

# Feuille de Travaux Dirigés n° 4

## Les graphiques et R

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

### 1 Contenu

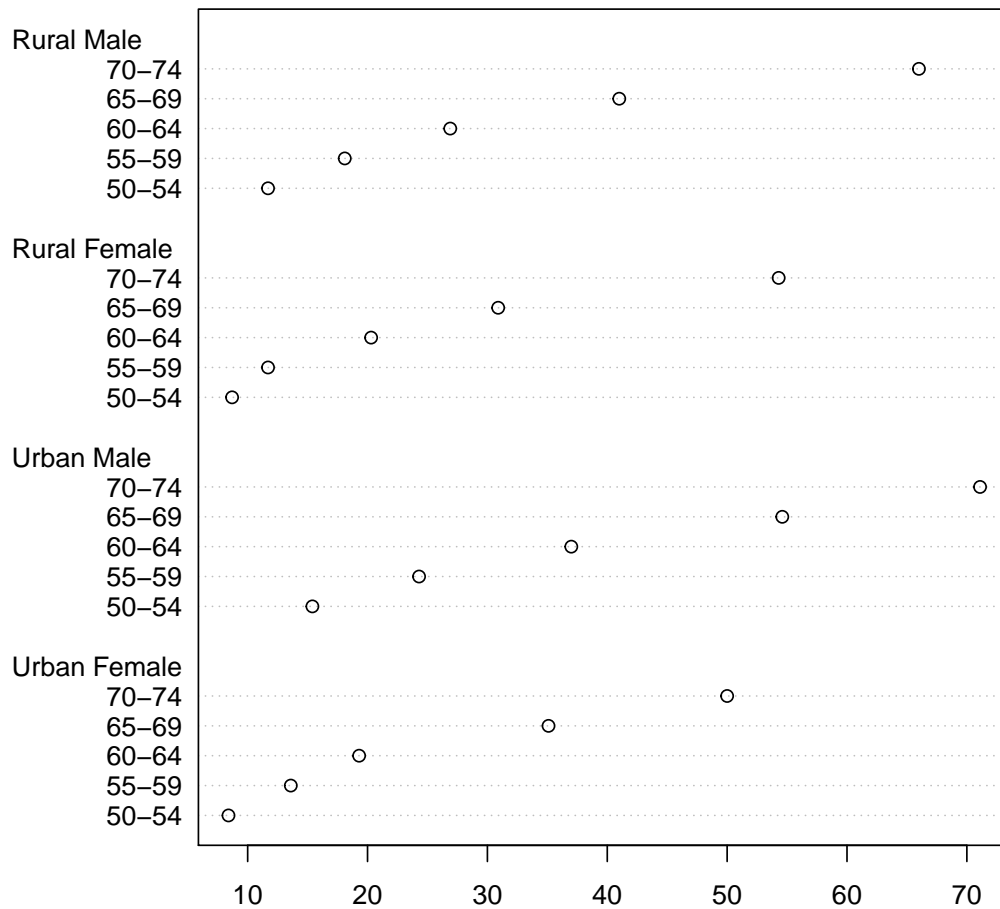
Nous allons nous intéresser à différents types de représentations graphiques adaptées à la nature des variables que nous souhaitons représenter.

```
library(graphics)
```

### 2 dotchart

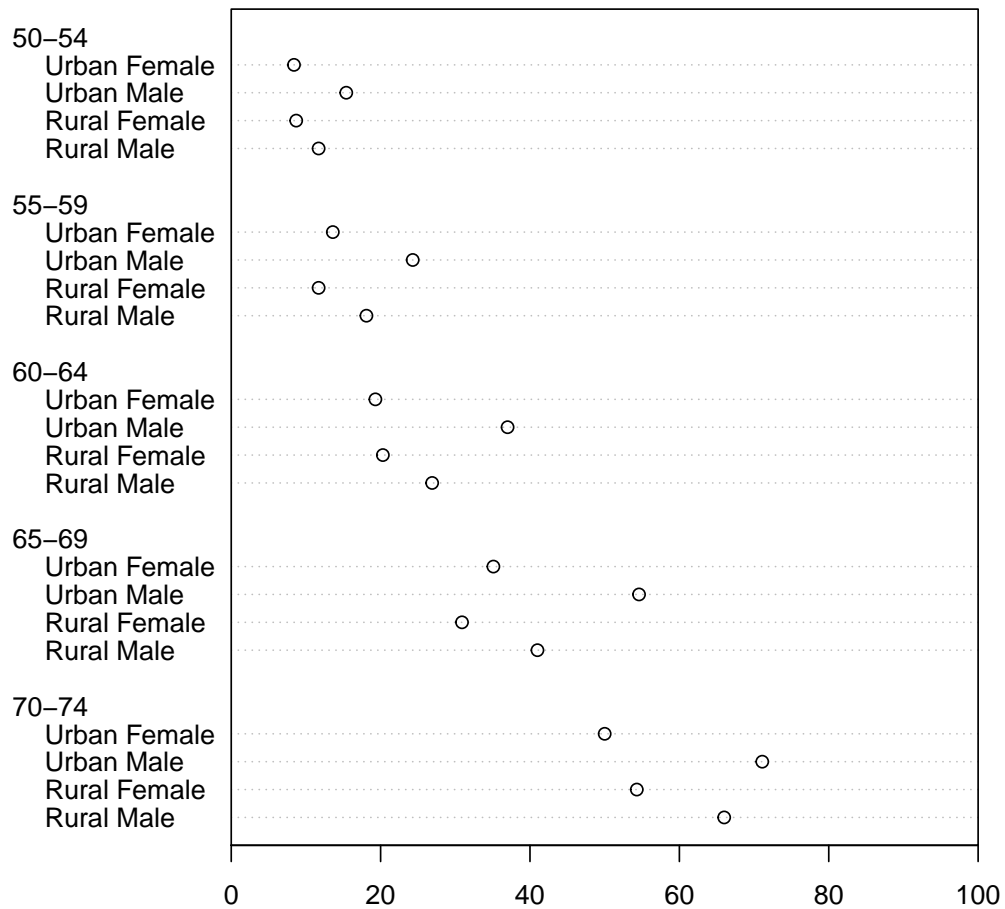
```
dotchart(VADeaths, main = "Death Rates in Virginia - 1940")
```

