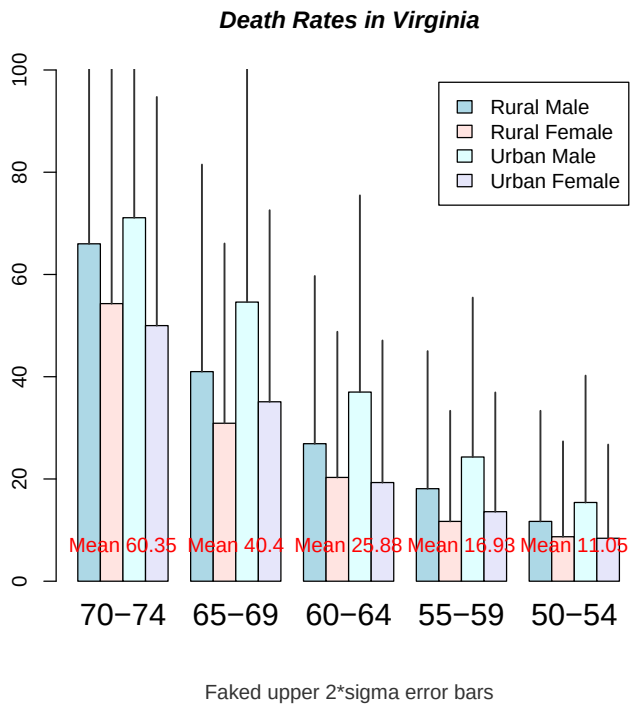

```
> barplot(VADeaths, plot = FALSE)

[1] 0.7 1.9 3.1 4.3

> barplot(VADeaths, plot = FALSE, beside = TRUE)

     [,1] [,2] [,3] [,4]
[1,]  1.5  7.5 13.5 19.5
[2,]  2.5  8.5 14.5 20.5
[3,]  3.5  9.5 15.5 21.5
[4,]  4.5 10.5 16.5 22.5
[5,]  5.5 11.5 17.5 23.5

> mp <- barplot(VADeaths)
> tot <- colMeans(VADeaths)
> text(mp, tot + 3, format(tot), xpd = TRUE, col = "blue")
```



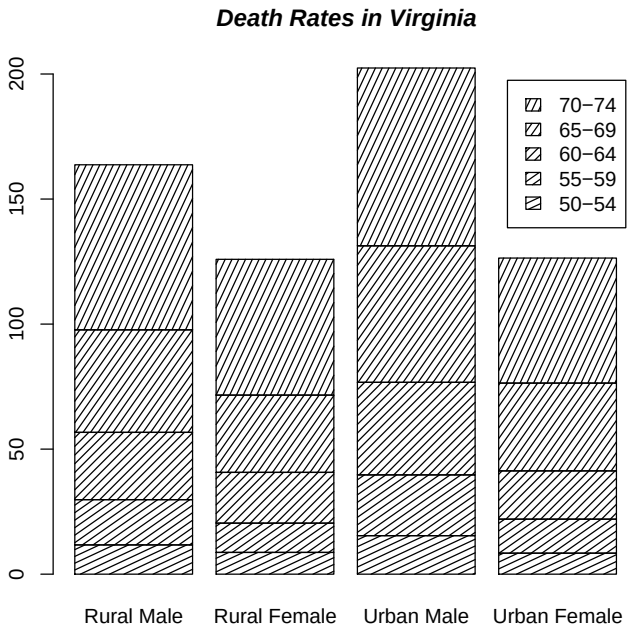

```
> barplot(VADeaths, beside = TRUE, col = c("lightblue",
+     "mistyrose", "lightcyan", "lavender", "cornsilk"),
+     legend = rownames(VADeaths), ylim = c(0, 100))
> title(main = "Death Rates in Virginia", font.main = 4)
```



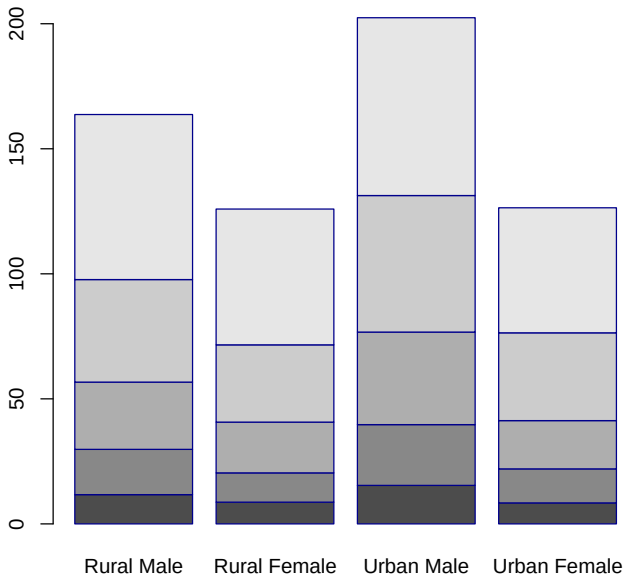
```
> hh <- t(VADeaths)[, 5:1]
> mybarcol <- "gray20"

> mp <- barplot(hh, beside = TRUE, col = c("lightblue",
+     "mistyrose", "lightcyan", "lavender"), legend = colnames(VADeaths),
+     ylim = c(0, 100), main = "Death Rates in Virginia",
+     font.main = 4, sub = "Faked upper 2*sigma error bars",
+     col.sub = mybarcol, cex.names = 1.5)
> segments(mp, hh, mp, hh + 2 * sqrt(1000 * hh/100), col = mybarcol,
+     lwd = 1.5)
> stopifnot(dim(mp) == dim(hh))
> mtext(side = 1, at = colMeans(mp), line = -2, text = paste("Mean",
+     formatC(colMeans(hh))), col = "red")
```




```
> barplot(VADeaths, angle = 15 + 10 * 1:5, density = 20,
+     col = "black", legend = rownames(VADeaths))
> title(main = list("Death Rates in Virginia", font = 4))
```

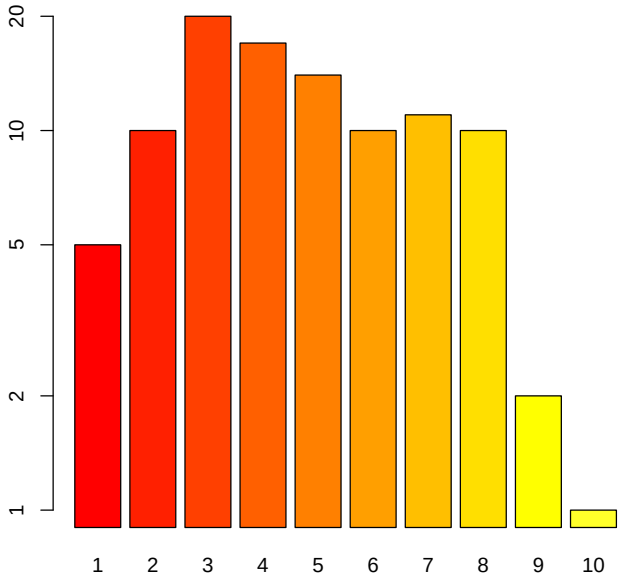



```
> barplot(VADeaths, border = "dark blue")
```

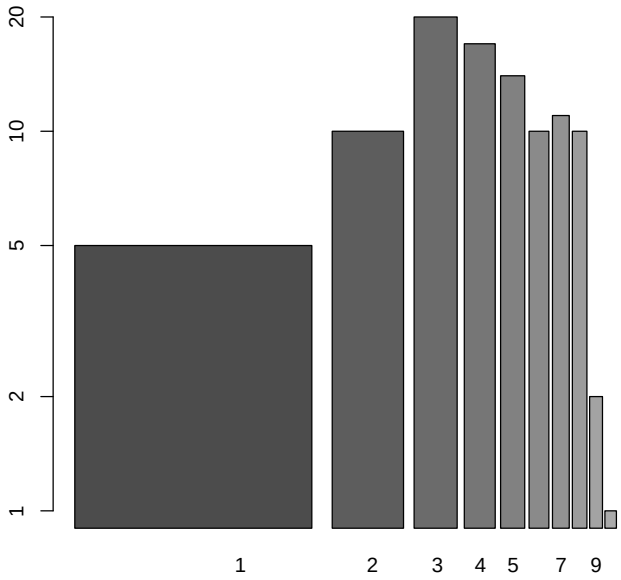


Échelles logarithmiques

```
> barplot(tN, col = heat.colors(12), log = "y")
```

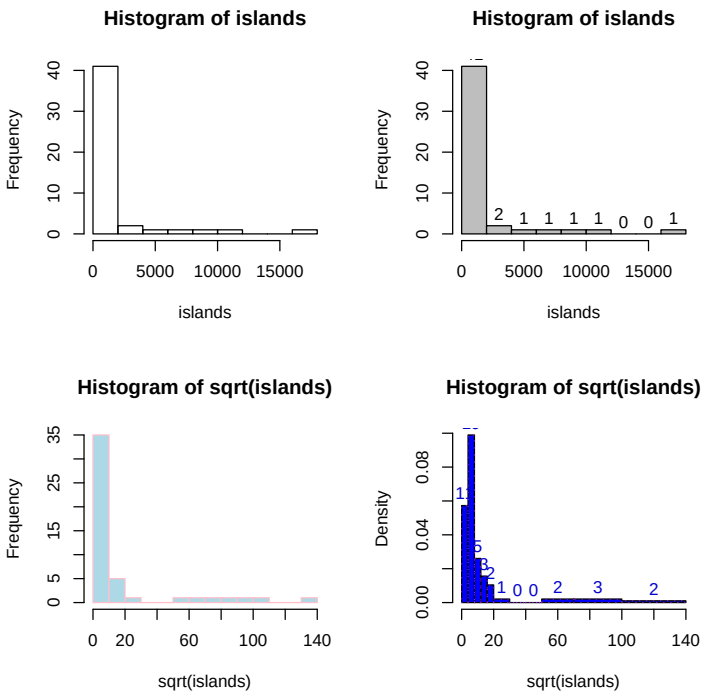



```
> barplot(tN, col = gray.colors(20), log = "xy")
```



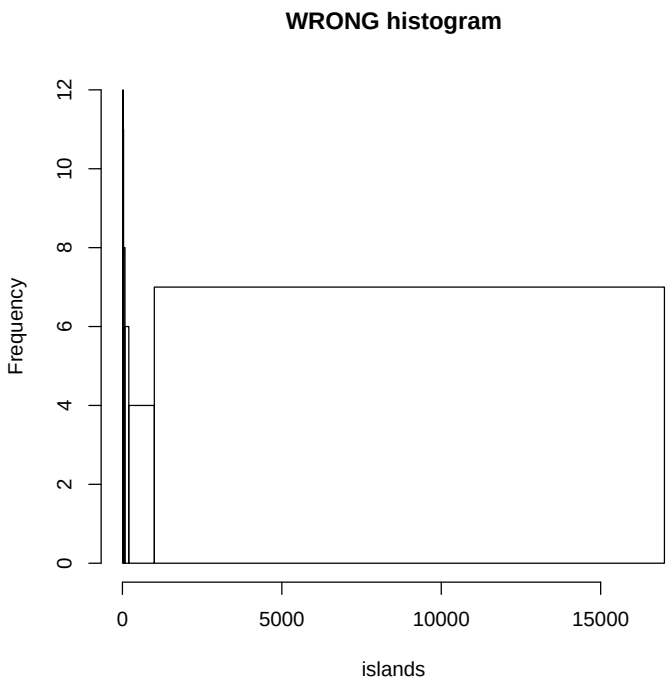
4 Histogramme

```
> op <- par(mfrow = c(2, 2))
> hist(islands)
> utils::str(hist(islands, col = "gray", labels = TRUE))
> hist(sqrt(islands), breaks = 12, col = "lightblue",
+     border = "pink")
> r <- hist(sqrt(islands), breaks = c(4 * 0:5, 10 * 3:5,
+     70, 100, 140), col = "blue1")
> text(r$mids, r$density, r$counts, adj = c(0.5, -0.5),
+     col = "blue3")
> sapply(r[2:3], sum)
> sum(r$density * diff(r$breaks))
> lines(r, lty = 3, border = "purple")
> par(op)
```




```
> require(utils)
> str(hist(islands, breaks = 12, plot = FALSE))

List of 7
 $ breaks     : num [1:10] 0 2000 4000 6000 8000 10000 12000 14000 16000 18000
 $ counts     : int [1:9] 41 2 1 1 1 1 0 0 1
 $ intensities: num [1:9] 4.27e-04 2.08e-05 1.04e-05 1.04e-05 1.04e-05 ...
 $ density    : num [1:9] 4.27e-04 2.08e-05 1.04e-05 1.04e-05 1.04e-05 ...
 $ mids       : num [1:9] 1000 3000 5000 7000 9000 11000 13000 15000 17000
```

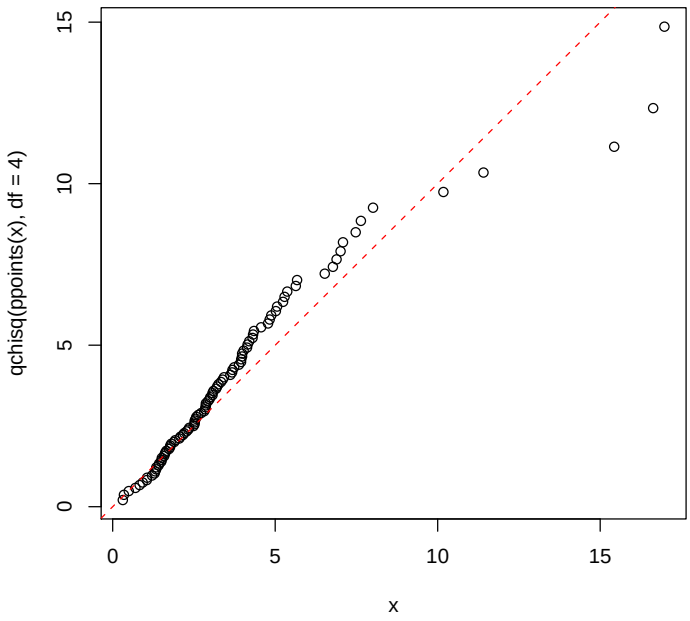
```
 $ xname      : chr "islands"
 $ equidist   : logi TRUE
 - attr(*, "class")= chr "histogram"

> str(hist(islands, breaks = c(12, 20, 36, 80, 200, 1000,
+     17000), plot = FALSE))

List of 7
 $ breaks     : num [1:7] 12 20 36 80 200 1000 17000
 $ counts     : int [1:6] 12 11 8 6 4 7
 $ intensities: num [1:6] 0.03125 0.014323 0.003788 0.001042 0.000104 ...
 $ density    : num [1:6] 0.03125 0.014323 0.003788 0.001042 0.000104 ...
 $ mids       : num [1:6] 16 28 58 140 600 9000
 $ xname      : chr "islands"
 $ equidist   : logi FALSE
 - attr(*, "class")= chr "histogram"

> hist(islands, breaks = c(12, 20, 36, 80, 200, 1000,
+     17000), freq = TRUE, main = "WRONG histogram")
```




```
> require(stats)
> set.seed(14)
> x <- rchisq(100, df = 4)
```

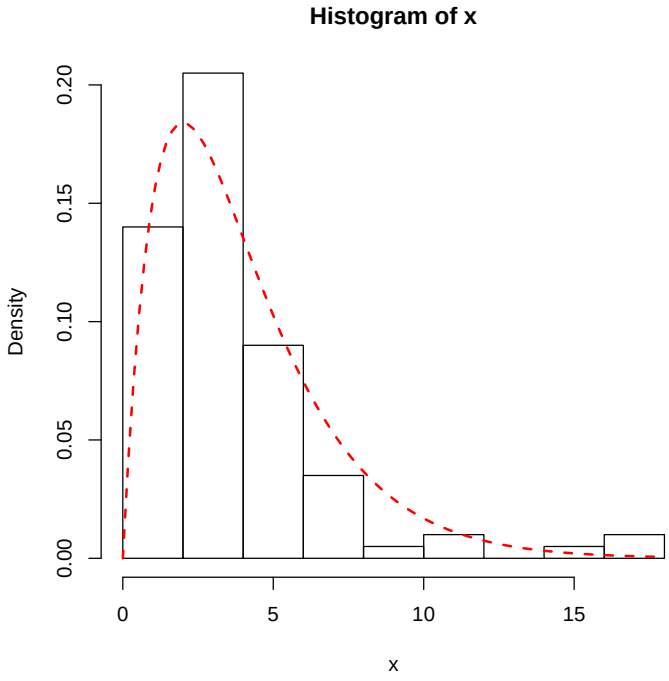
Comparing data with a model distribution should be done with qqplot() !

```
> qqplot(x, qchisq(ppoints(x), df = 4))
> abline(0, 1, col = 2, lty = 2)
```



if you really insist on using hist() ... :

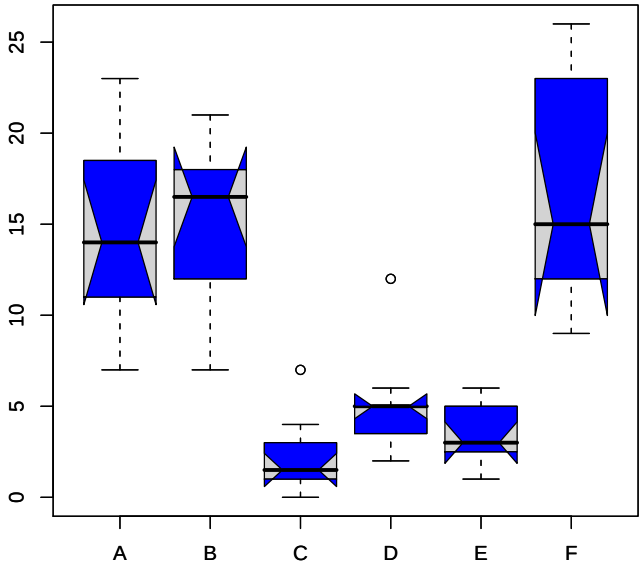
```
> hist(x, freq = FALSE, ylim = c(0, 0.2))
> curve(dchisq(x, df = 4), col = 2, lty = 2, lwd = 2,
+     add = TRUE)
```



5 Boîte à moustaches

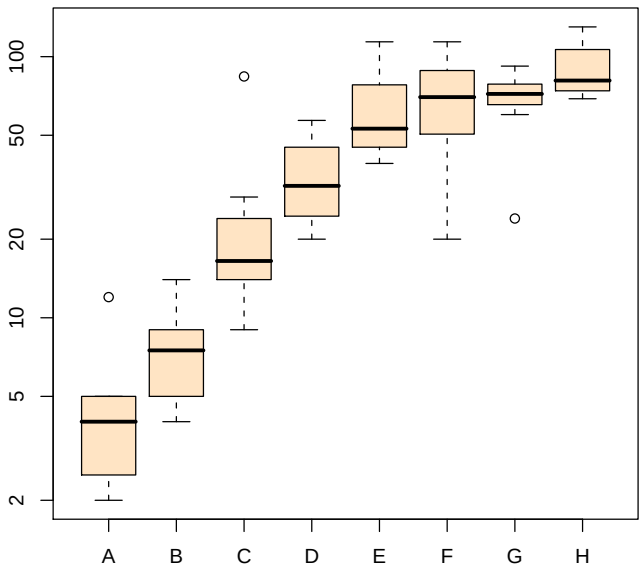
5.1 boxplot d'une formule

```
> boxplot(count ~ spray, data = InsectSprays, col = "lightgray")
> boxplot(count ~ spray, data = InsectSprays, notch = TRUE,
+     add = TRUE, col = "blue")
```