## Death Rates in Virginia – 1940



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

## Death Rates in Virginia – 1940

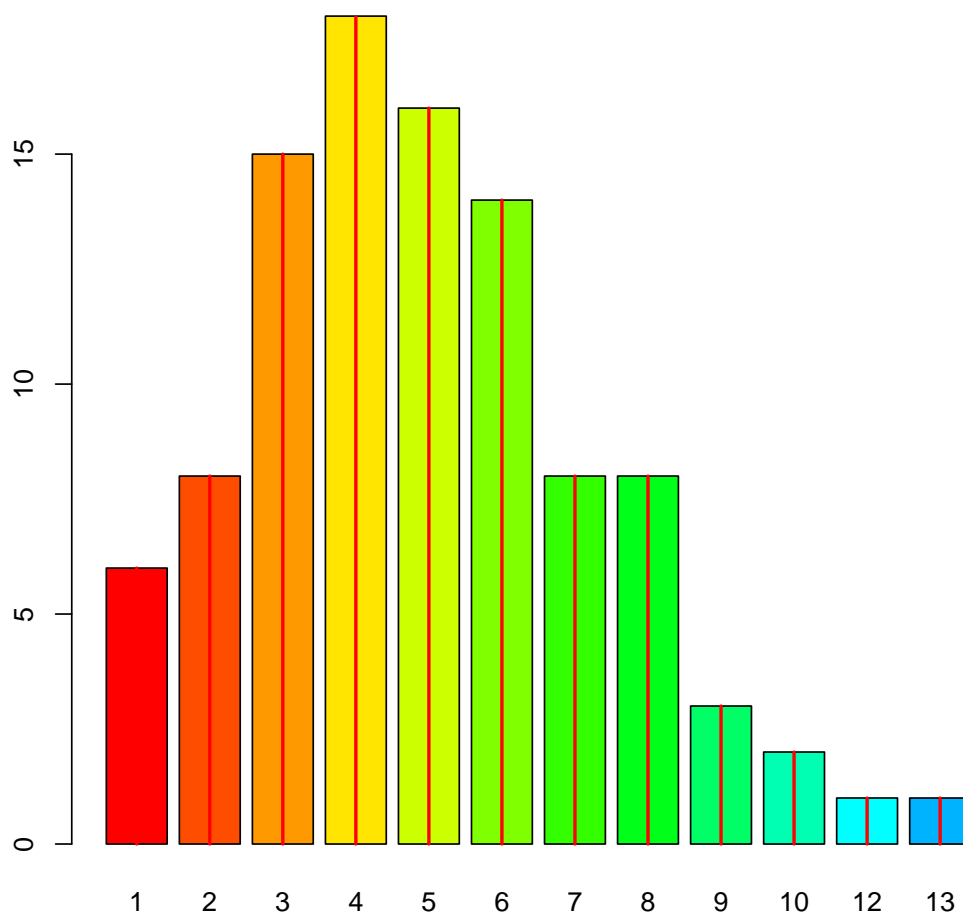


```
par(op)
```

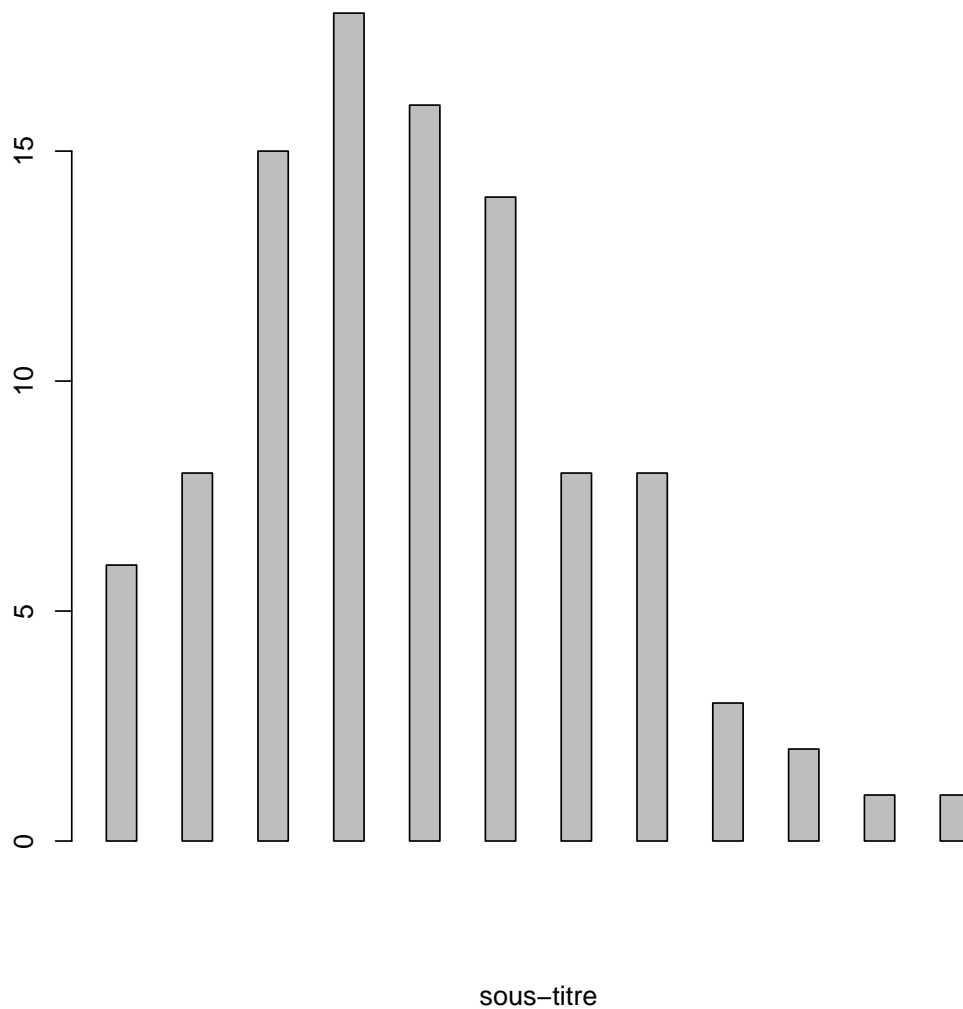
### 3 barplot

```
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")
```



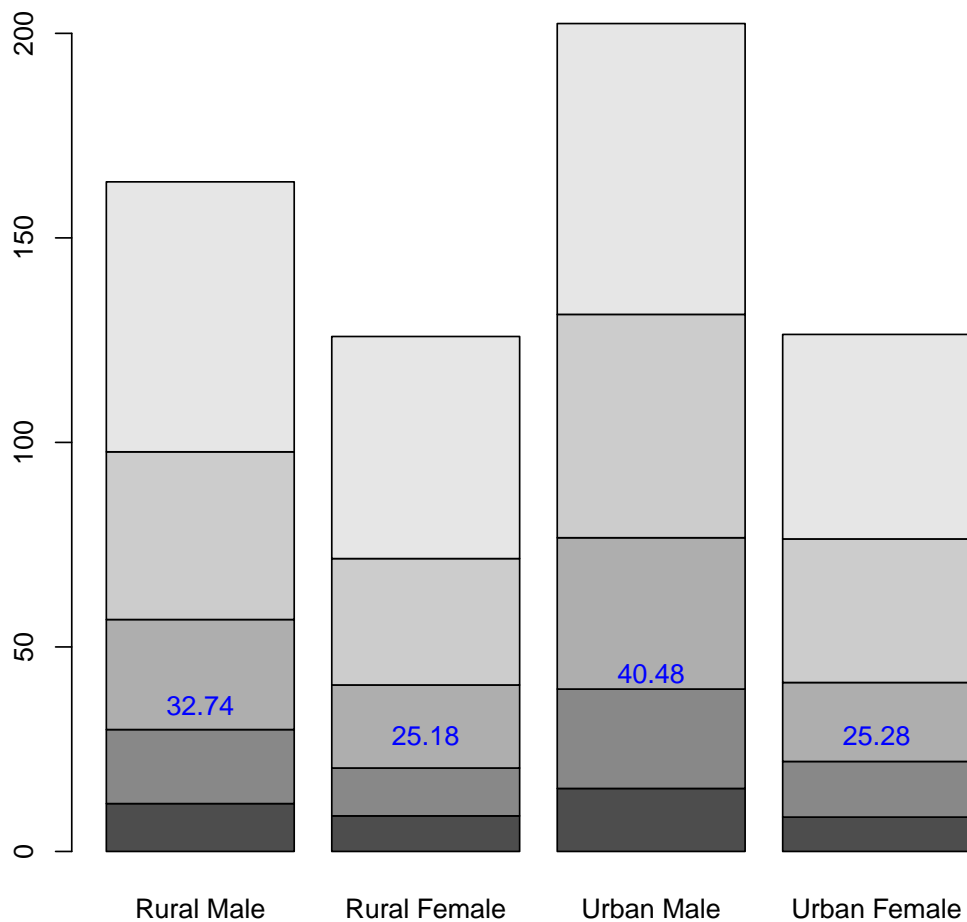
```
barplot(VADeaths, plot = FALSE)
```

```
## [1] 0.7 1.9 3.1 4.3
```