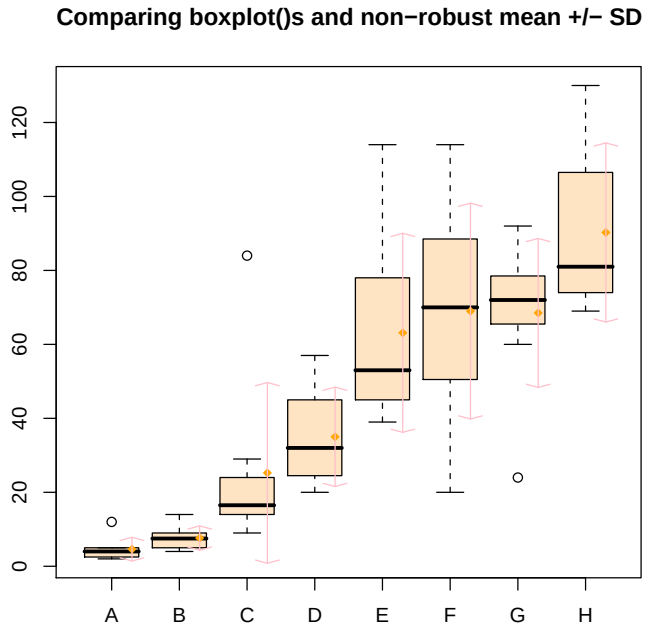


The last command add notches : If the notches of two plots do not overlap this is 'strong evidence' that the two medians differ.

```
> boxplot(decrease ~ treatment, data = OrchardSprays,
+     log = "y", col = "bisque")
```

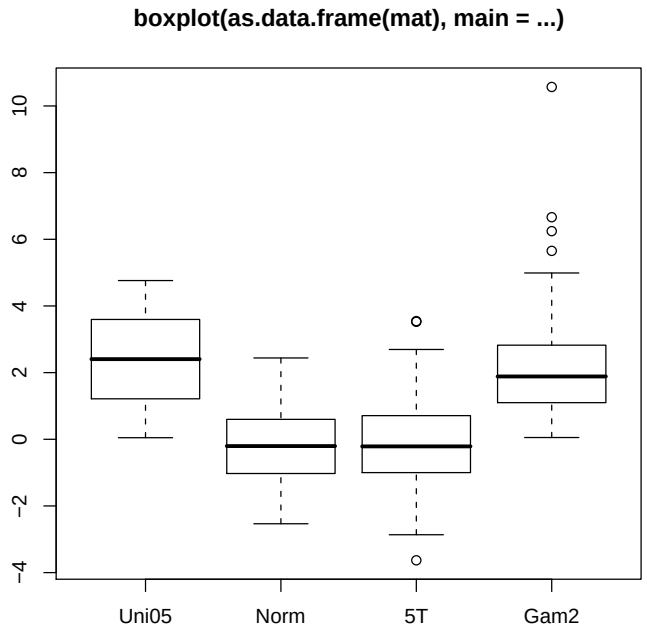


```
> rb <- boxplot(decrease ~ treatment, data = OrchardSprays,
+     col = "bisque")
> title("Comparing boxplot()s and non-robust mean +/- SD")
> mn.t <- tapply(OrchardSprays$decrease, OrchardSprays$treatment,
+     mean)
> sd.t <- tapply(OrchardSprays$decrease, OrchardSprays$treatment,
+     sd)
> xi <- 0.3 + seq(rb$n)
> points(xi, mn.t, col = "orange", pch = 18)
> arrows(xi, mn.t - sd.t, xi, mn.t + sd.t, code = 3, col = "pink",
+     angle = 75, length = 0.1)
```

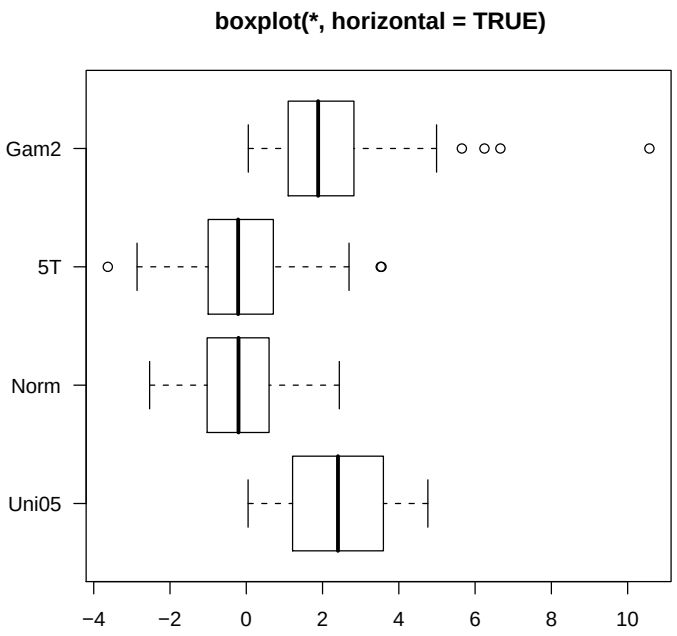


5.2 boxplot d'une matrice

```
> mat <- cbind(Uni05 = (1:100)/21, Norm = rnorm(100),
+     "5T" = rt(100, df = 5), Gam2 = rgamma(100, shape = 2))

> boxplot(as.data.frame(mat), main = "boxplot(as.data.frame(mat),
+     main = ...)")
```

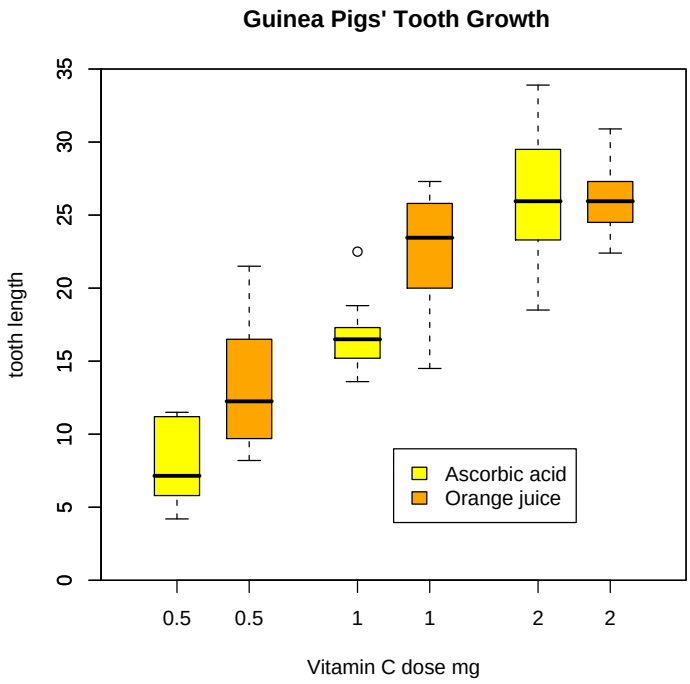



```
> par(las = 1)
> boxplot(as.data.frame(mat), main = "boxplot(*, horizontal = TRUE)",
+     horizontal = TRUE)
```



Using 'at = ' and adding boxplots – example idea by Roger Bivand :

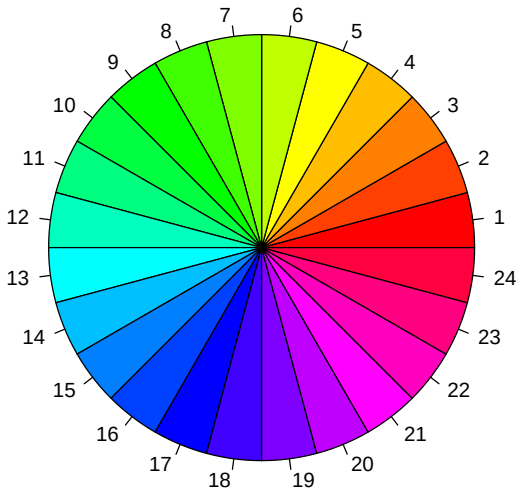
```
> boxplot(len ~ dose, data = ToothGrowth, boxwex = 0.25,
+     at = 1:3 - 0.2, subset = supp == "VC", col = "yellow",
+     main = "Guinea Pigs' Tooth Growth", xlab = "Vitamin C dose mg",
+     ylab = "tooth length", xlim = c(0.5, 3.5), ylim = c(0,
+         35), yaxs = "i")
> boxplot(len ~ dose, data = ToothGrowth, add = TRUE,
+     boxwex = 0.25, at = 1:3 + 0.2, subset = supp ==
+         "OJ", col = "orange")
> legend(2, 9, c("Ascorbic acid", "Orange juice"), fill = c("yellow",
+     "orange"))
```



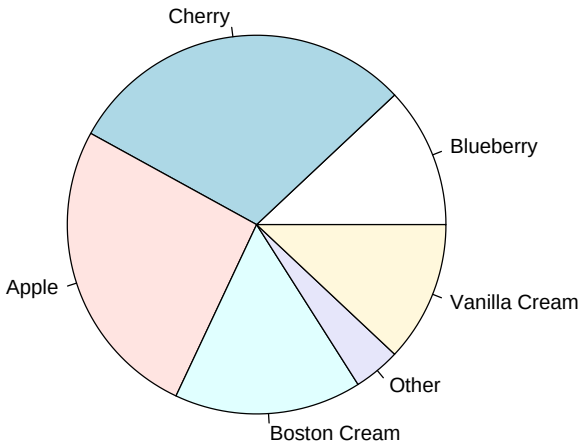
6 Graphique circulaire

```
> require(grDevices)
```

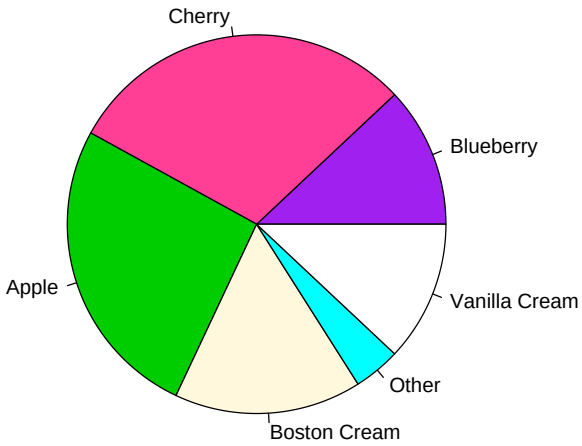
```
> pie(rep(1, 24), col = rainbow(24), radius = 0.9)
```



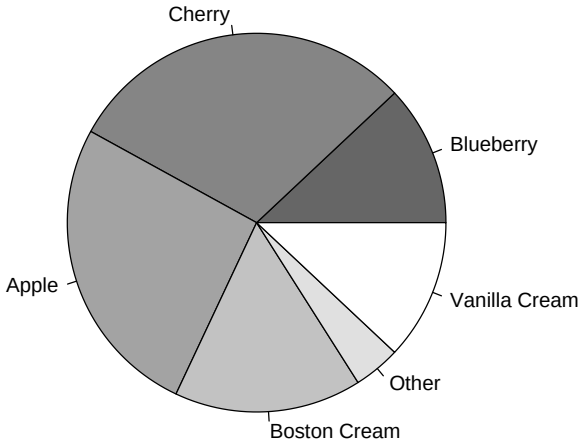

```
> pie.sales <- c(0.12, 0.3, 0.26, 0.16, 0.04, 0.12)
> names(pie.sales) <- c("Blueberry", "Cherry", "Apple",
+     "Boston Cream", "Other", "Vanilla Cream")

> pie(pie.sales)
```



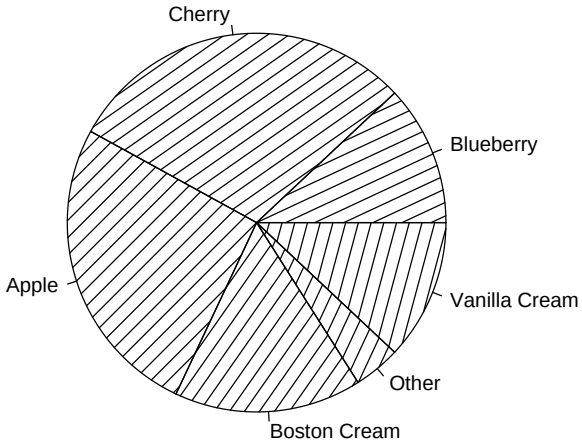
```
> pie(pie.sales, col = c("purple", "violetred1", "green3",
+     "cornsilk", "cyan", "white"))
```



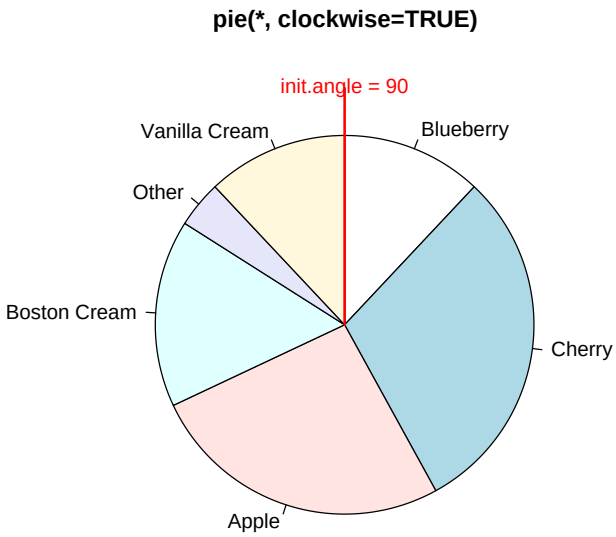

```
> pie(pie.sales, col = gray(seq(0.4, 1, length = 6)))
```




```
> pie(pie.sales, density = 10, angle = 15 + 10 * 1:6)
```

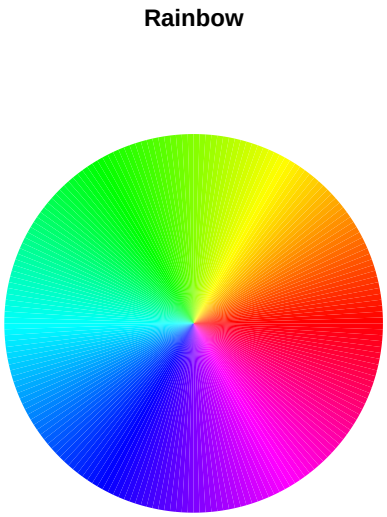



```
> pie(pie.sales, clockwise = TRUE, main = "pie(*, clockwise=TRUE)")
> segments(0, 0, 0, 1, col = "red", lwd = 2)
> text(0, 1, "init.angle = 90", col = "red")
```




```
> n <- 200
> pie(rep(1, n), labels = "", col = rainbow(n), border = NA,
```

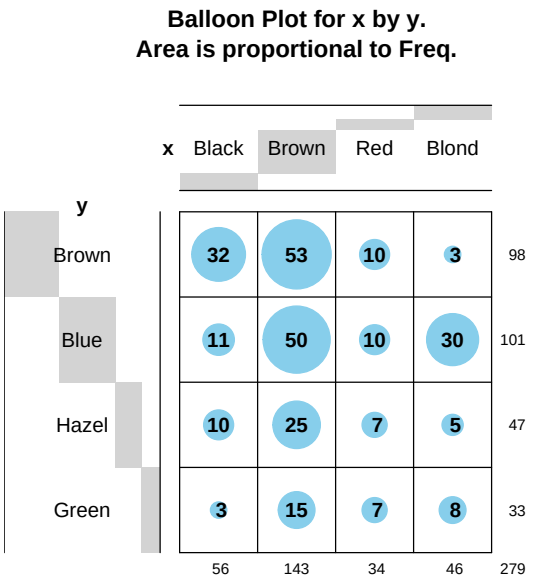
```
+     main = "Rainbow")
```



7 Tableaux de contingence

7.1 balloonplot

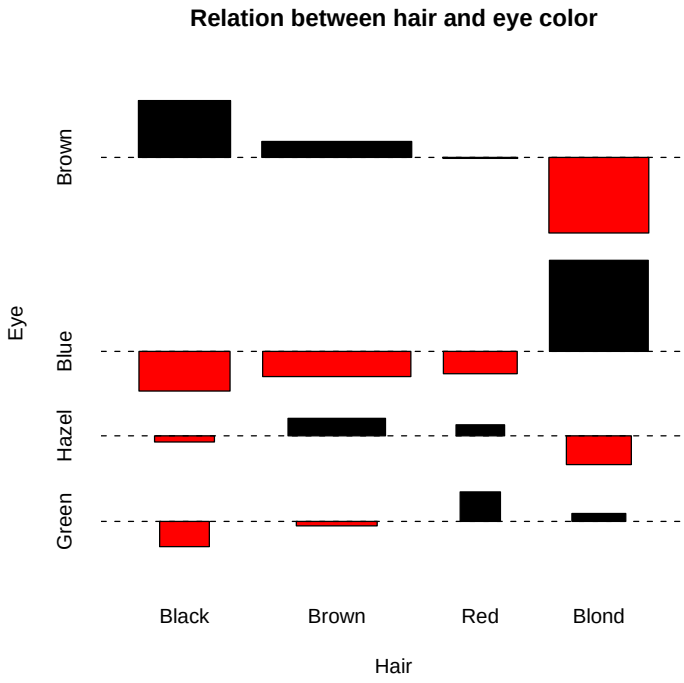
```
> library(gdata)
> library(gtools)
> library(gplots)

> balloonplot(as.table(HairEyeColor[, , Sex = "Male"]),
+     dotsize = 10)
> balloonplot(as.table(HairEyeColor[, , Sex = "Female"]),
+     dotsize = 10)
```



7.2 assocplot

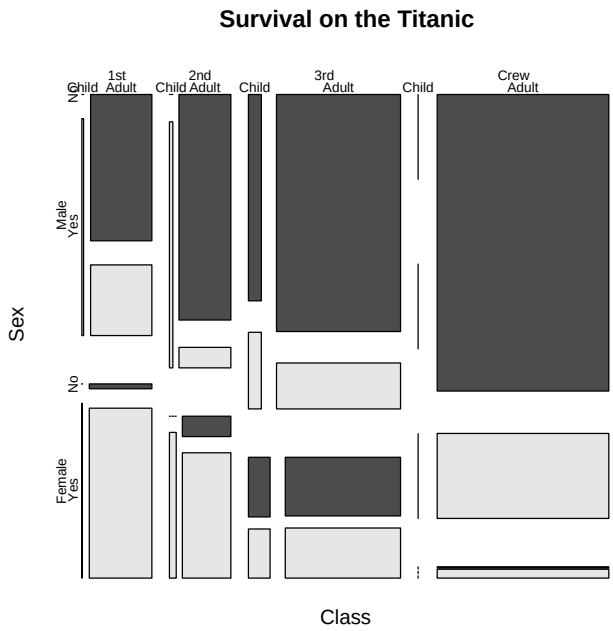
Aggregate over sex :

```
> x <- margin.table(HairEyeColor, c(1, 2))
> x

> assocplot(x, main = "Relation between hair and eye color")
```



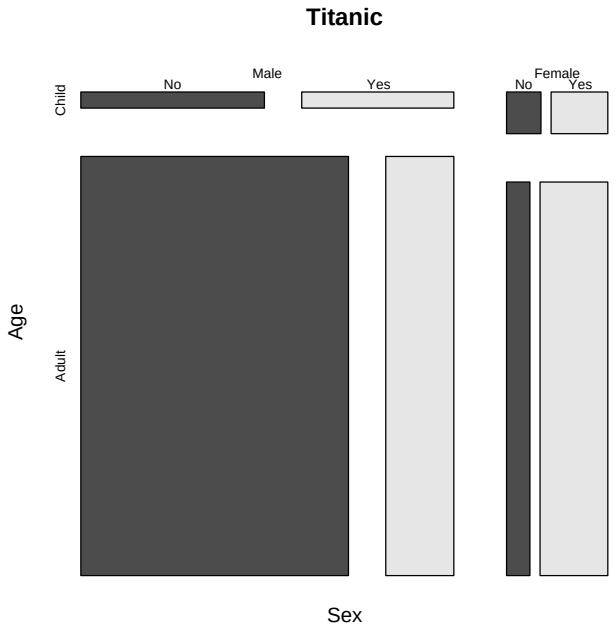
7.3 mosaicplot

```
> mosaicplot(Titanic, main = "Survival on the Titanic",
+     color = TRUE)
```

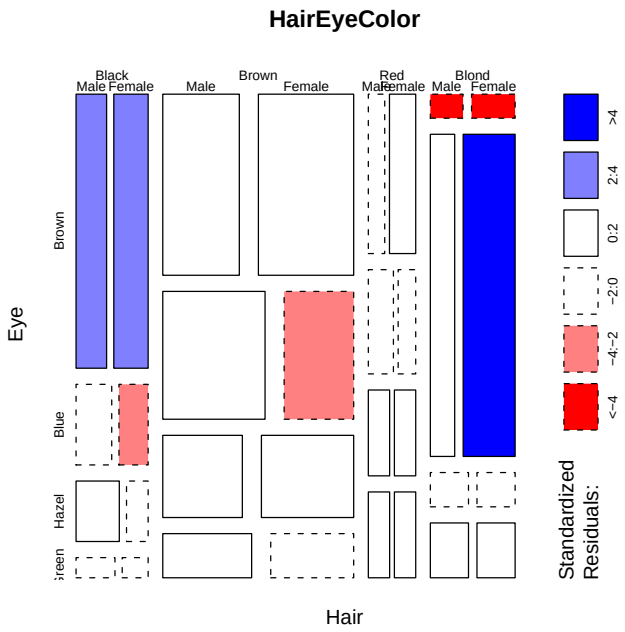


Formula interface for tabulated data :

```
> mosaicplot(~Sex + Age + Survived, data = Titanic, color = TRUE)
```

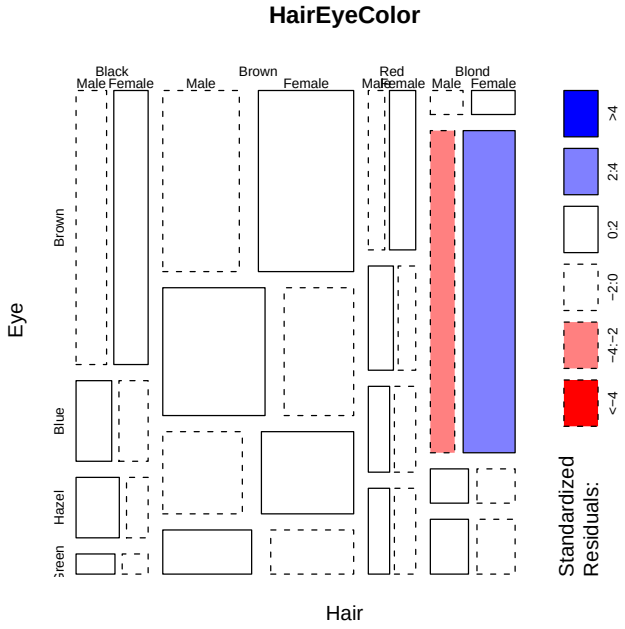



```
> mosaicplot(HairEyeColor, shade = TRUE)
```



Independence model of hair and eye color and sex. Indicates that ## there are more blue eyed blonde females than expected in the case ## of independence and too few brown eyed blonde females. ## The corresponding model is :

```
> fm <- loglin(HairEyeColor, list(1, 2, 3))
> pchisq(fm$pearson, fm$df, lower.tail = FALSE)

> mosaicplot(HairEyeColor, shade = TRUE, margin = list(1:2,
+     3))
```

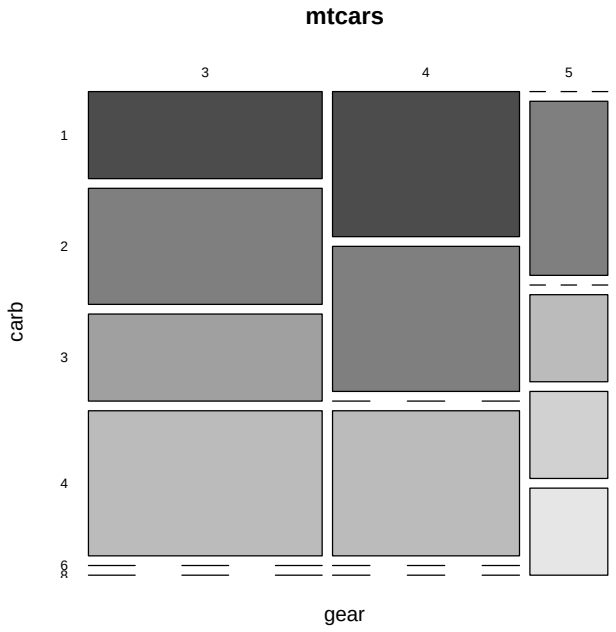


Model of joint independence of sex from hair and eye color. Males ## are underrepresented among people with brown hair and eyes, and are ## overrepresented among people with brown hair and blue eyes. ## The corresponding model is :

```
> fm <- loglin(HairEyeColor, list(1:2, 3))
> pchisq(fm$pearson, fm$df, lower.tail = FALSE)
```

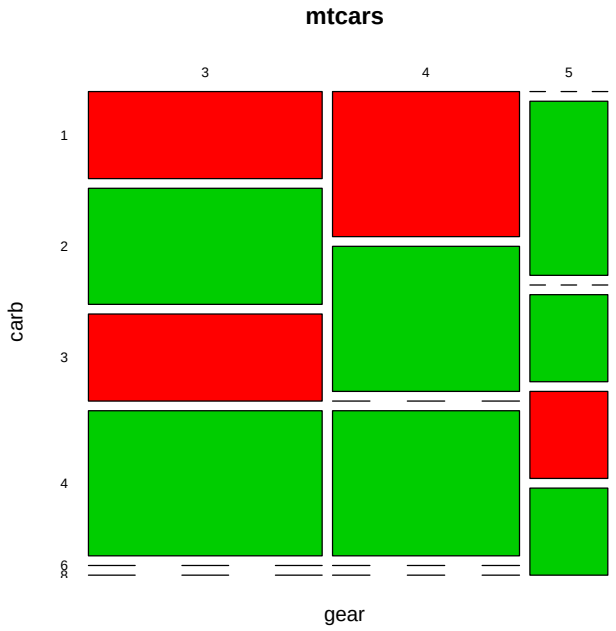
Formula interface for raw data : visualize cross-tabulation of numbers ## of gears and carburettors in Motor Trend car data.

```
> mosaicplot(~gear + carb, data = mtcars, color = TRUE,
+     las = 1)
```



color recycling

```
> mosaicplot(~gear + carb, data = mtcars, color = 2:3,
+     las = 1)
```



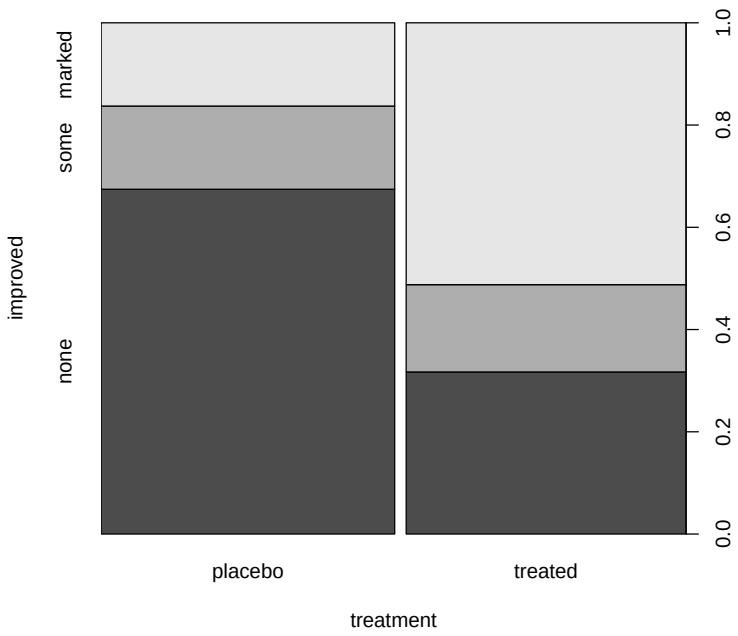
7.4 splineplot

treatment and improvement of patients with rheumatoid arthritis

```
> treatment <- factor(rep(c(1, 2), c(43, 41)), levels = c(1,
+     2), labels = c("placebo", "treated"))
> improved <- factor(rep(c(1, 2, 3, 1, 2, 3), c(29, 7,
+     7, 13, 7, 21)), levels = c(1, 2, 3), labels = c("none",
+     "some", "marked"))
```

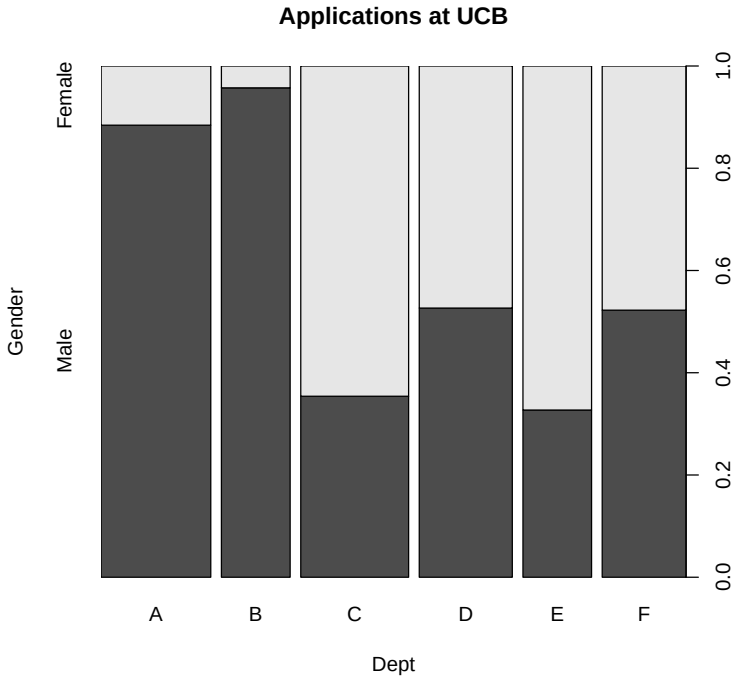
(dependence on a categorical variable)

```
> (spineplot(improved ~ treatment))
```

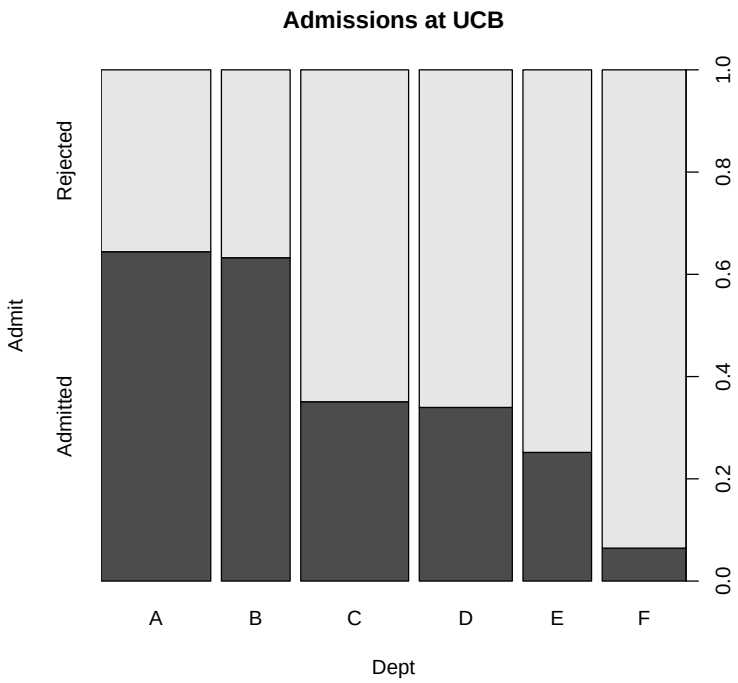


applications and admissions by department at UC Berkeley ## (two-way tables)

```
> (spineplot(margin.table(UCBAdmissions, c(3, 2)),
+     main = "Applications at UCB"))
```




```
> (spineplot(margin.table(UCBAdmissions, c(3, 1)),
+     main = "Admissions at UCB"))
```

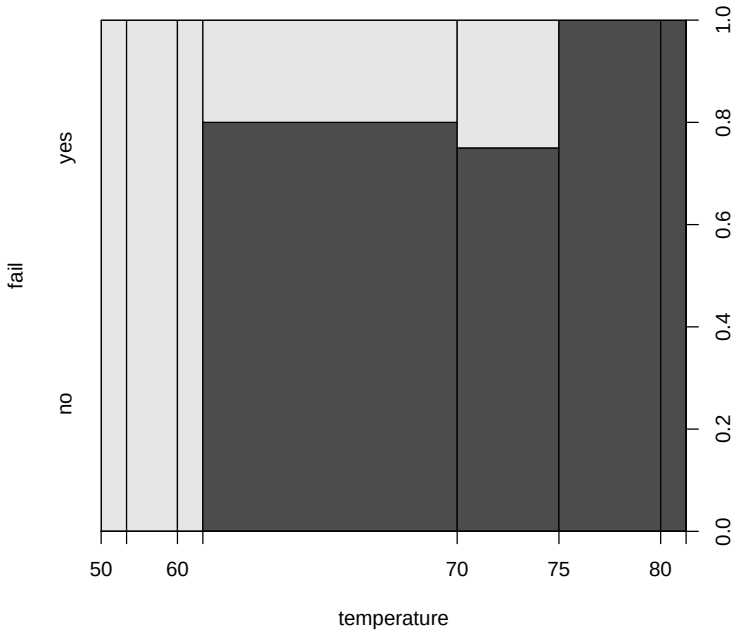


NASA space shuttle o-ring failures

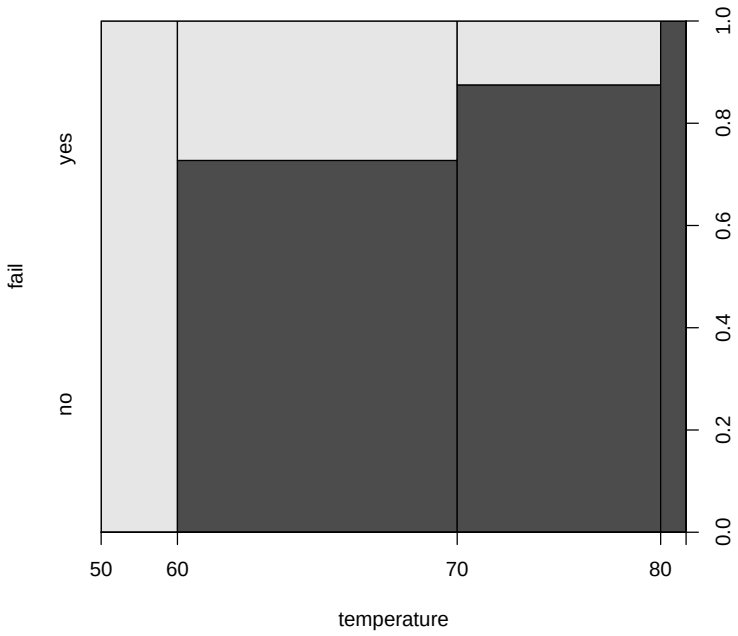
```
> fail <- factor(c(2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 2, 1,
+     2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1), levels = c(1,
+     2), labels = c("no", "yes"))
> temperature <- c(53, 57, 58, 63, 66, 67, 67, 67, 68,
+     69, 70, 70, 70, 70, 72, 73, 75, 75, 76, 76, 78,
+     79, 81)
```

(dependence on a numerical variable)

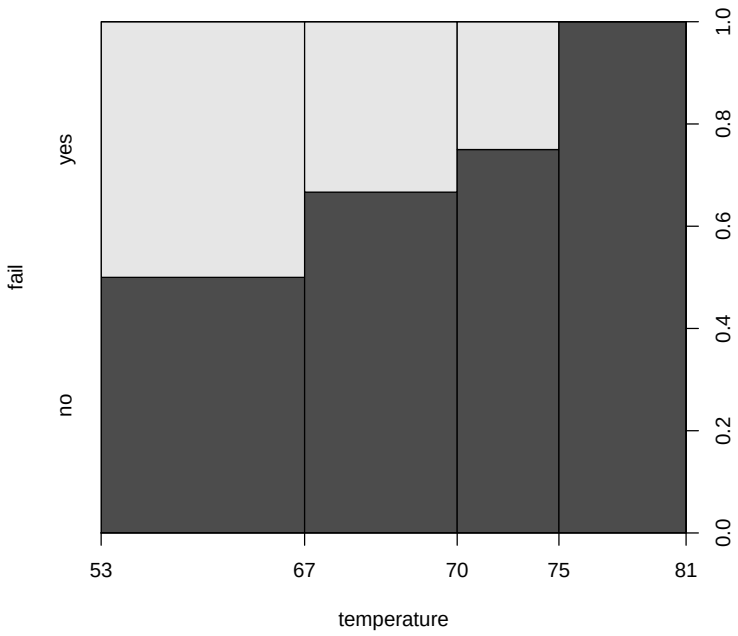
```
> (spineplot(fail ~ temperature))
```




```
> (spineplot(fail ~ temperature, breaks = 3))
```

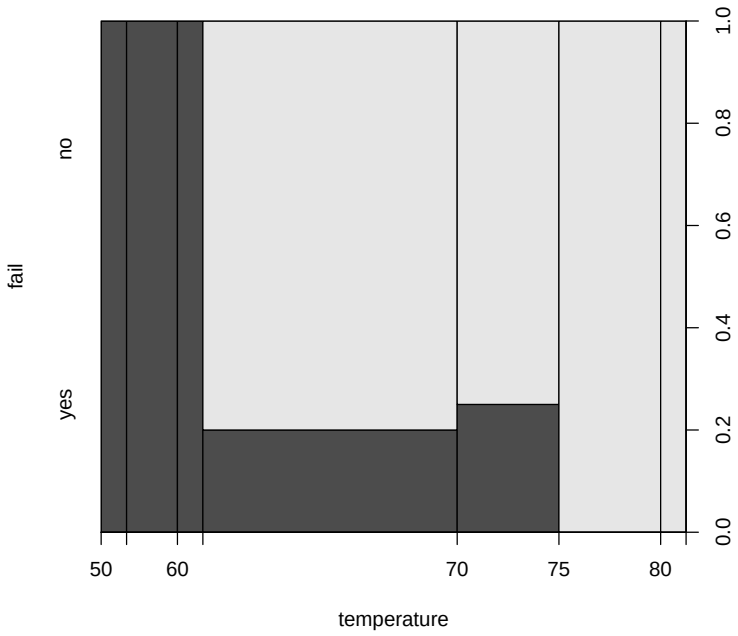



```
> (spineplot(fail ~ temperature, breaks = quantile(temperature)))
```



highlighting for failures

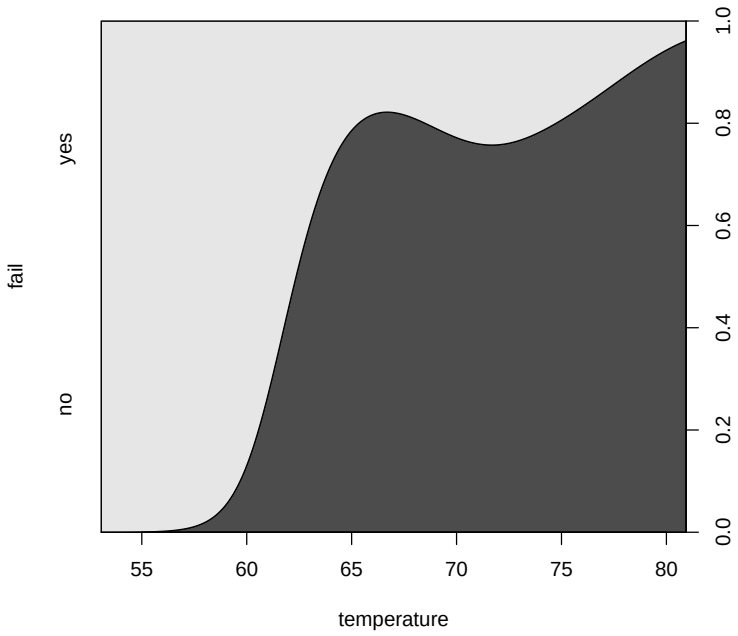
```
> spineplot(fail ~ temperature, ylevels = 2:1)
```