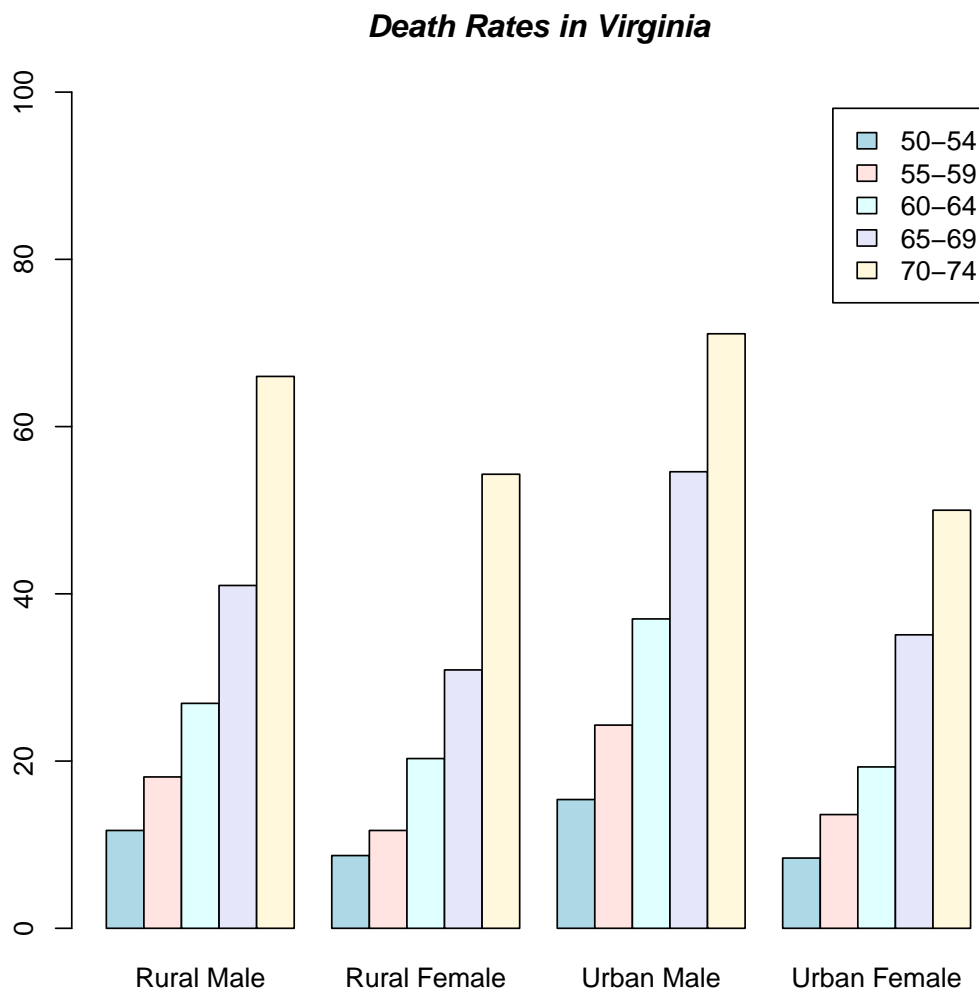
```
barplot(VADeaths, plot = FALSE, beside = TRUE)
```

```
##      [,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")
```



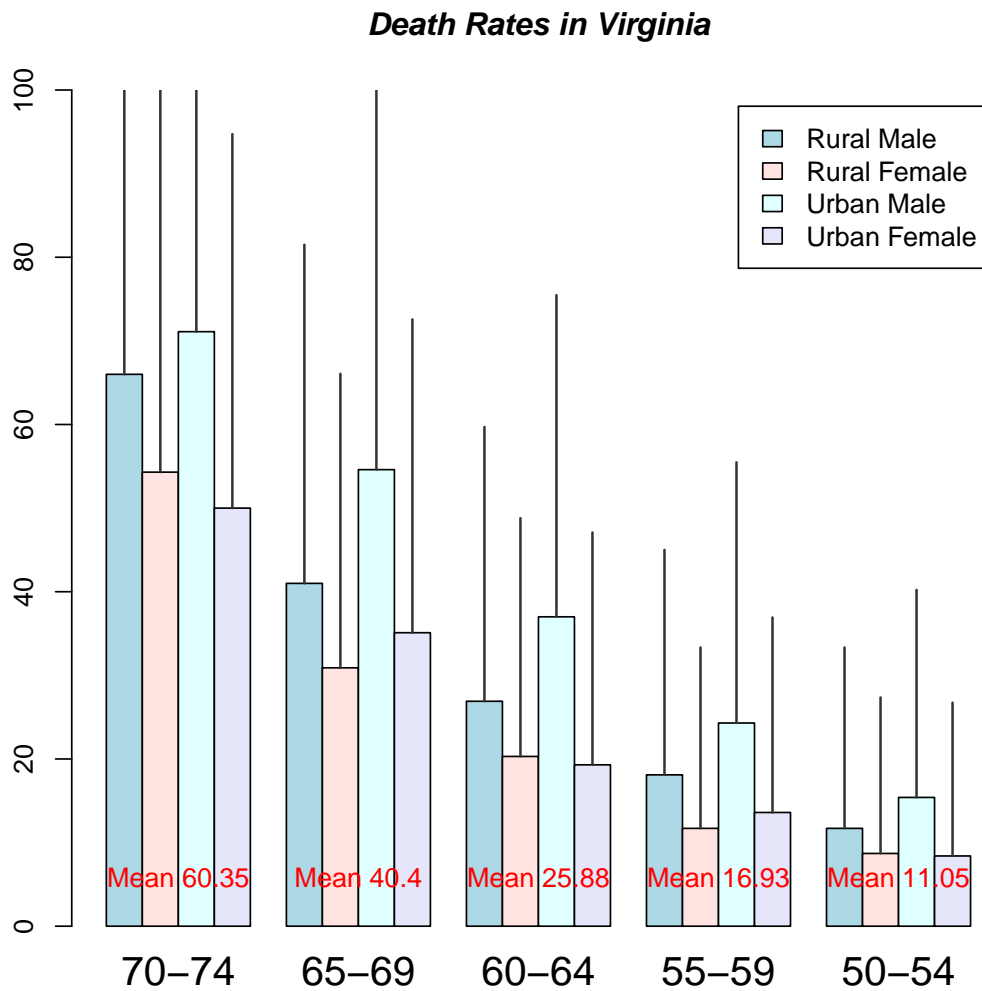
```
barplot(VADeaths, beside = TRUE,
        col = c("lightblue", "mistyrose", "lightcyan",
                "lavender", "cornsilk"),
        legend = rownames(VADeaths), ylim = c(0, 100))
title(main = "Death Rates in Virginia", font.main = 4)
```



```
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)) # corresponding matrices
```