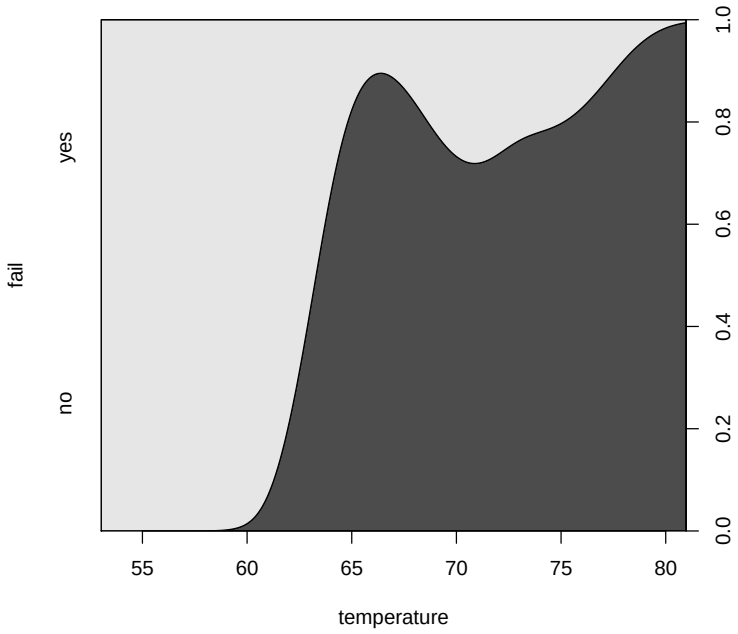
7.5 cdplot (Conditional Density Plots)

NASA space shuttle o-ring failures

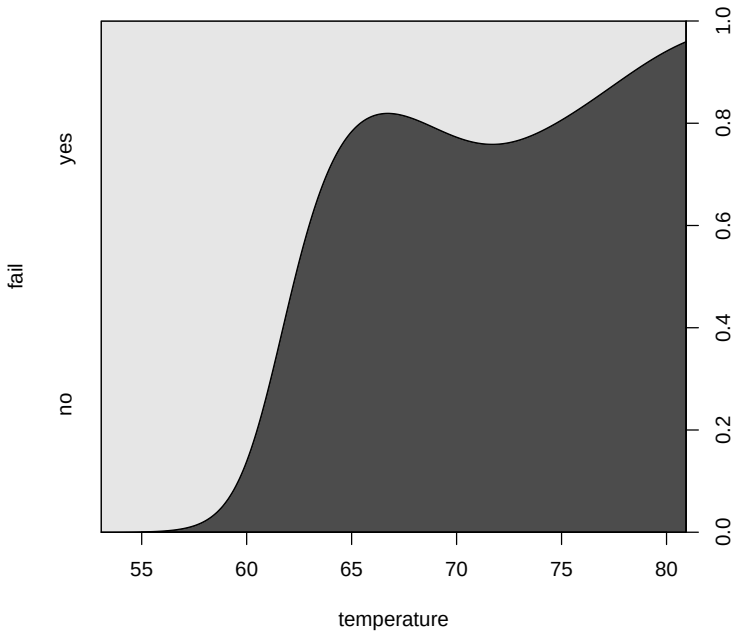
```
> fail <- factor(c(2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 2, 1,
+     2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1), levels = 1:2,
+     labels = c("no", "yes"))
> temperature <- c(53, 57, 58, 63, 66, 67, 67, 67, 68,
+     69, 70, 70, 70, 70, 72, 73, 75, 75, 76, 76, 78,
+     79, 81)

> cdplot(fail ~ temperature)
```




```
> cdplot(fail ~ temperature, bw = 2)
```

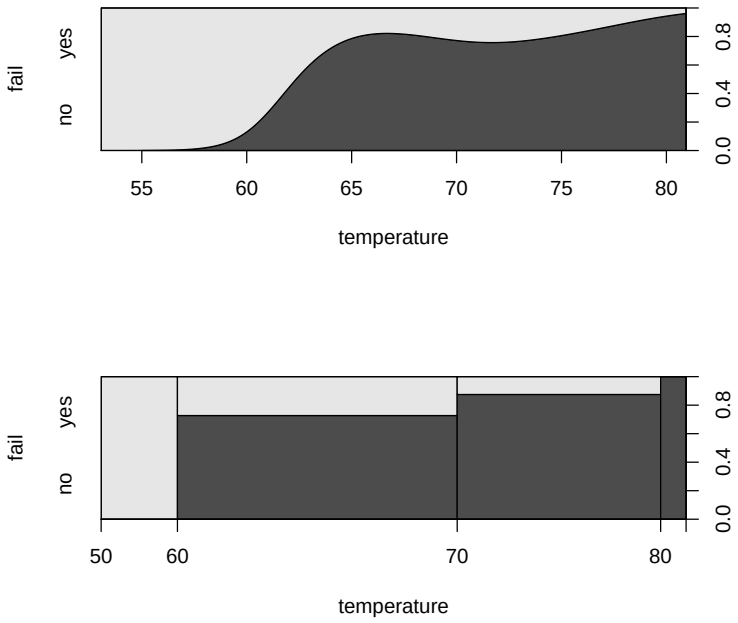



```
> cdplot(fail ~ temperature, bw = "SJ")
```



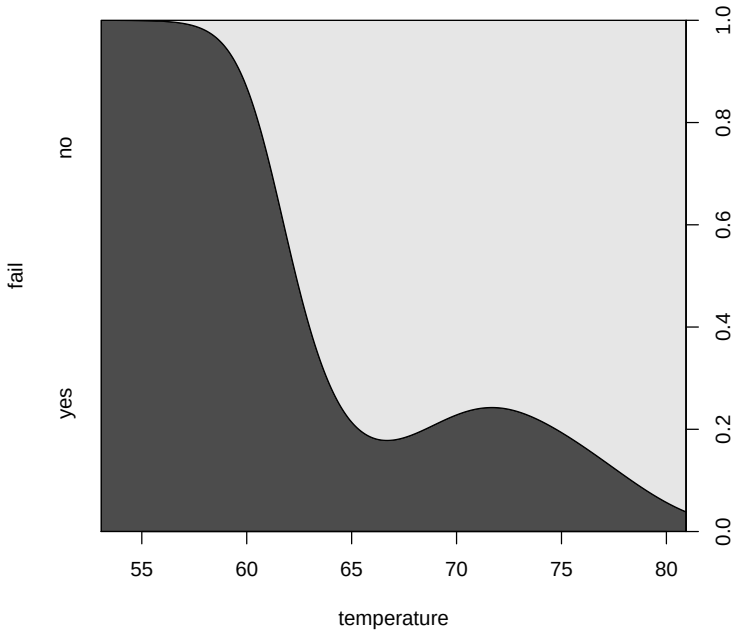
compare with spinogram on the same graph

```
> layout(1:2)
> cdplot(fail ~ temperature)
> (spineplot(fail ~ temperature, breaks = 3))
> layout(1)
```



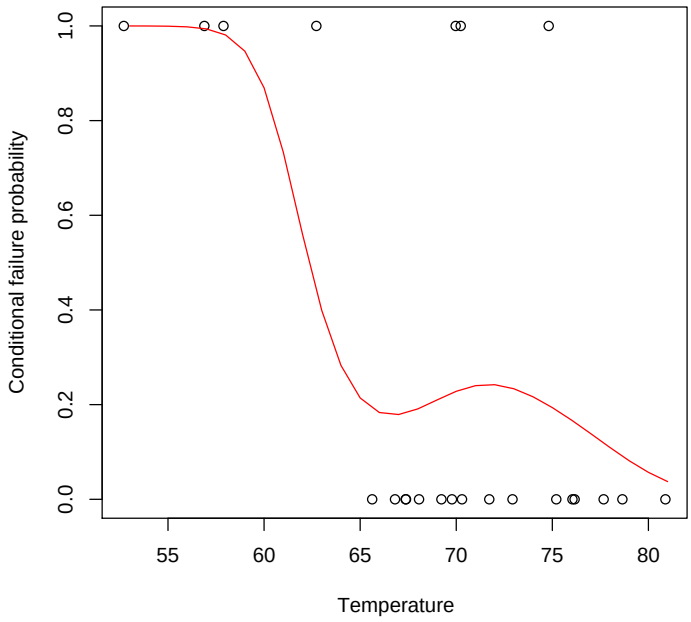
highlighting for failures

```
> cdplot(fail ~ temperature, ylevels = 2:1)
```



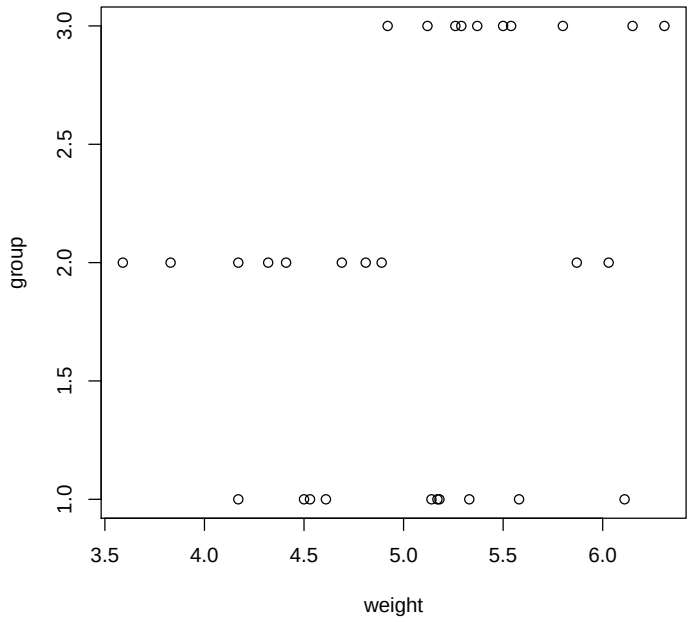
scatter plot with conditional density

```
> cdens <- cdplot(fail ~ temperature, plot = FALSE)

> plot(I(as.numeric(fail) - 1) ~ jitter(temperature, factor = 2),
+     xlab = "Temperature", ylab = "Conditional failure probability")
> lines(53:81, 1 - cdens[[1]](53:81), col = 2)
```

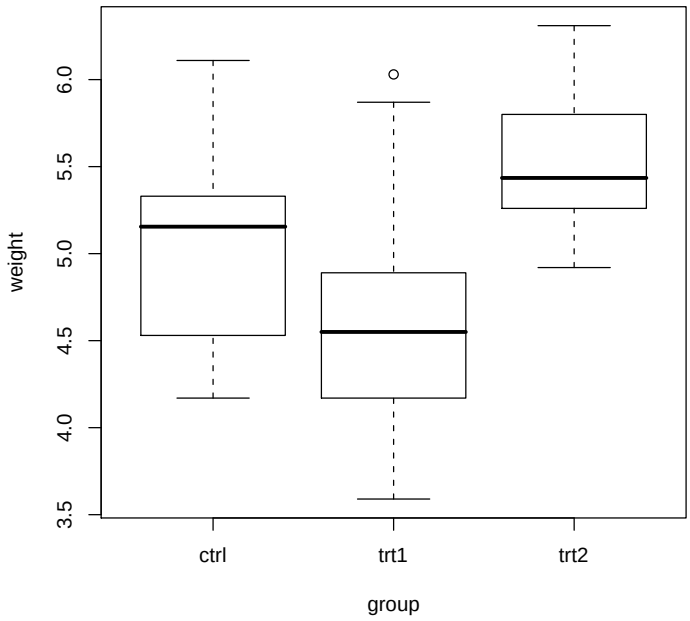


8 Représentation graphique de variables qualitatives

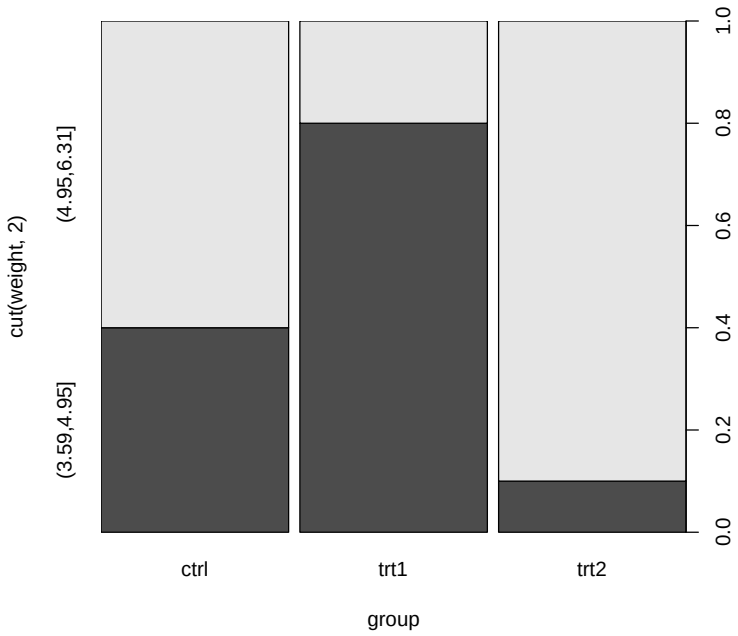
```
> require(grDevices)
> plot(PlantGrowth)
```




```
> plot(weight ~ group, data = PlantGrowth)
```

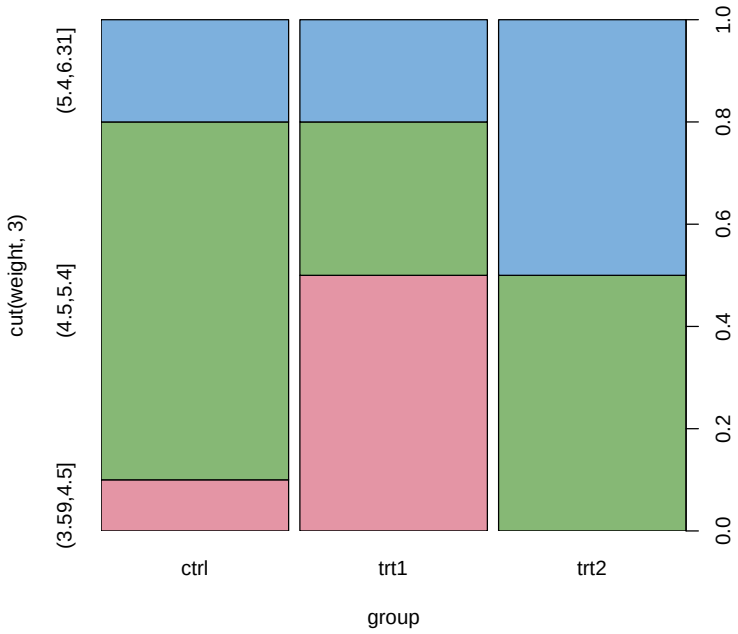



```
> plot(cut(weight, 2) ~ group, data = PlantGrowth)
```



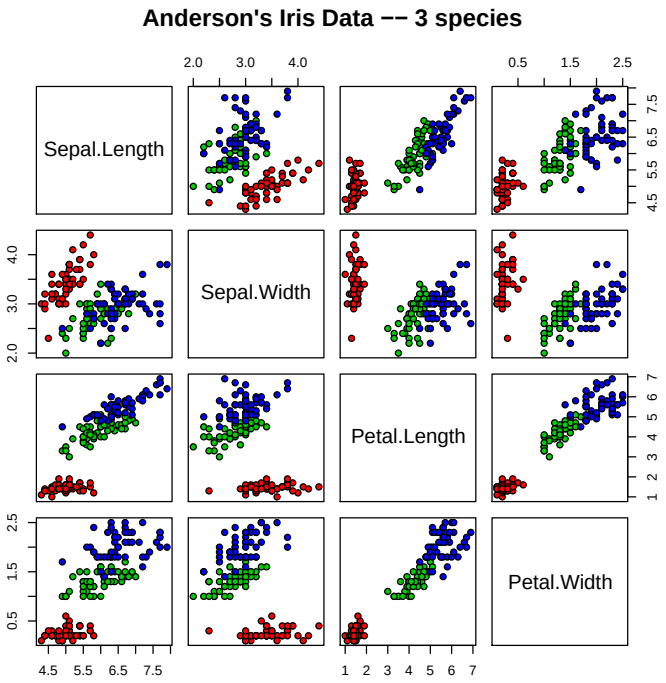
passing "..." to spineplot() eventually :

```
> plot(cut(weight, 3) ~ group, data = PlantGrowth, col = hcl(c(0,
+     120, 240), 50, 70))
```



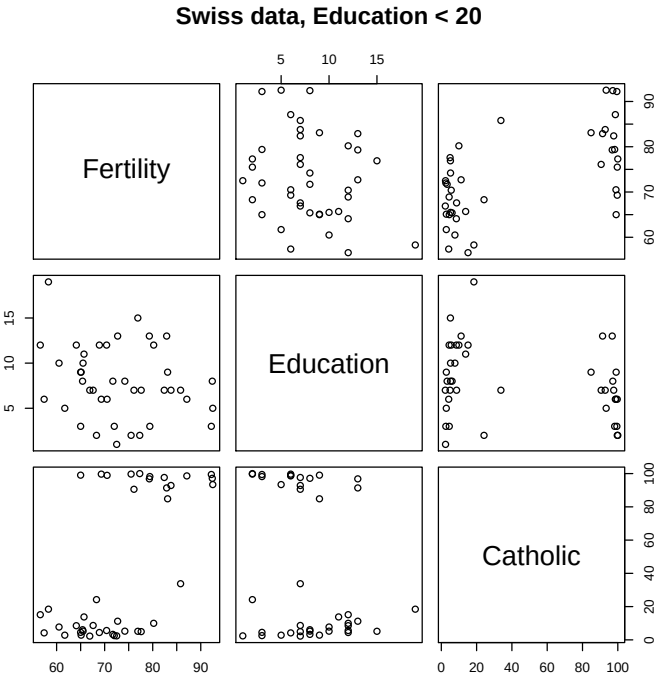
9 Représentation graphique d'une matrice

```
> pairs(iris[1:4], main = "Anderson's Iris Data -- 3 species",
+     pch = 21, bg = c("red", "green3", "blue")[unclass(iris$Species)])
```