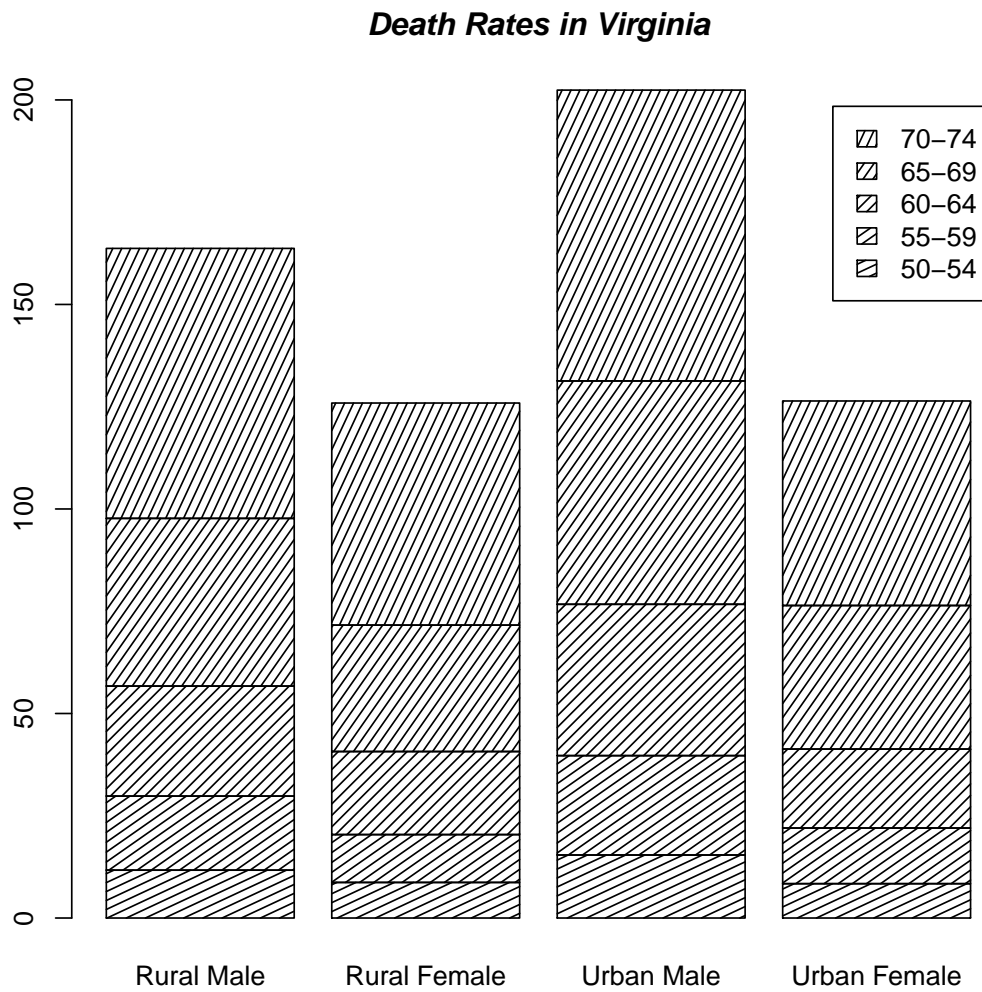
```
mtext(side = 1, at = colMeans(mp), line = -2,
      text = paste("Mean", formatC(colMeans(hh))), col = "red")
```



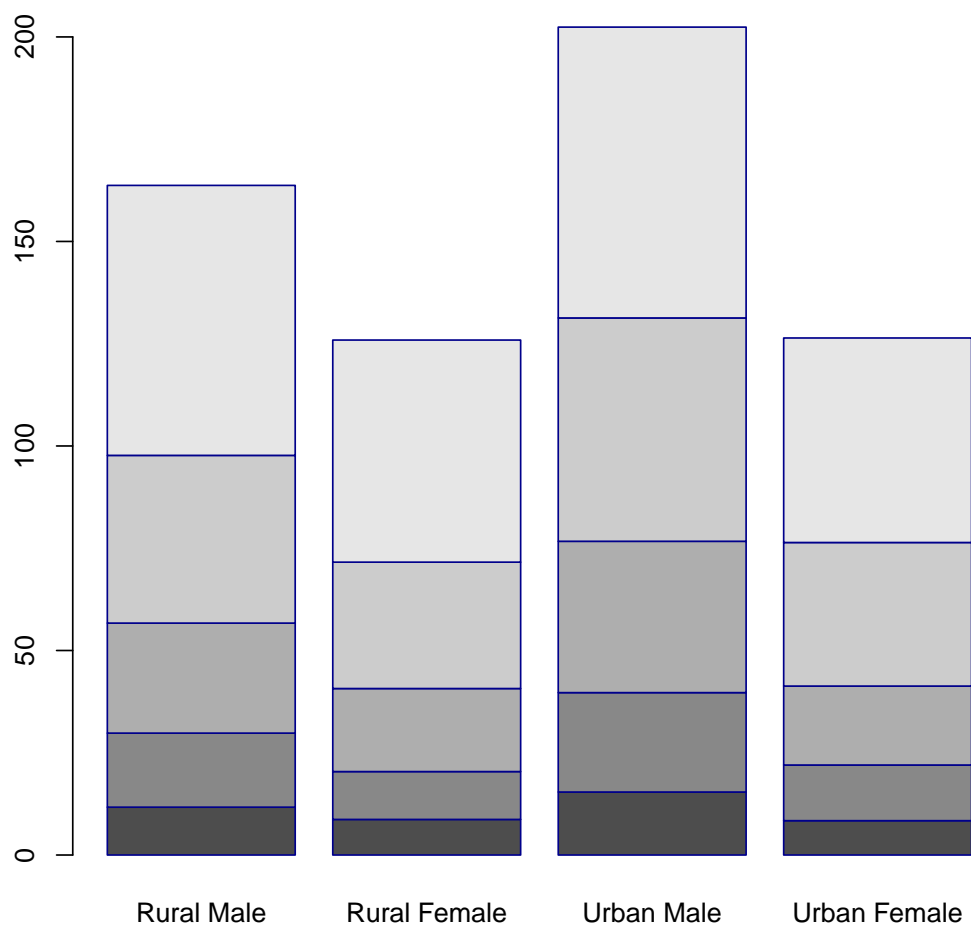
Faked upper 2\*sigma error bars

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



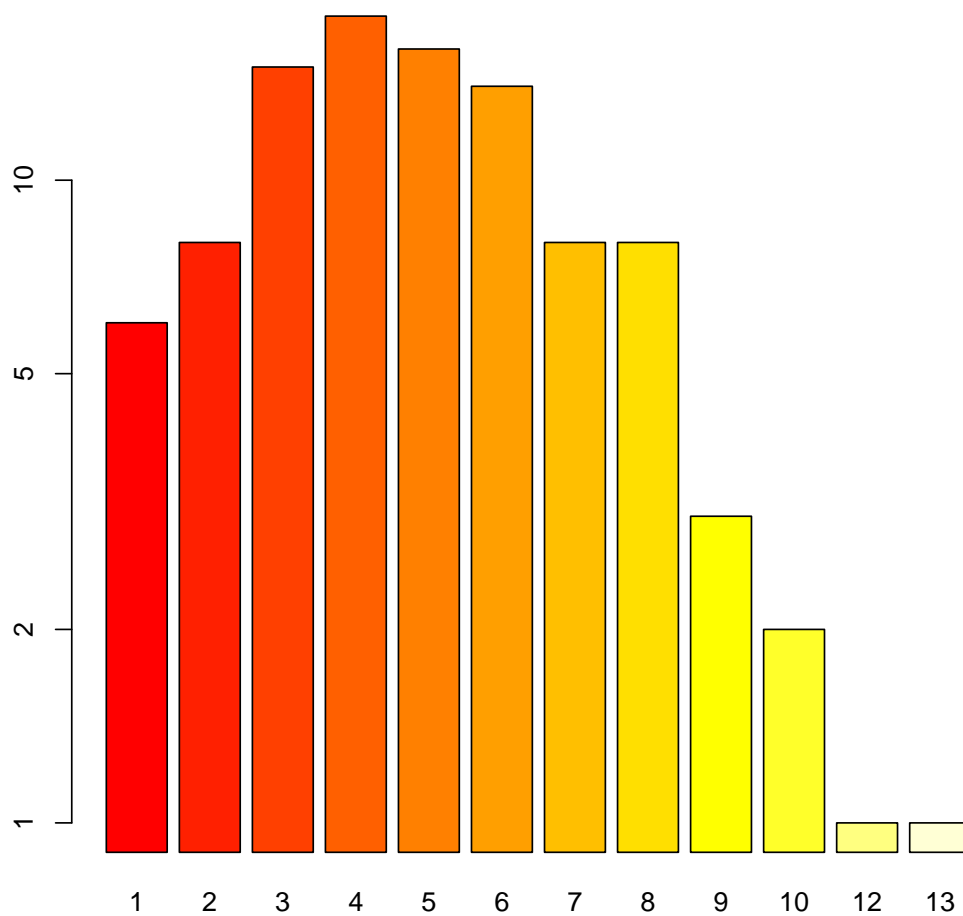


```
barplot(VADeaths, border = "dark blue")
```

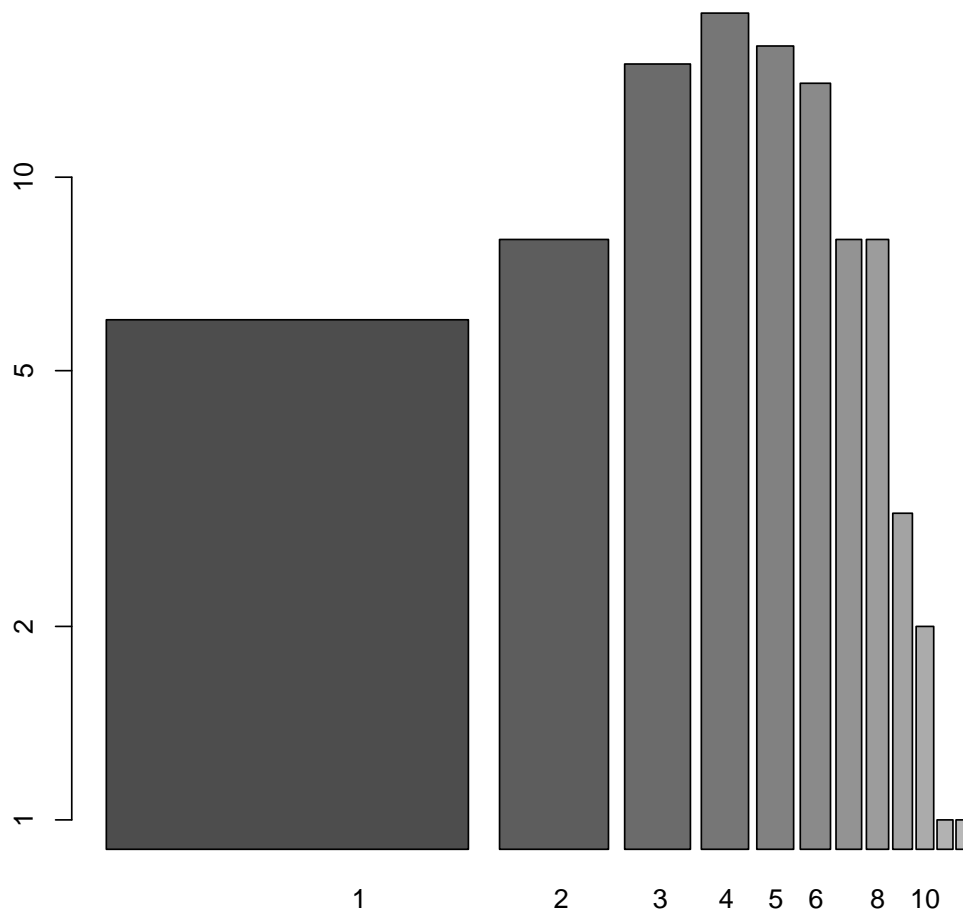


Échelles logarithmiques

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



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



## 4 hist

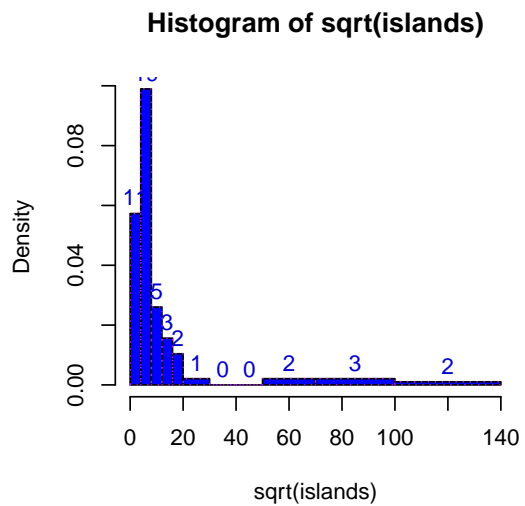
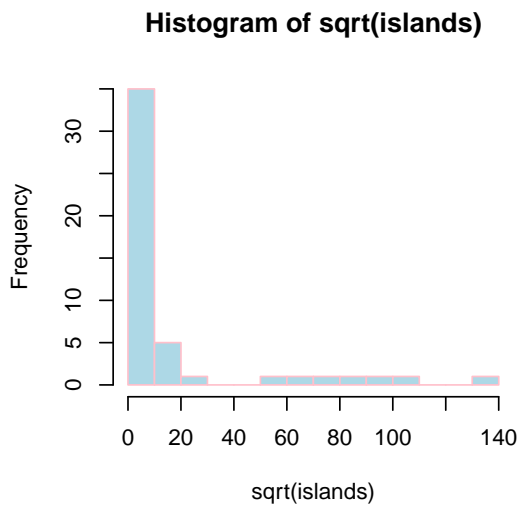
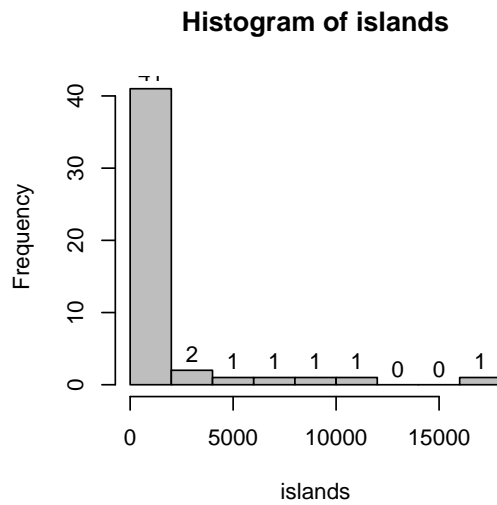
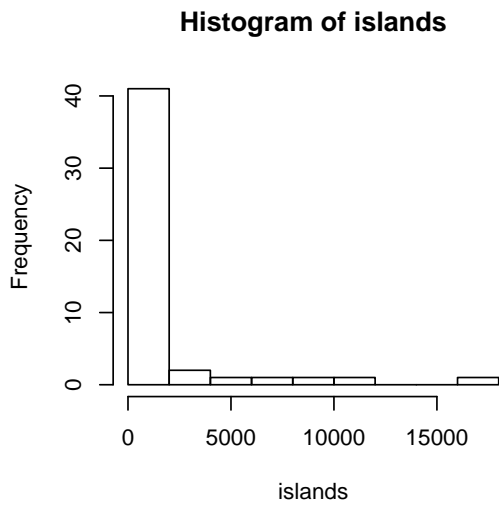
```

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")

##-- For non-equidistant breaks, counts should NOT be graphed unscaled:
r <- hist(sqrt(islands), breaks = c(4*0:5, 10*3:5, 70, 100, 140),
          col='blue1')
text(r$mids, r$density, r$counts, adj=c(.5, -.5), col='blue3')
sapply(r[2:3], sum)
sum(r$density * diff(r$breaks)) # == 1