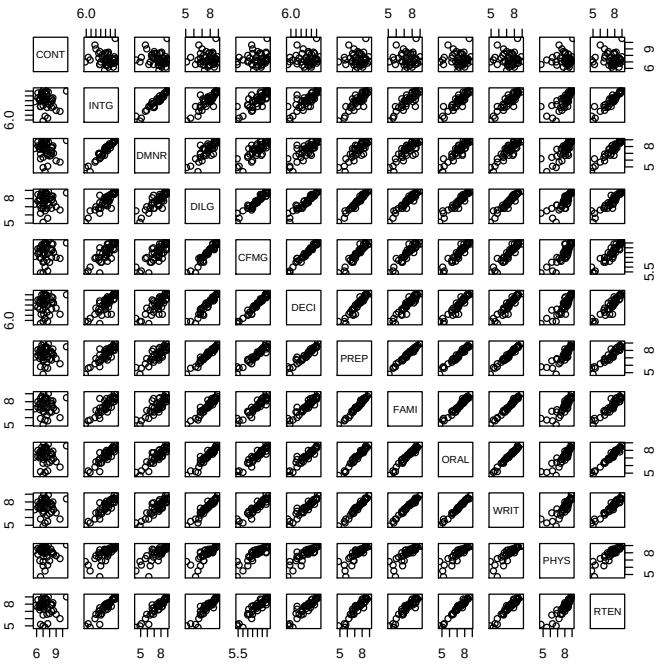


formula method

```
> pairs(~Fertility + Education + Catholic, data = swiss,
+     subset = Education < 20, main = "Swiss data, Education < 20")
```



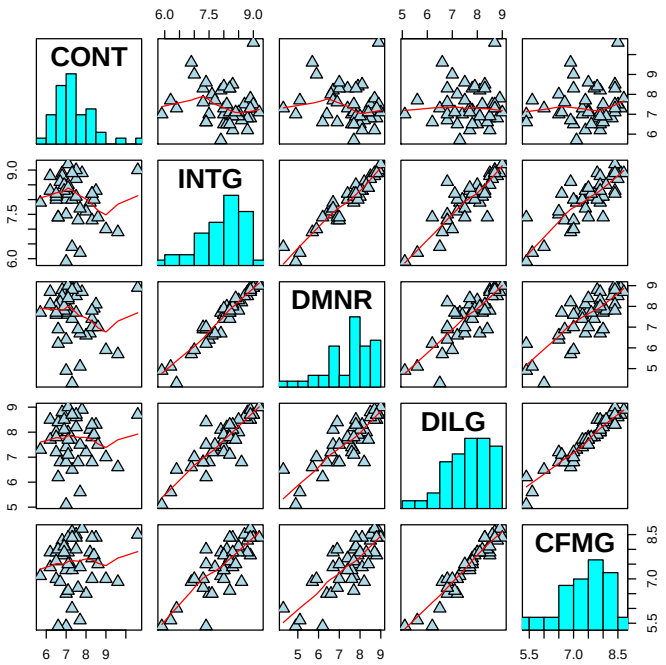

```
> pairs(USJudgeRatings)
```



put histograms on the diagonal

```
> panel.hist <- function(x, ...) {
+     usr <- par("usr")
```

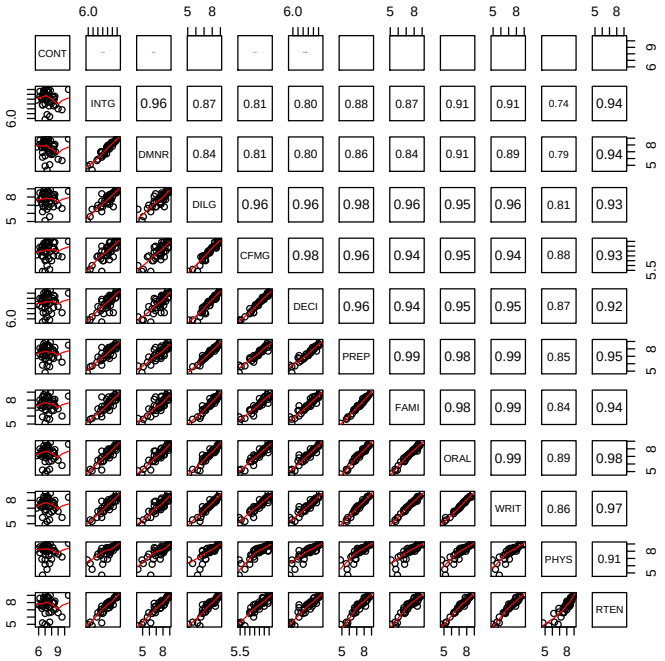
```
+     on.exit(par(usr))
+     par(usr = c(usr[1:2], 0, 1.5))
+     h <- hist(x, plot = FALSE)
+     breaks <- h$breaks
+     nB <- length(breaks)
+     y <- h$counts
+     y <- y/max(y)
+     rect(breaks[-nB], 0, breaks[-1], y, col = "cyan",
+         ...)
+ }

> pairs(USJudgeRatings[1:5], panel = panel.smooth, cex = 1.5,
+     pch = 24, bg = "light blue", diag.panel = panel.hist,
+     cex.labels = 2, font.labels = 2)
```



put (absolute) correlations on the upper panels, with size proportional to the correlations.

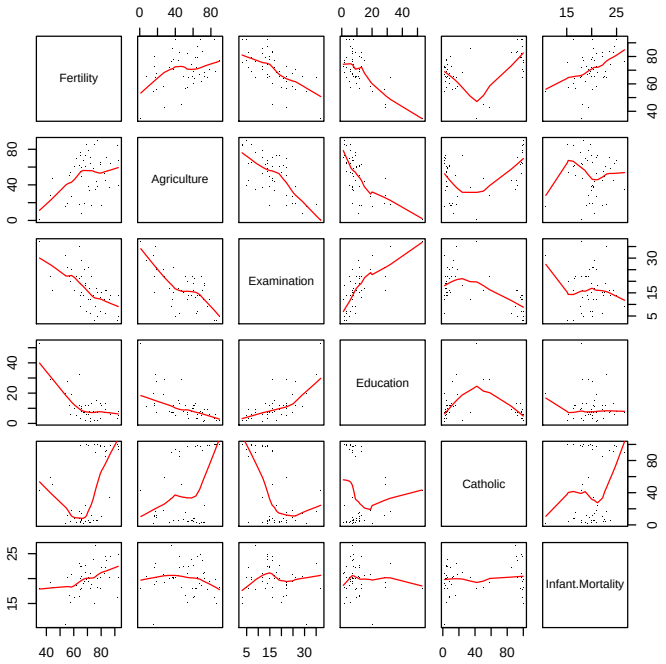
```
> panel.cor <- function(x, y, digits = 2, prefix = "",
+     cex.cor) {
+     usr <- par("usr")
+     on.exit(par(usr))
+     par(usr = c(0, 1, 0, 1))
+     r <- abs(cor(x, y))
+     txt <- format(c(r, 0.123456789), digits = digits)[1]
+     txt <- paste(prefix, txt, sep = "")
```

```
+     if (missing(cex.cor))
+         cex.cor <- 0.8/strwidth(txt)
+     text(0.5, 0.5, txt, cex = cex.cor * r)
+ }

> pairs(USJudgeRatings, lower.panel = panel.smooth, upper.panel =
+     panel.cor)
```

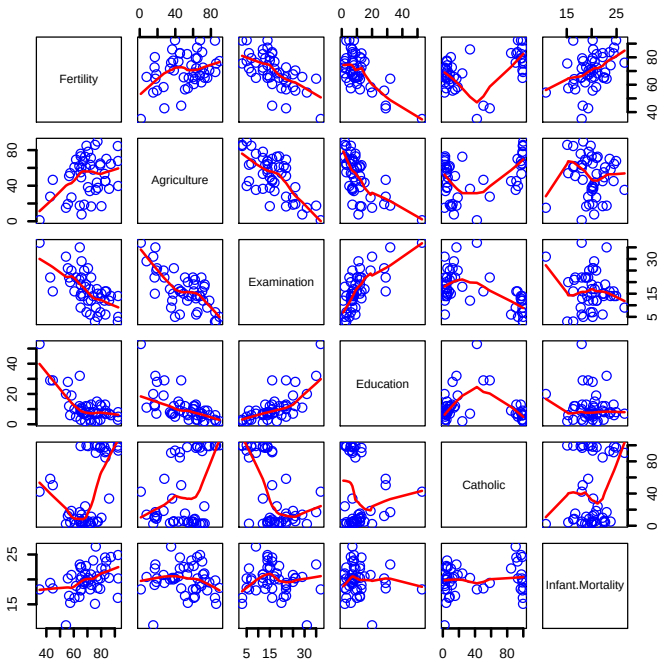


10 Simplepanel plots

```
> pairs(swiss, panel = panel.smooth, pch = ".")
```

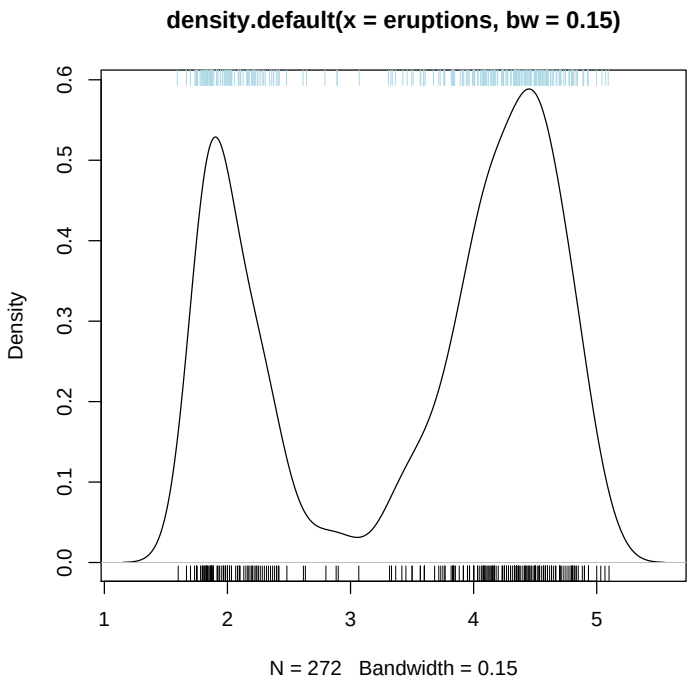



```
> pairs(swiss, panel = panel.smooth, lwd = 2, cex = 1.5,
+       col = "blue")
```



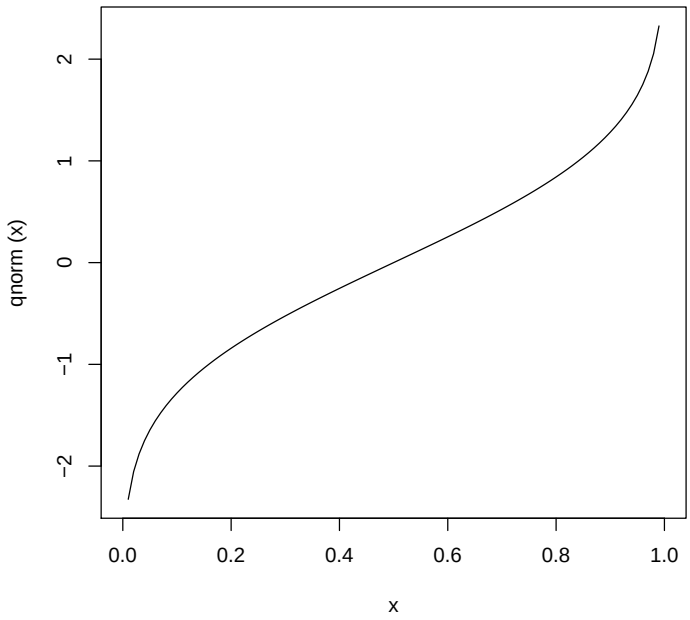
11 Jitter

```
> require(stats)
> with(faithful, {
+     plot(density(eruptions, bw = 0.15))
+     rug(eruptions)
+     rug(jitter(eruptions, amount = 0.01), side = 3,
+         col = "light blue")
+ })
```

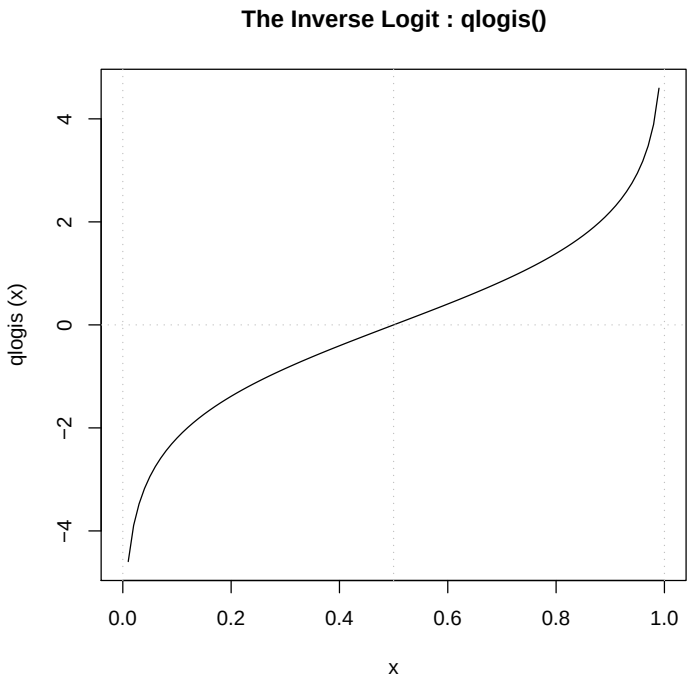


12 Courbe

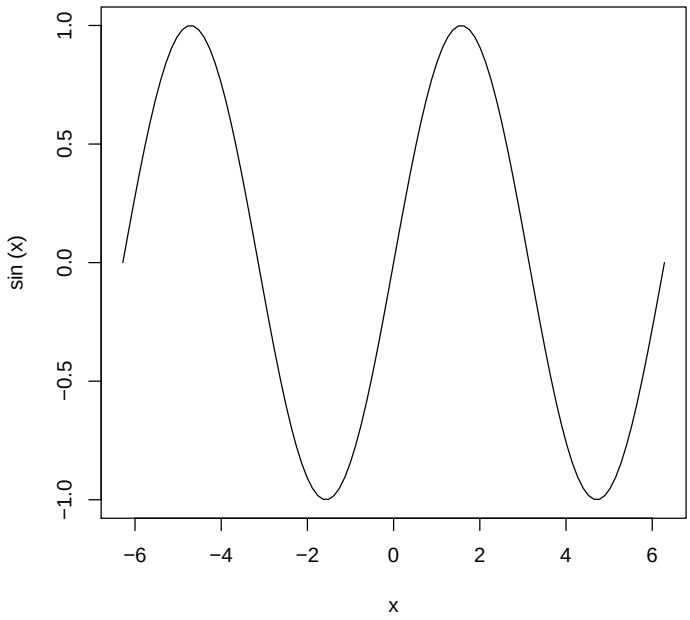
```
> plot(qnorm)
```



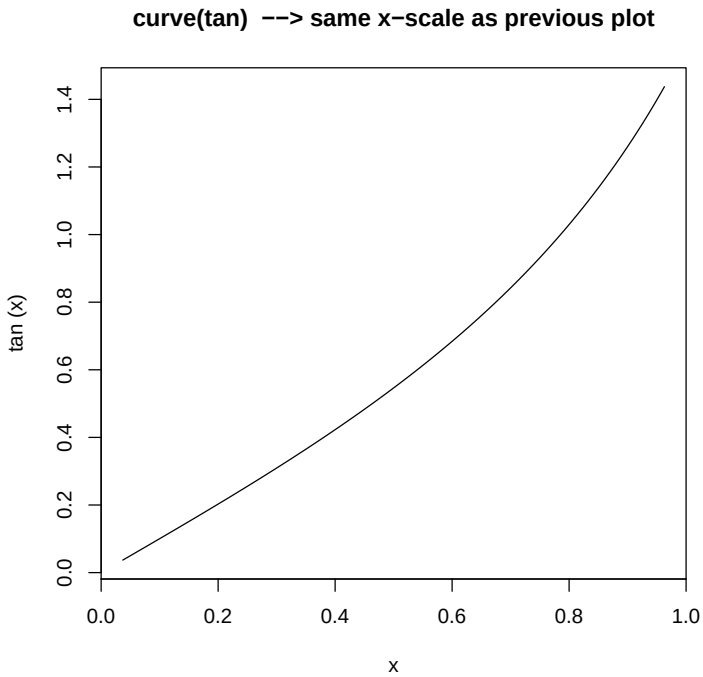

```
> plot(qlogis, main = "The Inverse Logit : qlogis()")
> abline(h = 0, v = 0:2/2, lty = 3, col = "gray")
```




```
> curve(sin, -2 * pi, 2 * pi)
```

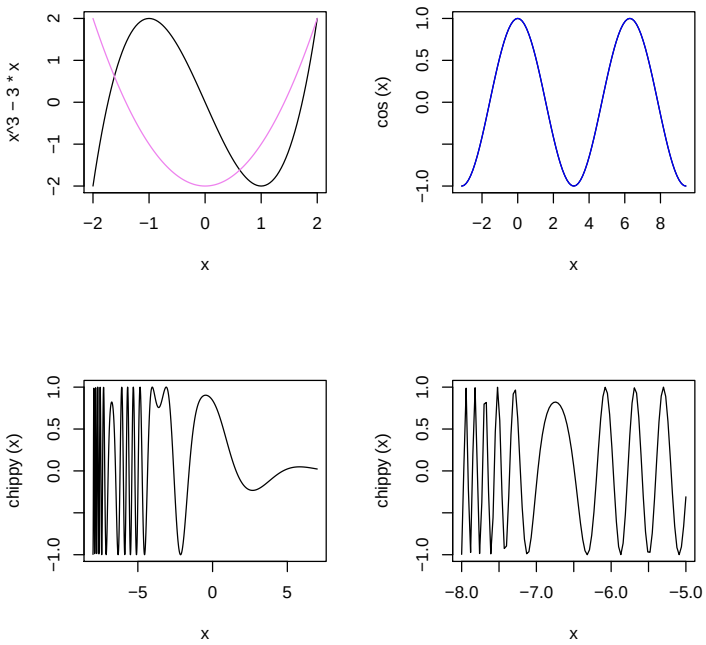



```
> curve(tan, main = "curve(tan)  --> same x-scale as previous plot")
```

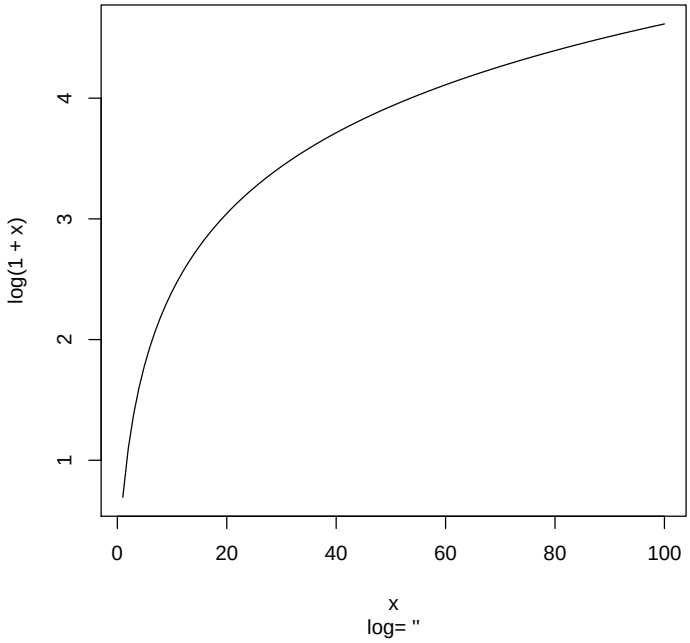



```
> op <- par(mfrow = c(2, 2))
> curve(x^3 - 3 * x, -2, 2)
> curve(x^2 - 2, add = TRUE, col = "violet")
```

```
> plot(cos, -pi, 3 * pi)
> plot(cos, xlim = c(-pi, 3 * pi), n = 1001, col = "blue",
+     add = TRUE)
> chippy <- function(x) sin(cos(x) * exp(-x/2))
> curve(chippy, -8, 7, n = 2001)
> plot(chippy, -8, -5)
```




```
> for (ll in c("", "x", "y", "xy")) curve(log(1 + x),
+     1, 100, log = ll, sub = paste("log= '", ll, "'",
+         sep = ""))
> par(op)
```