```

```
lines(r, lty = 3, border = "purple") # -> lines.histogram(*)
```



```
par(op)
```

```
require(utils) # for str
str(hist(islands, breaks=12, plot= FALSE)) #-> 10 (~= 12) breaks

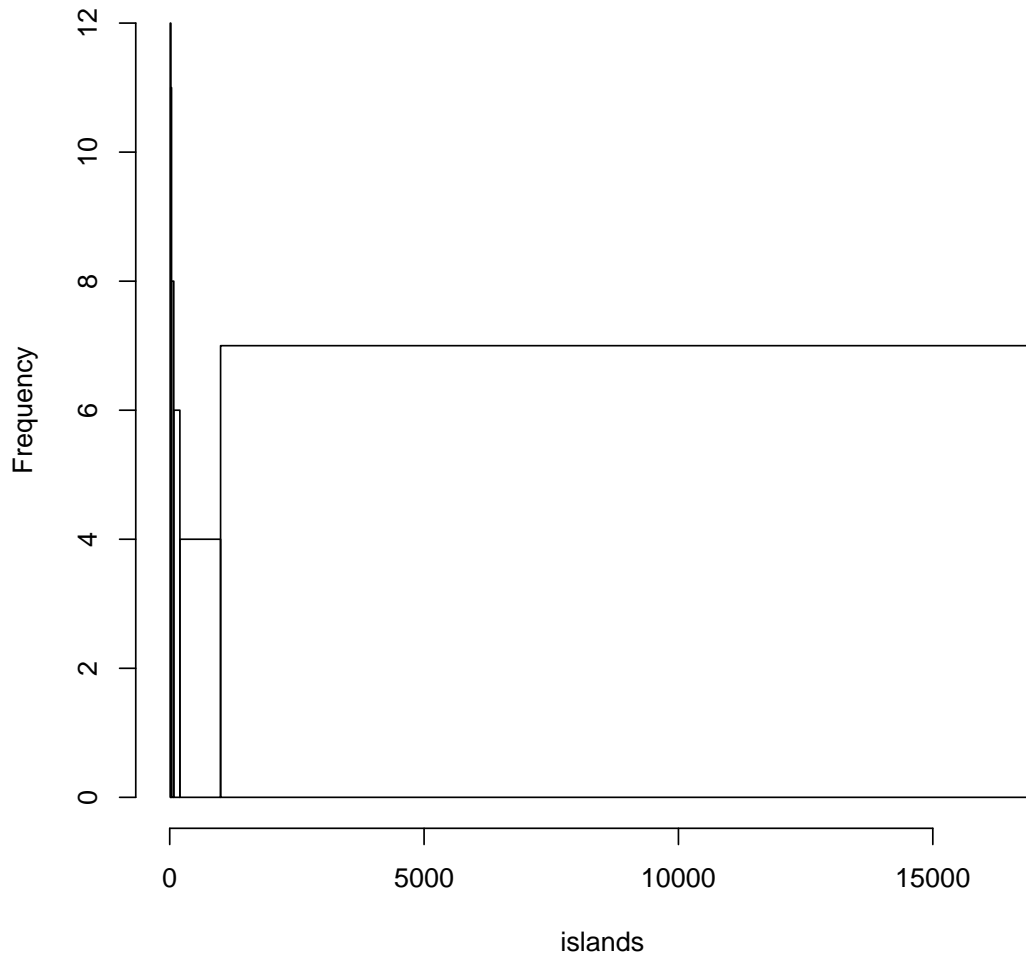
## List of 6
## $ breaks : num [1:10] 0 2000 4000 6000 8000 10000 12000 14000 16000 18000
## $ counts : int [1:9] 41 2 1 1 1 1 0 0 1
## $ density : num [1:9] 4.27e-04 2.08e-05 1.04e-05 1.04e-05 1.04e-05 ...
## $ mids : num [1:9] 1000 3000 5000 7000 9000 11000 13000 15000 17000
```

```
## $ 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 6
## $ breaks : num [1:7] 12 20 36 80 200 1000 17000
## $ counts : int [1:6] 12 11 8 6 4 7
## $ 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"
```

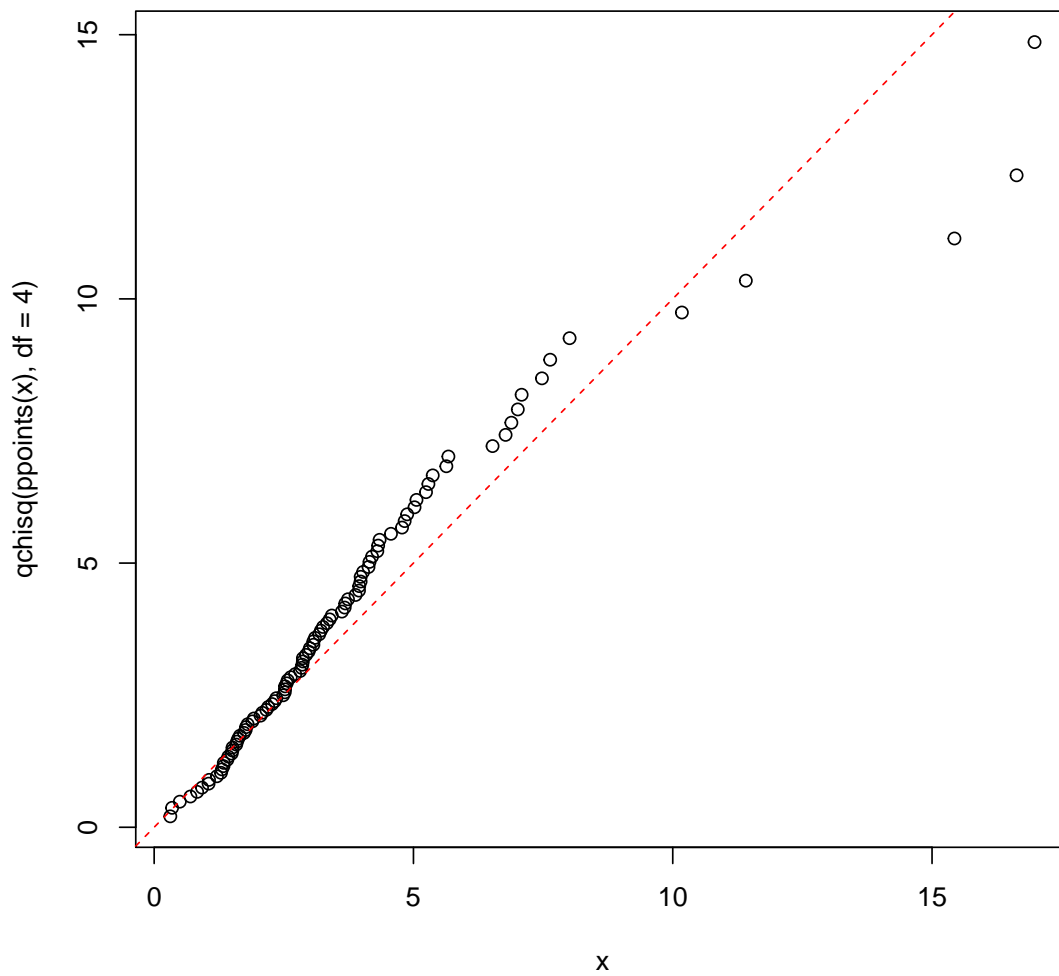
```
hist(islands, breaks=c(12,20,36,80,200,1000,17000), freq = TRUE,
      main = "WRONG histogram") # and warning
## Warning in plot.histogram(r, freq = freq1, col = col, border =
border, angle = angle, : the AREAS in the plot are wrong - rather use
'freq = FALSE'
```

**WRONG histogram**

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

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

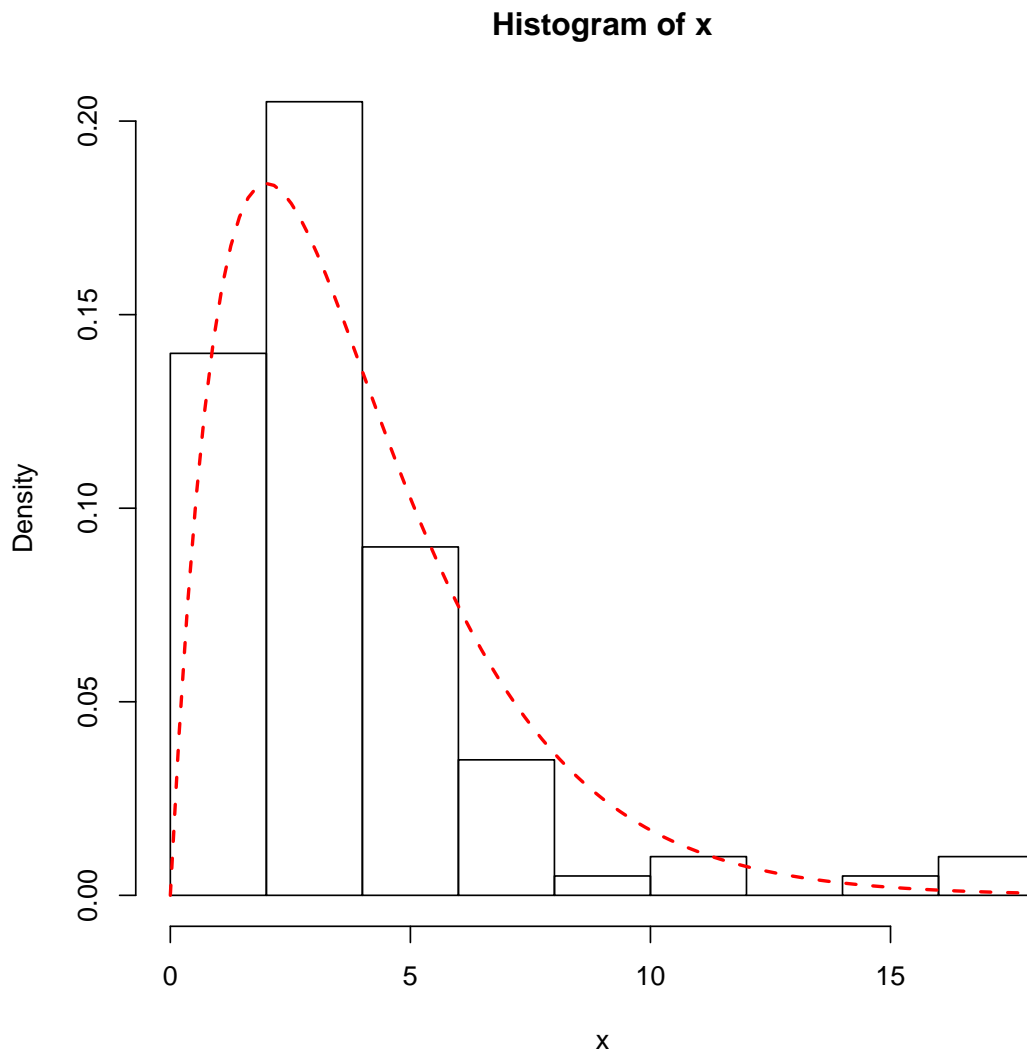
```
qqplot(x, qchisq(ppoints(x), df = 4)); abline(0,1, col = 2, lty = 2)
```