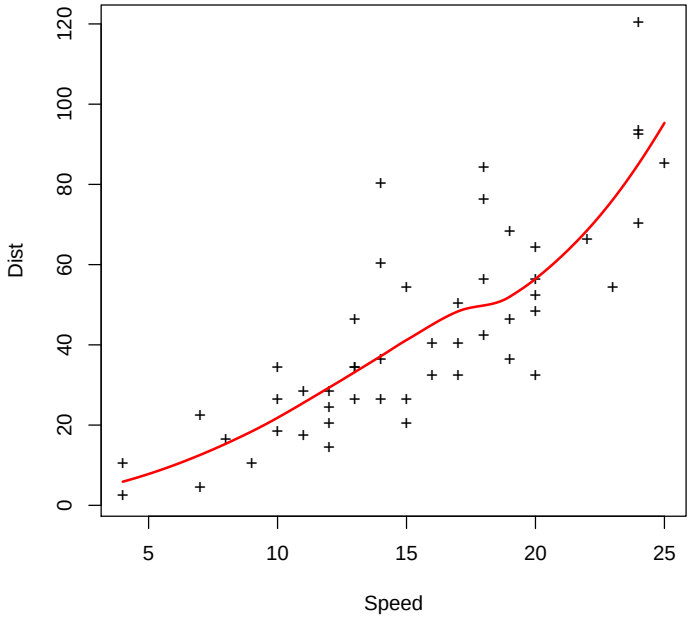


13 Loess (régression non-paramétrique)

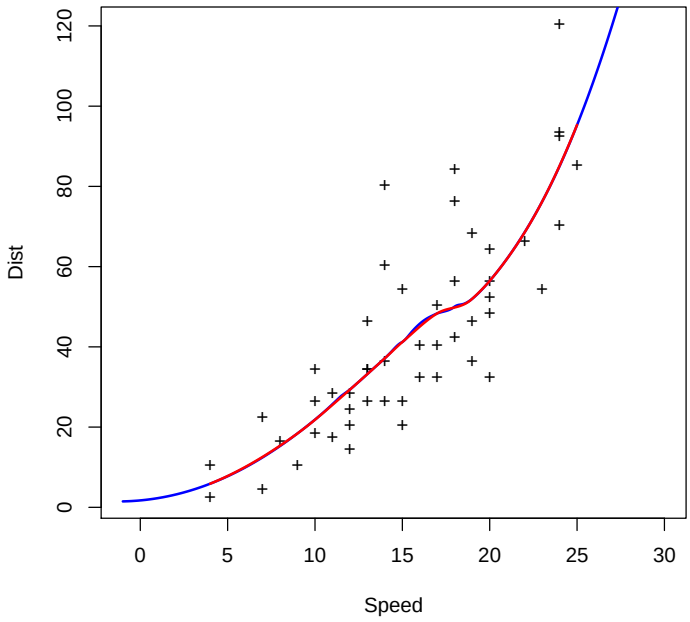
```
> cars.lo <- loess(dist ~ speed, cars)
> predict(cars.lo, data.frame(speed = seq(5, 30, 1)),
+     se = TRUE)

$fit
 [1]  7.810489 10.041808 12.567960 15.369183 18.425712 21.828039
 [7] 25.539675 29.350386 33.230660 37.167935 41.205226 45.055736
[13] 48.355889 49.824812 51.986702 56.445263 62.008703 68.529340
[19] 76.193111 85.142467 95.323096        NA        NA        NA
[25]        NA        NA

$se.fit
 [1] 7.568539 5.943649 4.976453 4.515801 4.316362 4.030120 3.750561
 [8] 3.715593 3.776298 4.091044 4.708759 4.244697 4.035236 3.752765
[15] 4.004017 4.056945 4.005540 4.065234 4.579053 5.948757 8.300416
[22]       NA       NA       NA       NA       NA

$residual.scale
[1] 15.29233

$df
[1] 44.62733
```

to allow extrapolation

```
> cars.lo2 <- loess(dist ~ speed, cars,
+     control = loess.control(surface = "direct"))
> predict(cars.lo2, data.frame(speed = seq(5, 30, 1)),
+     se = TRUE)

$fit
 [1]   7.741006   9.926596  12.442424  15.281082  18.425712
 [6]  21.865315  25.713413  29.350386  33.230660  37.167935
[11]  41.205226  45.781544  48.355889  50.067148  51.986702
[16]  56.445263  62.025404  68.569313  76.193111  85.053364
[21]  95.300523 106.974661 120.092581 134.665851 150.698545
[26] 168.190283

$se.fit
 [1]  7.565991  5.959097  5.012013  4.550013  4.321596  4.119331
 [7]  3.939804  3.720098  3.780877  4.096004  4.714469  4.398936
[13]  4.040129  4.184257  4.008873  4.061865  4.033998  4.078904
[19]  4.584606  5.952480  8.306901 11.601911 15.792480 20.864660
[25] 26.823827 33.683999

$residual.scale
[1] 15.31087

$df
[1] 44.55085

> plot(cars.lo, xlab = "Speed", ylab = "Dist", pch = "+")
> lines(seq(min(cars$speed), max(cars$speed), 0.1), predict(cars.lo,
+     data.frame(speed = seq(min(cars$speed), max(cars$speed),
+         0.1)), se = TRUE)$fit, col = "red", lwd = 2)
```

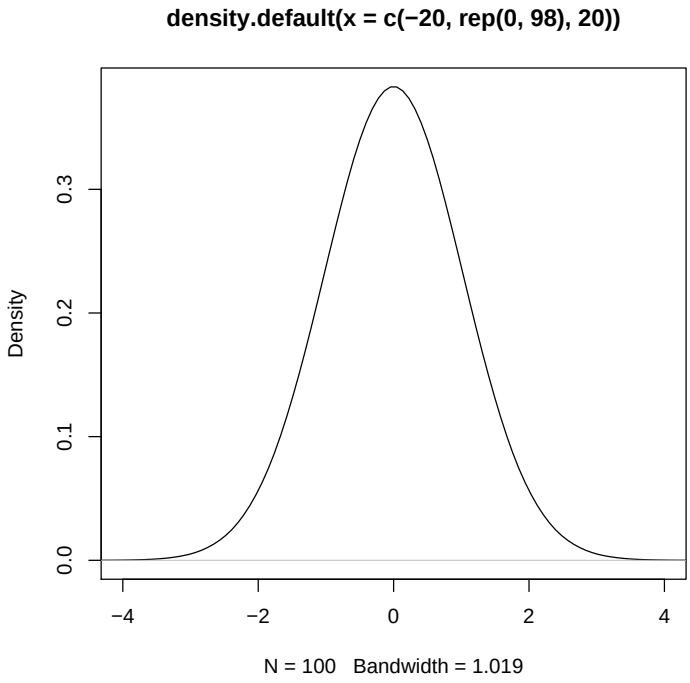



```
> plot(cars.lo2, xlab = "Speed", ylab = "Dist", pch = "+",
+     xlim = c(min(cars$speed) - 5, max(cars$speed) +
+         5))
> lines(seq(min(cars$speed) - 5, max(cars$speed) + 5,
+     0.1), predict(cars.lo2, data.frame(speed = seq(min(cars$speed) -
+     5, max(cars$speed) + 5, 0.1)), se = TRUE)$fit, col = "blue",
+     lwd = 2)
> lines(seq(min(cars$speed) - 5, max(cars$speed) + 5,
+     0.1), predict(cars.lo, data.frame(speed = seq(min(cars$speed) -
+     5, max(cars$speed) + 5, 0.1)), se = TRUE)$fit, col = "red",
+     lwd = 2)
```

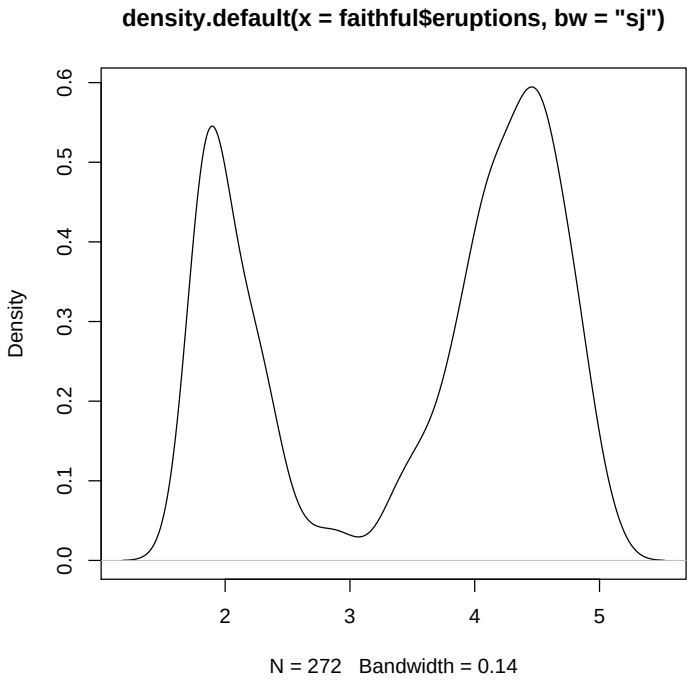


14 Représentation graphique de l'estimation de la fonction de densité

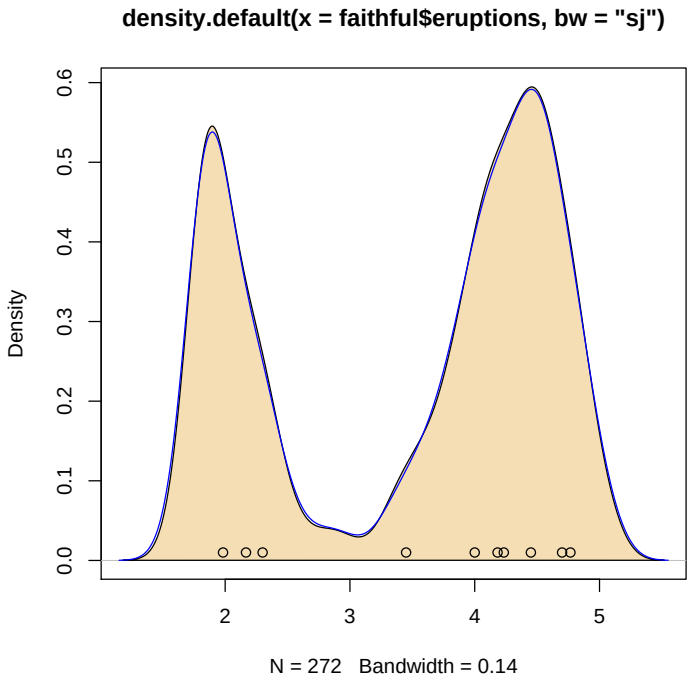
```
> require(graphics)
> plot(density(c(-20, rep(0, 98), 20)), xlim = c(-4, 4))
```



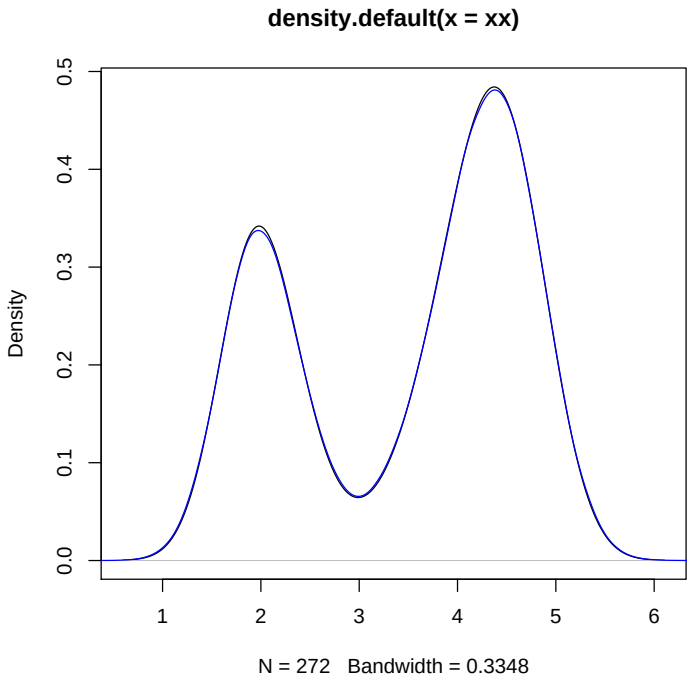

```
> d <- density(faithful$eruptions, bw = "sj")
> d

Call:
        density.default(x = faithful$eruptions, bw = "sj")

Data: faithful$eruptions (272 obs.);        Bandwidth ‘bw’ = 0.14

       x               y
 Min.   :1.180   Min.   :0.0001834
 1st Qu.:2.265   1st Qu.:0.0422638
 Median :3.350   Median :0.1709243
 Mean   :3.350   Mean   :0.2301726
 3rd Qu.:4.435   3rd Qu.:0.4134348
 Max.   :5.520   Max.   :0.5945634

> plot(d)
```



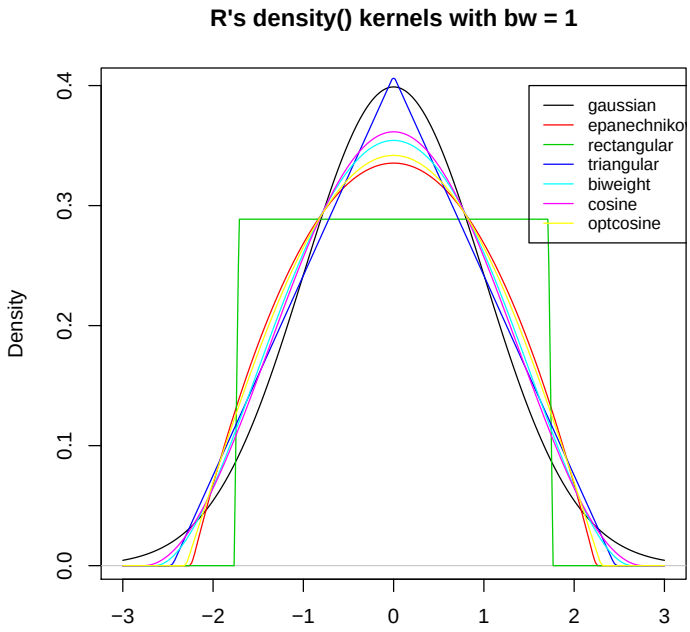

```
> plot(d, type = "n")
> polygon(d, col = "wheat")
> x <- xx <- faithful$eruptions
> x[i.out <- sample(length(x), 10)] <- NA
> doR <- density(x, bw = 0.15, na.rm = TRUE)
> lines(doR, col = "blue")
> points(xx[i.out], rep(0.01, 10))
```



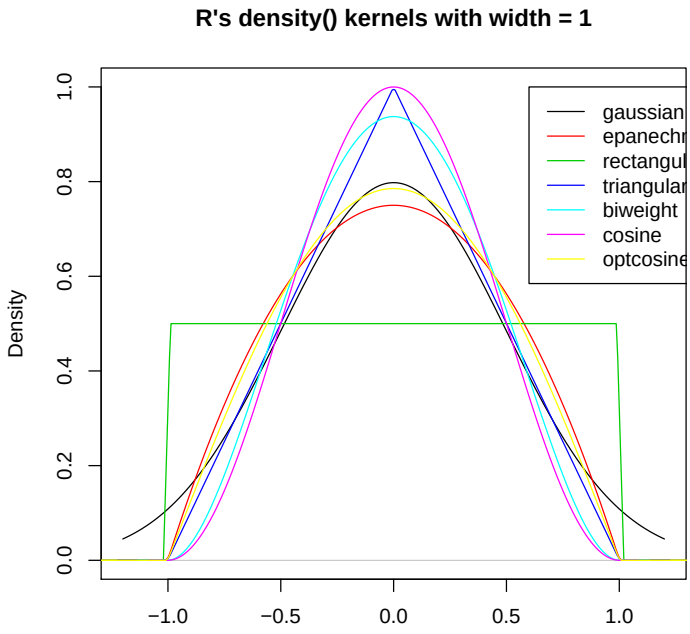

```
> fe <- sort(faithful$eruptions)
> dw <- density(unique(fe), weights = table(fe)/length(fe),
+     bw = d$bw)
> utils::str(dw)

List of 7
 $ x        : num [1:512] 1.18 1.19 1.2 1.21 1.21 ...
 $ y        : num [1:512] 0.000183 0.000223 0.00027 0.000328 0.000397 ...
 $ bw       : num 0.14
 $ n        : int 126
 $ call     : language density.default(x = unique(fe), bw = d$bw,
              weights = table(fe)/length(fe))
 $ data.name: chr "unique(fe)"
 $ has.na   : logi FALSE
 - attr(*, "class")= chr "density"

> stopifnot(all.equal(d[1:3], dw[1:3]))

> fit <- density(xx)
> N <- 1e+06
> x.new <- rnorm(N, sample(xx, size = N, replace = TRUE),
+     fit$bw)
> plot(fit)
> lines(density(x.new), col = "blue")
```




```
> (kernels <- eval(formals(density.default)$kernel))

[1] "gaussian"     "epanechnikov" "rectangular"  "triangular"
[5] "biweight"     "cosine"       "optcosine"

> plot(density(0, bw = 1), xlab = "", main = "R's density() kernels
+     with bw = 1")
> for (i in 2:length(kernels)) lines(density(0, bw = 1,
+     kernel = kernels[i]), col = i)
> legend(1.5, 0.4, legend = kernels, col = seq(kernels),
+     lty = 1, cex = 0.8, y.intersp = 1)
```

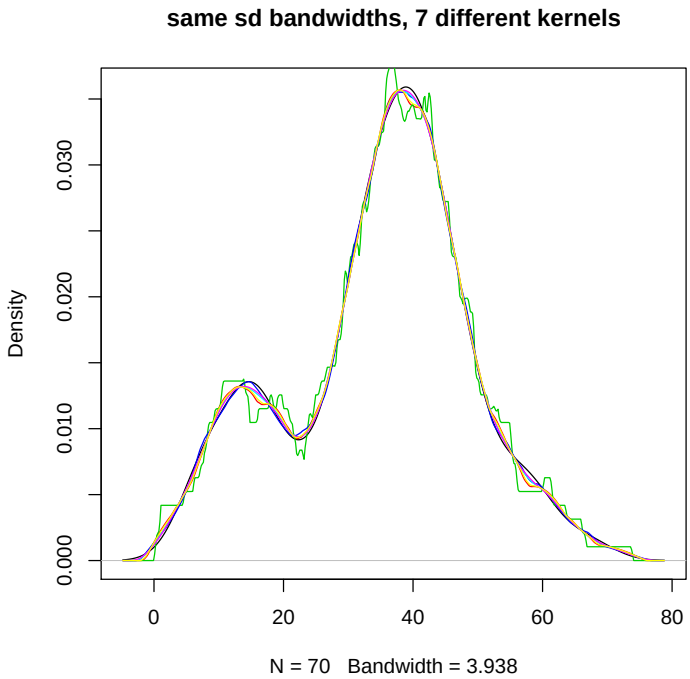



```
> plot(density(0, from = -1.2, to = 1.2, width = 2, kernel =
+     "gaussian"), type = "l", ylim = c(0, 1), xlab = "",
+     main = "R's density() kernels with width = 1")
> for (i in 2:length(kernels)) lines(density(0, width = 2,
+     kernel = kernels[i]), col = i)
> legend(0.6, 1, legend = kernels, col = seq(kernels),
+     lty = 1)
```

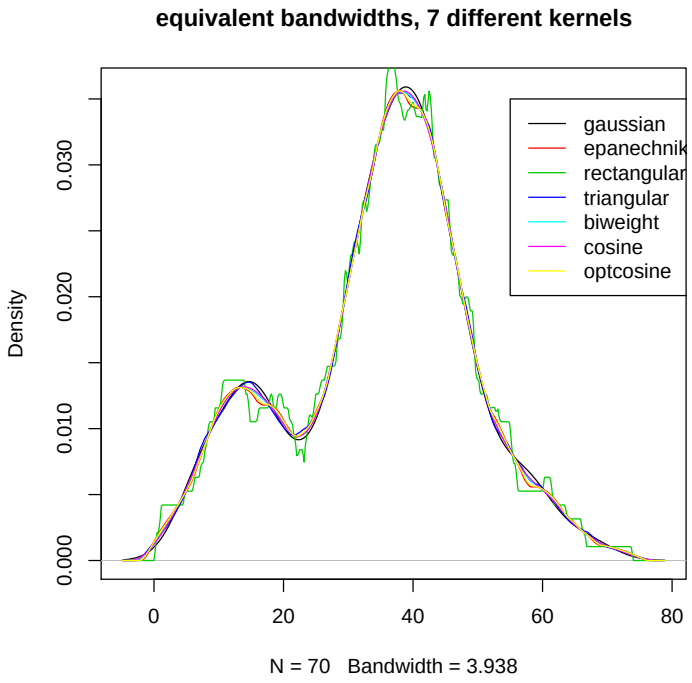



```
> (RKs <- cbind(sapply(kernels, function(k) density(kernel = k,
+     give.Rkern = TRUE))))

                  [,1]
gaussian     0.2820948
epanechnikov 0.2683282
rectangular  0.2886751
triangular   0.2721655
biweight     0.2699746
cosine       0.2711340
optcosine    0.2684756

> 100 * round(RKs["epanechnikov", ]/RKs, 4)

               [,1]
gaussian      95.12
epanechnikov 100.00
rectangular   92.95
triangular    98.59
biweight      99.39
cosine        98.97
optcosine     99.95

> bw <- bw.SJ(precip)
> plot(density(precip, bw = bw), main = "same sd bandwidths,
+     7 different kernels")
```

```
> for (i in 2:length(kernels)) lines(density(precip, bw = bw,
+     kernel = kernels[i]), col = i)
```

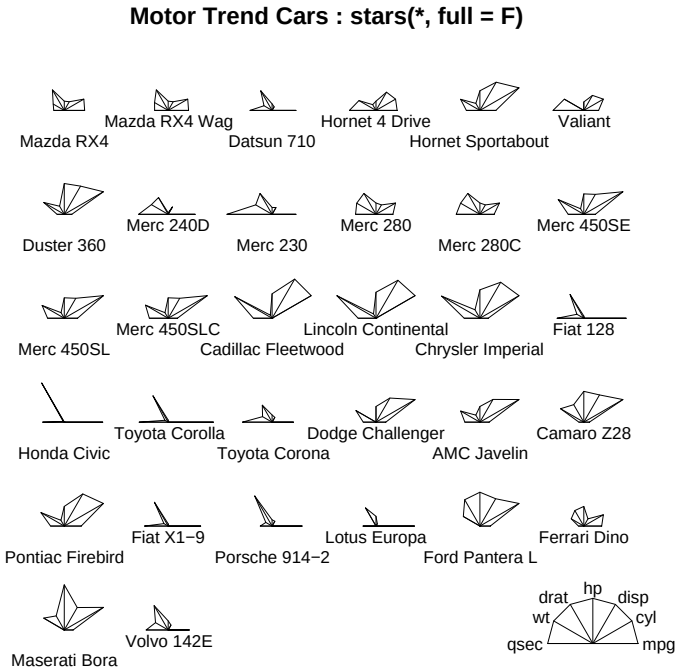



```
> h.f <- sapply(kernels, function(k) density(kernel = k,
+     give.Rkern = TRUE))
> (h.f <- (h.f["gaussian"]/h.f)^0.2)

    gaussian epanechnikov  rectangular   triangular     biweight
   1.0000000    1.0100567    0.9953989    1.0071923    1.0088217
      cosine    optcosine
   1.0079575    1.0099458

> plot(density(precip, bw = bw), main = "equivalent bandwidths,
+     7 different kernels")
> for (i in 2:length(kernels)) lines(density(precip, bw = bw,
+     adjust = h.f[i], kernel = kernels[i]), col = i)
> legend(55, 0.035, legend = kernels, col = seq(kernels),
+     lty = 1)
```

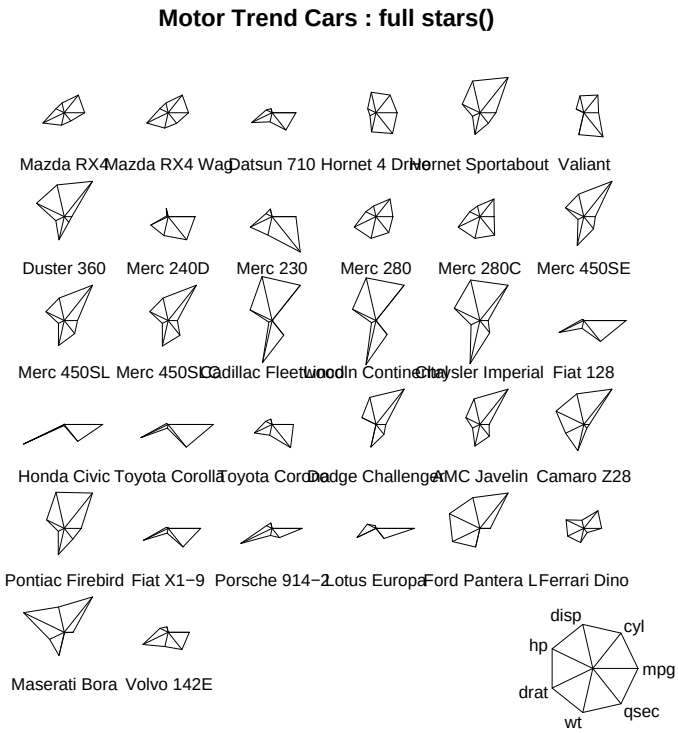


15 Radar plots

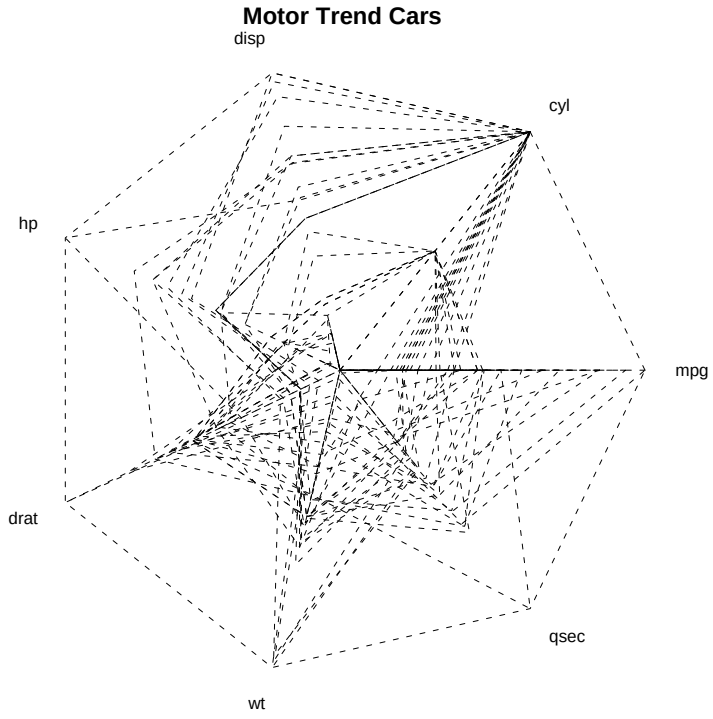
```
> require(grDevices)
> stars(mtcars[, 1:7], key.loc = c(14, 2), main = "Motor Trend Cars :
+     stars(*, full = F)", full = FALSE)
```



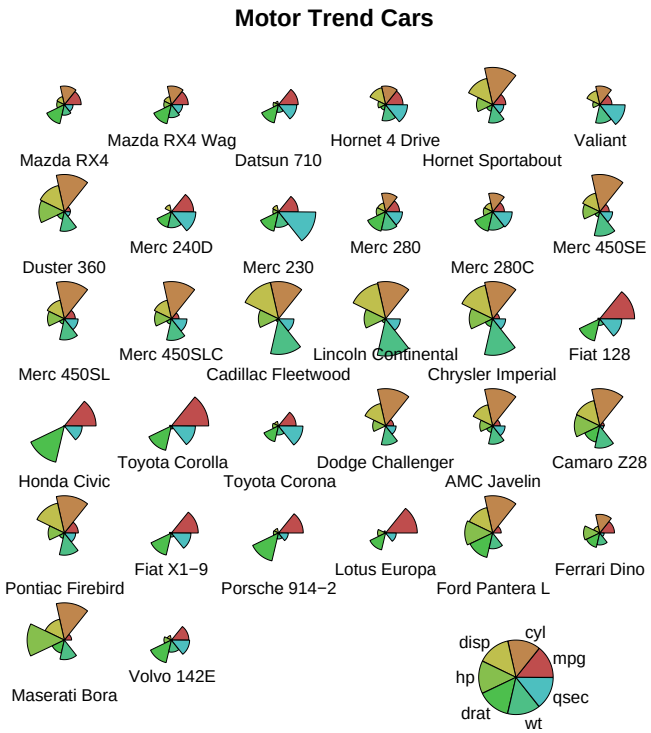

```
> stars(mtcars[, 1:7], key.loc = c(14, 1.5), main = "Motor Trend Cars :
+     full stars()", flip.labels = FALSE)
```




```
> stars(mtcars[, 1:7], locations = c(0, 0), radius = FALSE,
+     key.loc = c(0, 0), main = "Motor Trend Cars", lty = 2)
```

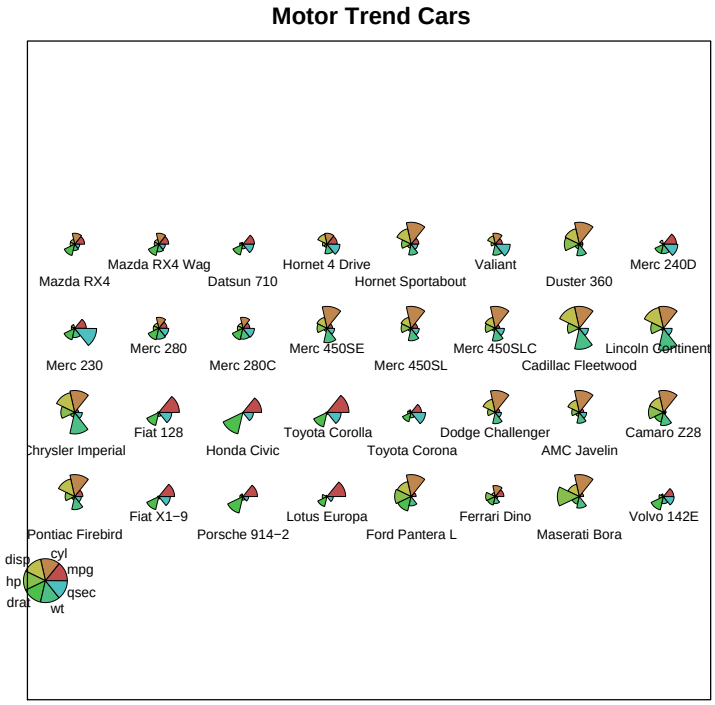



```
> palette(rainbow(12, s = 0.6, v = 0.75))
> stars(mtcars[, 1:7], len = 0.8, key.loc = c(12, 1.5),
+     main = "Motor Trend Cars", draw.segments = TRUE)
```

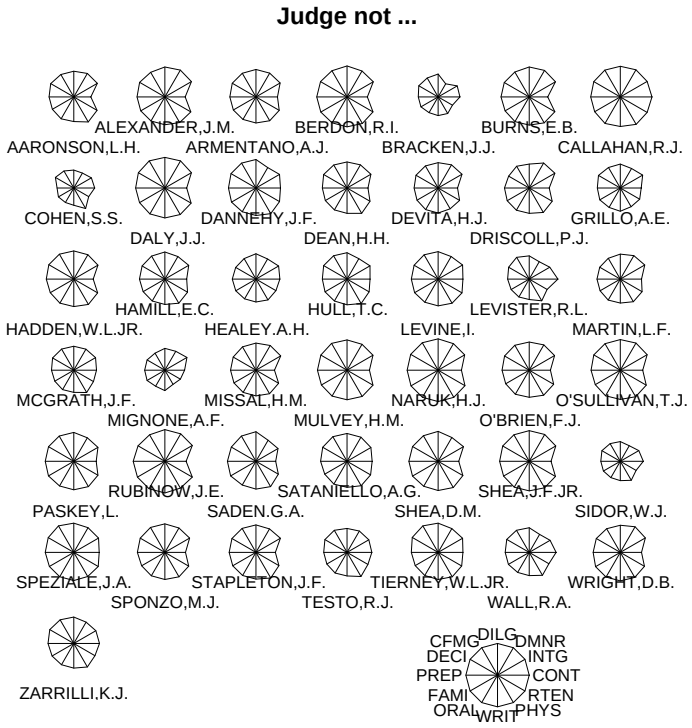



```
> stars(mtcars[, 1:7], len = 0.6, key.loc = c(1.5, 0),
+     main = "Motor Trend Cars", draw.segments = TRUE,
```

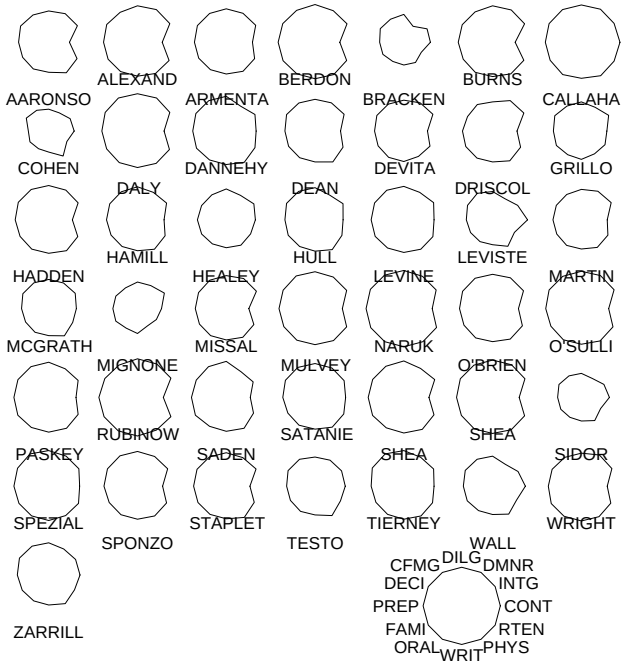
```
+     frame.plot = TRUE, nrow = 4, cex = 0.7)
```




```
> USJudge <- apply(USJudgeRatings, 2, function(x) x/max(x))
> Jnam <- row.names(USJudgeRatings)
> Snam <- abbreviate(substring(Jnam, 1, regexpr("[,.]",
+     Jnam) - 1), 7)
> stars(USJudge, labels = Jnam, scale = FALSE, key.loc = c(13,
+     1.5), main = "Judge not ...", len = 0.8)
```

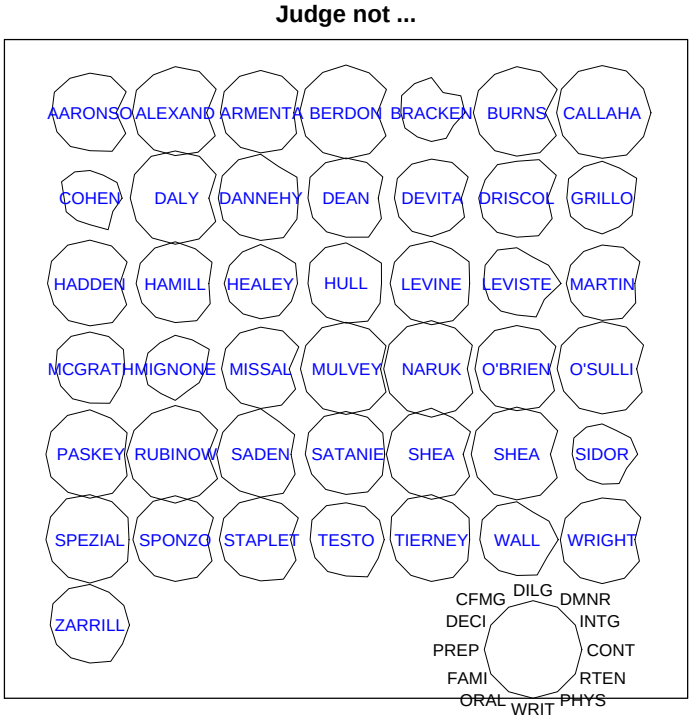



```
> stars(USJudge, labels = Snam, scale = FALSE, key.loc = c(13,
+     1.5), radius = FALSE)
```

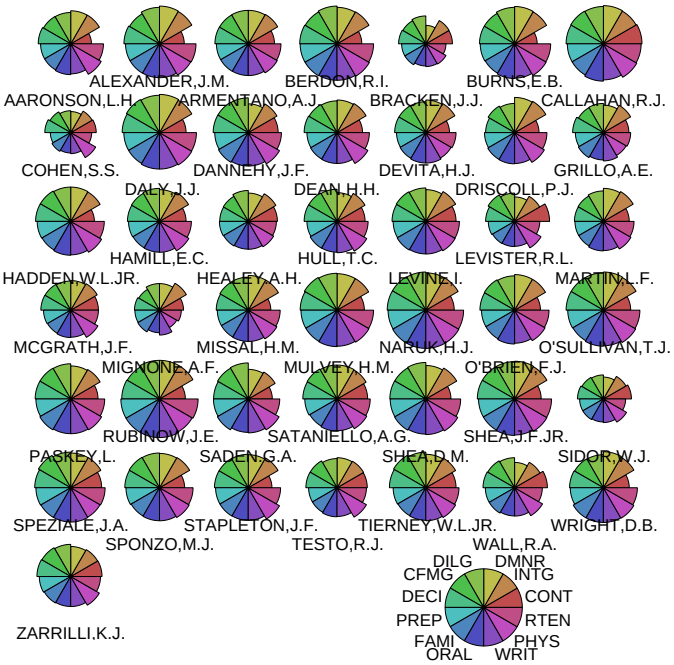



```
> loc <- stars(USJudge, labels = NULL, scale = FALSE,
+     radius = FALSE, frame.plot = TRUE, key.loc = c(13,
```

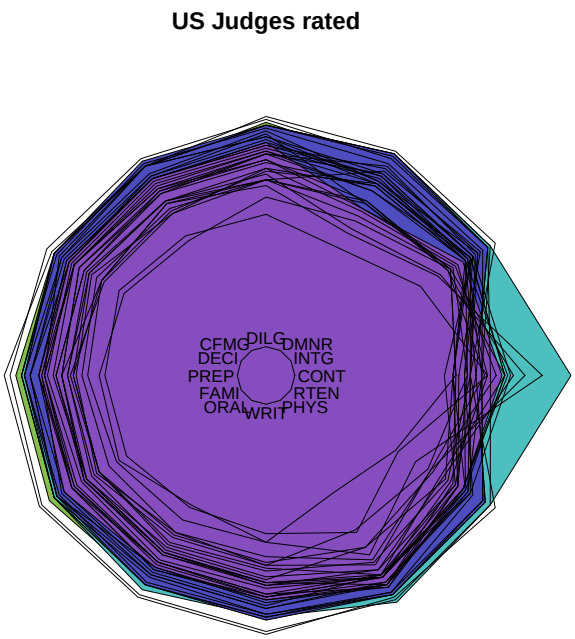
```
+          1.5), main = "Judge not ...", len = 1.2)
> text(loc, Snam, col = "blue", cex = 0.8, xpd = TRUE)
```



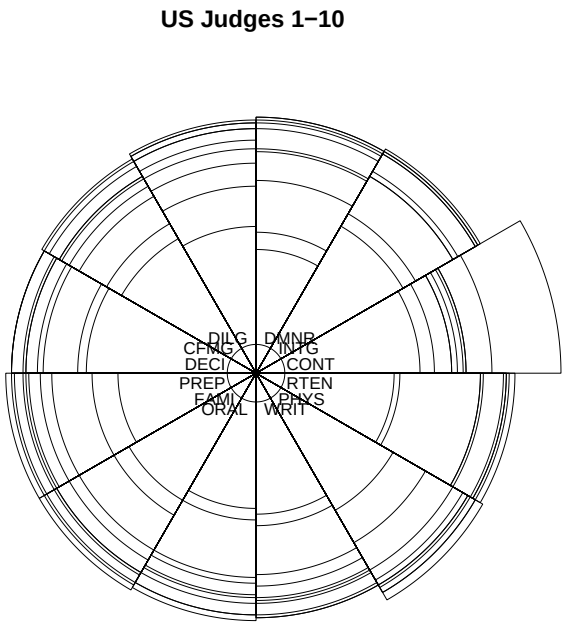

```
> stars(USJudge, draw.segments = TRUE, scale = FALSE,
+     key.loc = c(13, 1.5))
```



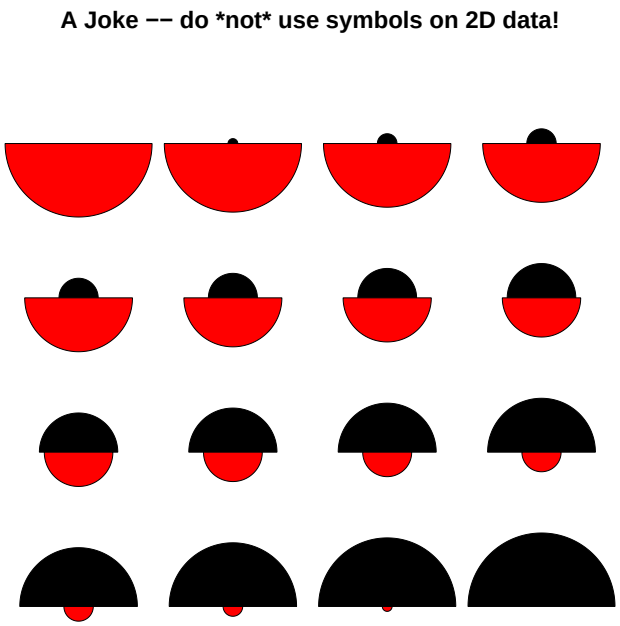
```
> stars(USJudgeRatings, locations = c(0, 0), scale = FALSE,
+     radius = FALSE, col.stars = 1:10, key.loc = c(0,
+         0), main = "US Judges rated")
```




```
> stars(USJudgeRatings[1:10, ], locations = 0:1, scale = FALSE,
+     draw.segments = TRUE, col.segments = 0, col.stars = 1:10,
+     key.loc = 0:1, main = "US Judges 1-10 ")
```




```
> palette("default")
> stars(cbind(1:16, 10 * (16:1)), draw.segments = TRUE,
+     main = "A Joke -- do *not* use symbols on 2D data!")
```

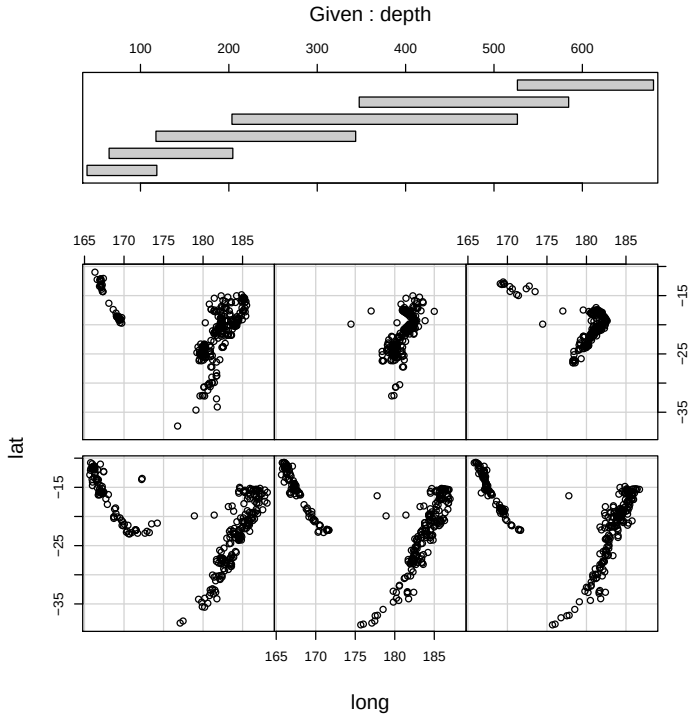


16 Steam and leaf

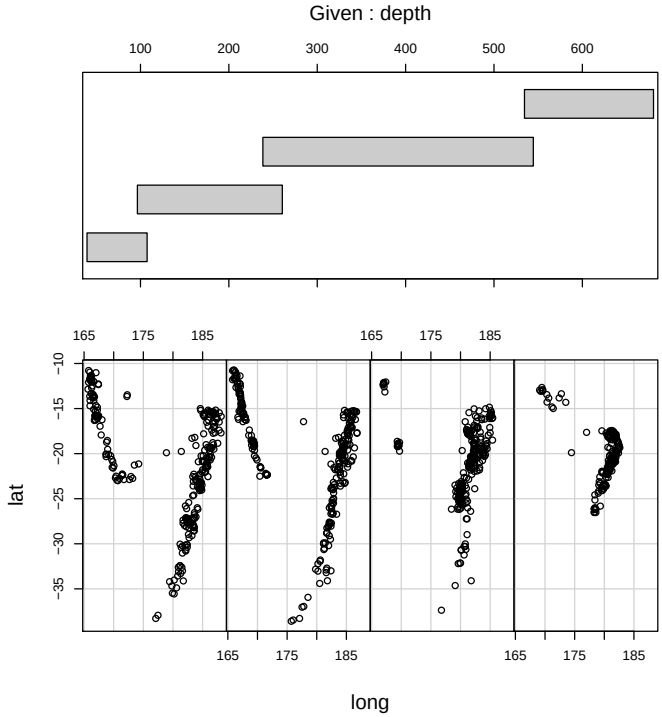
```
> stem(islands)

  The decimal point is 3 digit(s) to the right of the |

   0 | 00000000000000000000000000000111111222338
   2 | 07
   4 | 5
   6 | 8
   8 | 4
  10 | 5
  12 |
  14 |
  16 | 0

> stem(log10(islands))

  The decimal point is at the |

  1 | 1111112222233444
  1 | 5555556666667899999
  2 | 3344
  2 | 59
  3 |
  3 | 5678
  4 | 012
```

17 Conditioning plots

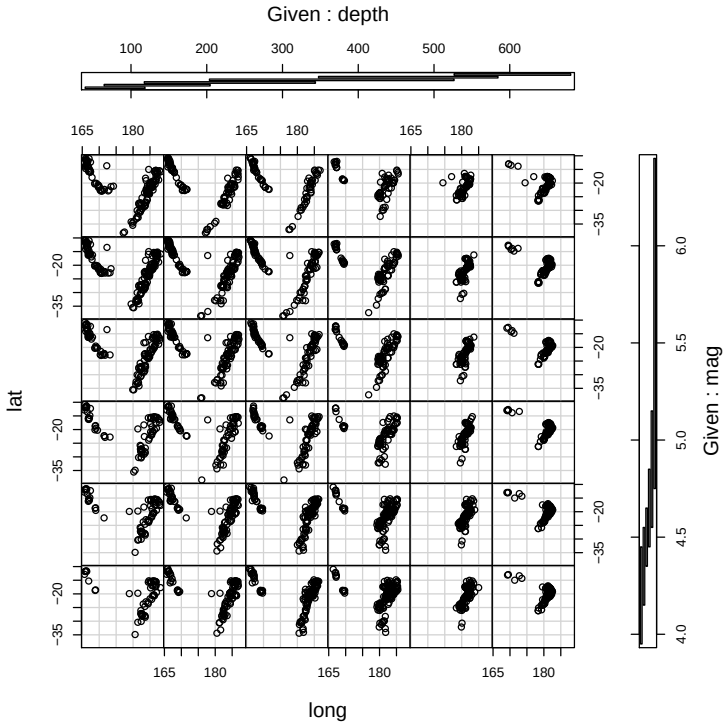
```
> coplot(lat ~ long | depth, data = quakes)
```



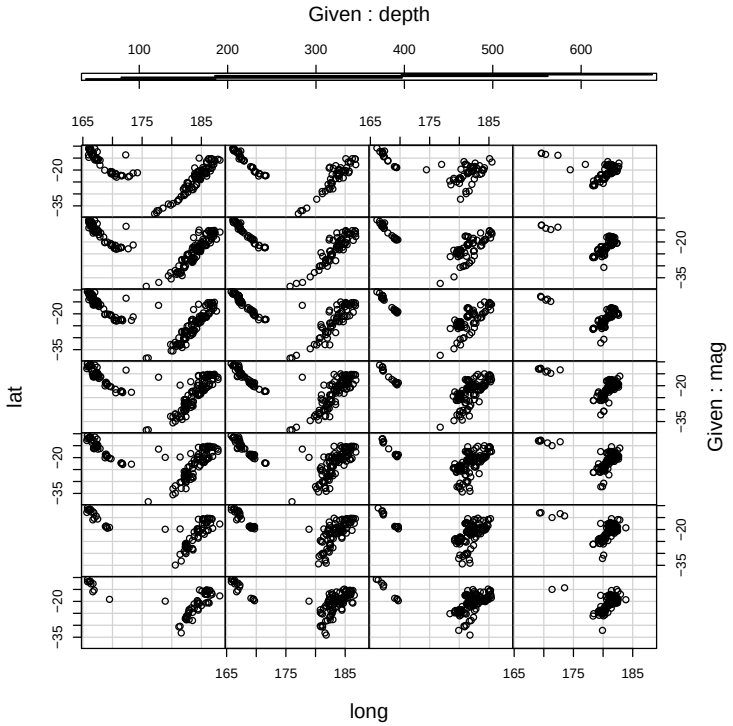

```
> given.depth <- co.intervals(quakes$depth, number = 4,
+     overlap = 0.1)
> coplot(lat ~ long | depth, data = quakes, given.v = given.depth,
+     rows = 1)
```



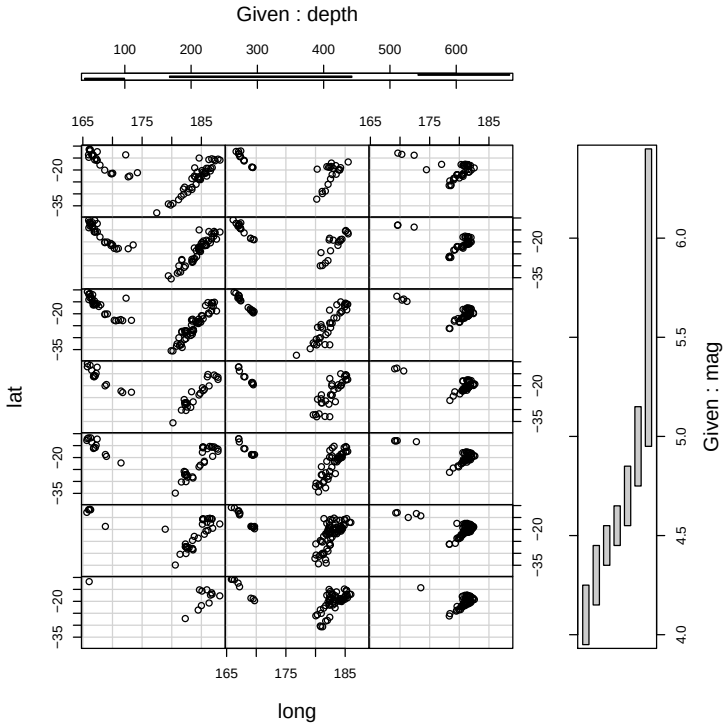
```
> ll.dm <- lat ~ long | depth * mag
> coplot(ll.dm, data = quakes)
```



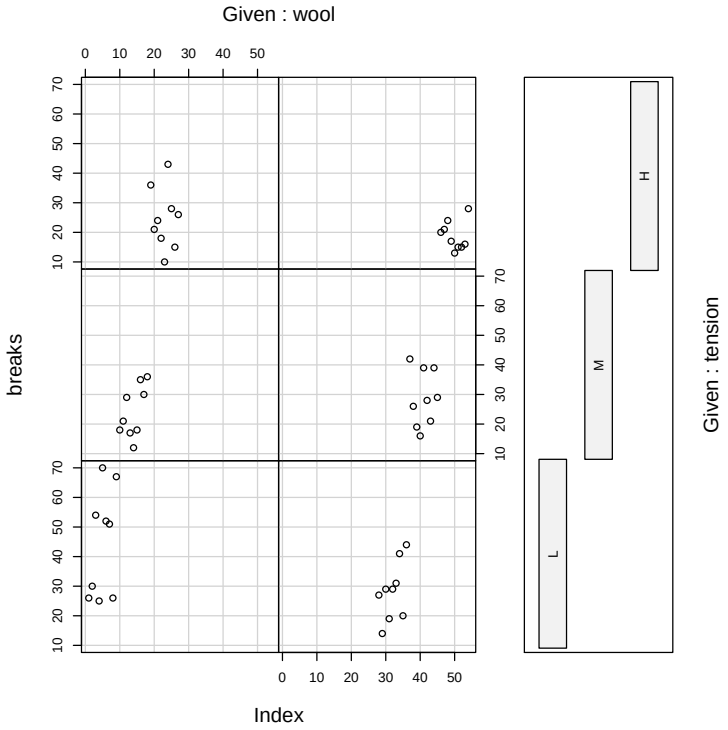

```
> coplot(ll.dm, data = quakes, number = c(4, 7), show.given = c(TRUE,
+     FALSE))
```



```
> coplot(ll.dm, data = quakes, number = c(3, 7), overlap = c(-0.5,
+     0.1))
```



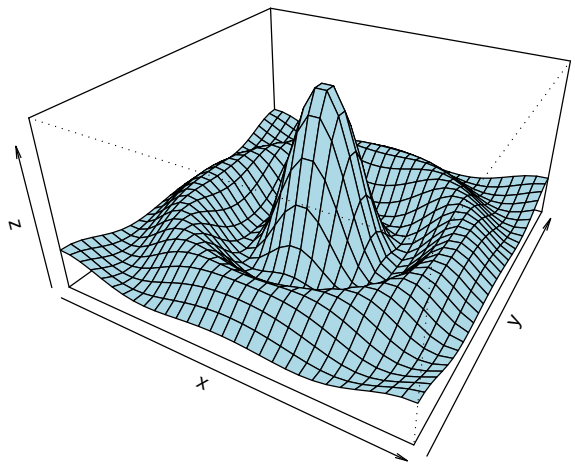

```
> Index <- seq(length = nrow(warpbreaks))
> coplot(breaks ~ Index | wool * tension, data = warpbreaks,
+     show.given = 0:1)
```



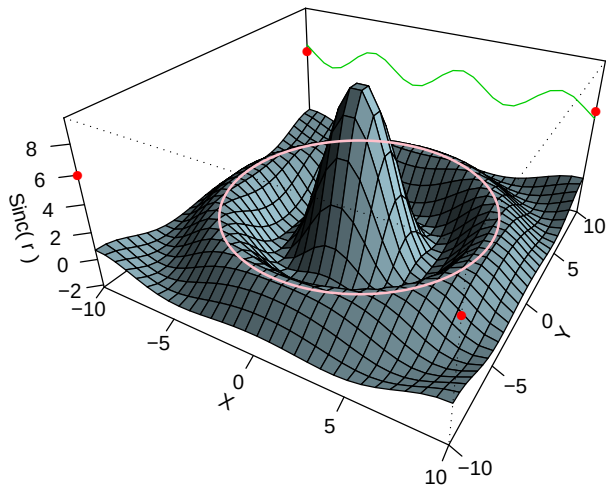
```
> coplot(breaks ~ Index | wool * tension, data = warpbreaks,
+     col = "red", bg = "pink", pch = 21, bar.bg = c(fac = "light blue"))

> with(data.frame(state.x77), {
+     coplot(Life.Exp ~ Income | Illiteracy * state.region,
+         number = 3, panel = function(x, y, ...) panel.smooth(x,
+             y, span = 0.8, ...))
+ })

> with(data.frame(state.x77), {
+     coplot(Life.Exp ~ state.region | Income * state.division,
+         panel = panel.smooth)
+ })
```

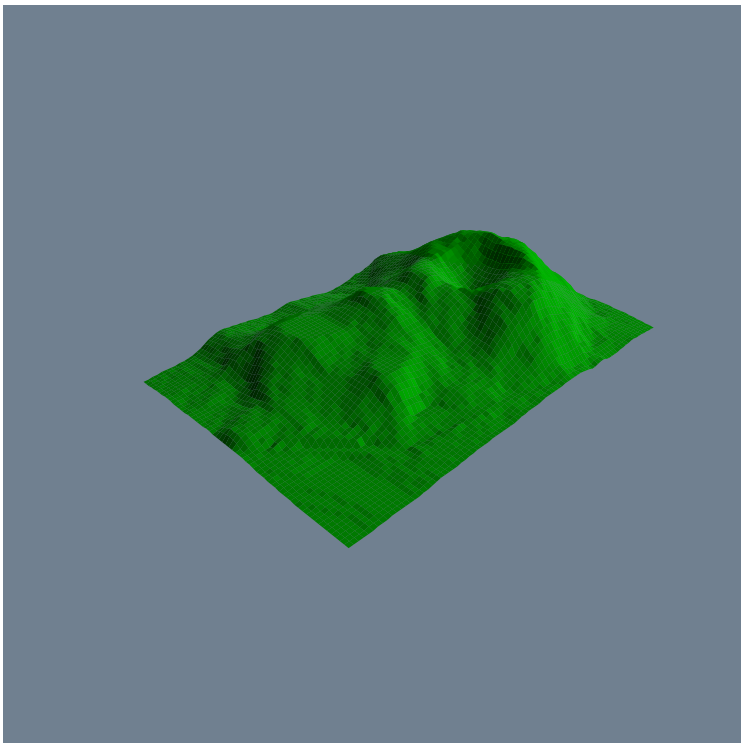
18 Persp

```
> require(grDevices)
> 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 = "white")

> persp(x, y, z, theta = 30, phi = 30, expand = 0.5, col = "lightblue")
```




```
> res <- 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 )")
> round(res, 3)

      [,1]   [,2]   [,3]   [,4]
[1,] 0.087 -0.025  0.043 -0.043
[2,] 0.050  0.043 -0.075  0.075
[3,] 0.000  0.074  0.042 -0.042
[4,] 0.000 -0.273 -2.890  3.890

> 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
> xr <- r1 * cos(phi)
> yr <- r1 * sin(phi)
> lines(trans3d(xr, yr, f(xr, yr), res), col = "pink",
+     lwd = 2)
```




```
> z <- 2 * volcano
> x <- 10 * (1:nrow(z))
> y <- 10 * (1:ncol(z))
> par(bg = "slategray")
> persp(x, y, z, theta = 135, phi = 30, col = "green3",
+     scale = FALSE, ltheta = -120, shade = 0.75, border = NA,
+     box = FALSE)
> par(op)
```