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)
```

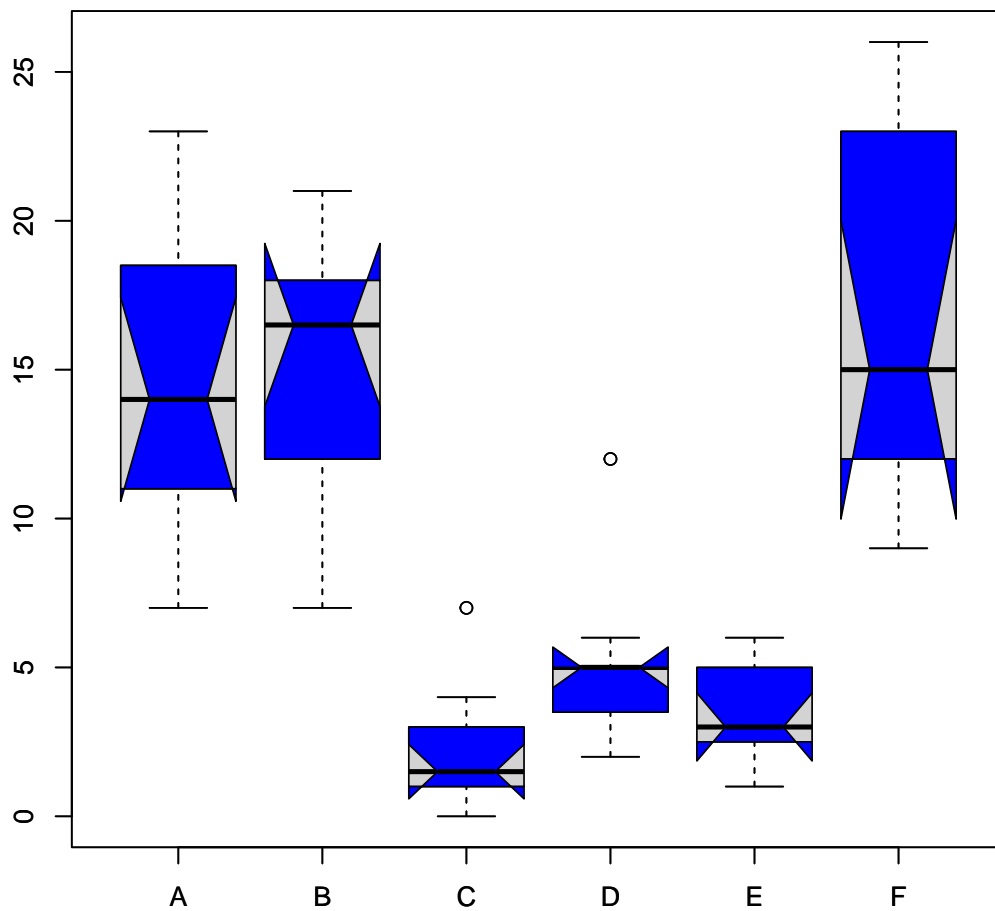




## 5 Boîtes à moustaches

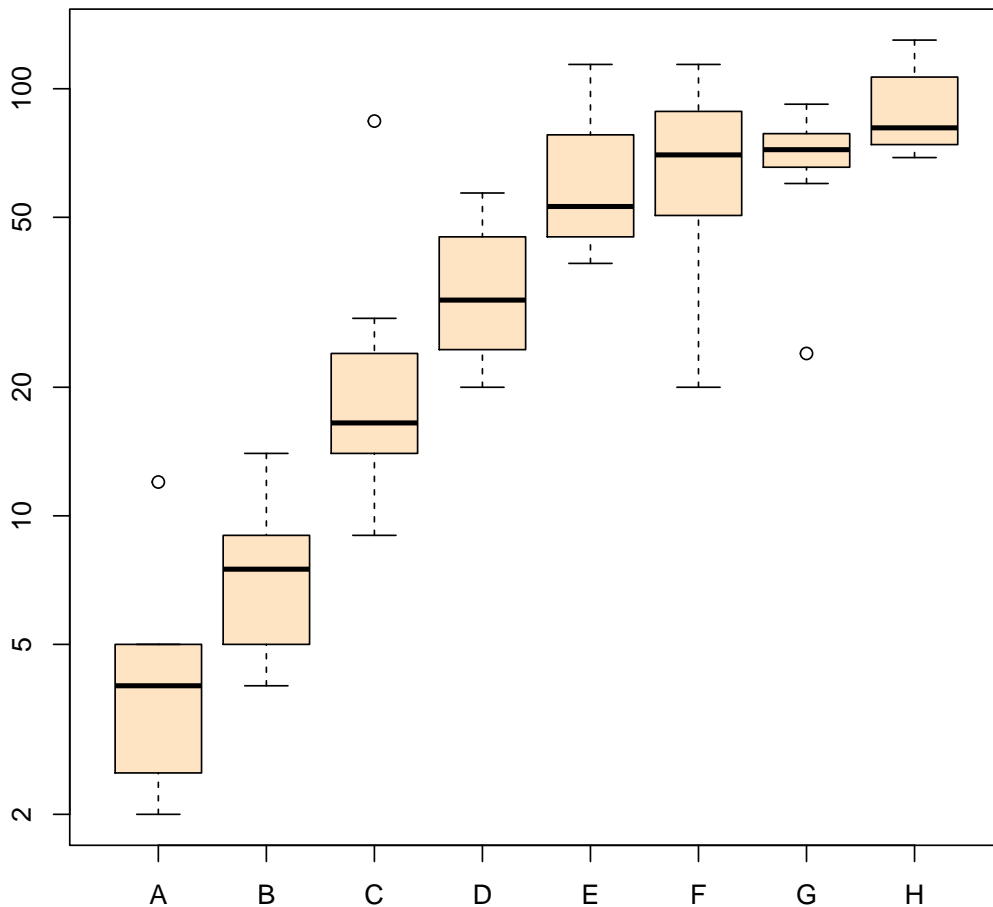
### 5.1 boxplot d'une formule

```
boxplot(count ~ spray, data = InsectSprays, col = "lightgray")
boxplot(count ~ spray, data = InsectSprays,
        notch = TRUE, add=TRUE, col = "blue")
## Warning in bxp(structure(list(stats = structure(c(7, 11, 14, 18.5,
23, 7, : some notches went outside hinges ('box'): maybe set
notch=FALSE
```



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

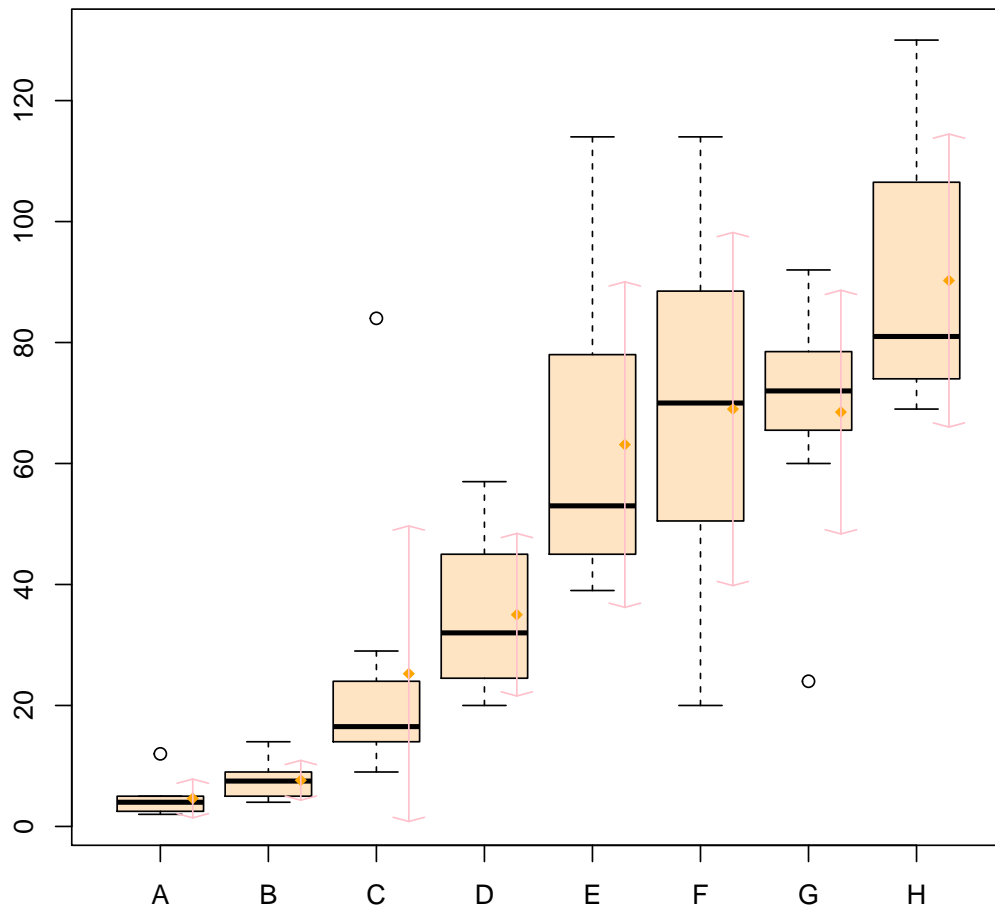
```
boxplot(decrease ~ treatment, data = OrchardSprays,  
        log = "y", col = "bisque")
```



```
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 = .1)
```

### Comparing boxplot(s) and non-robust mean $\pm$ SD

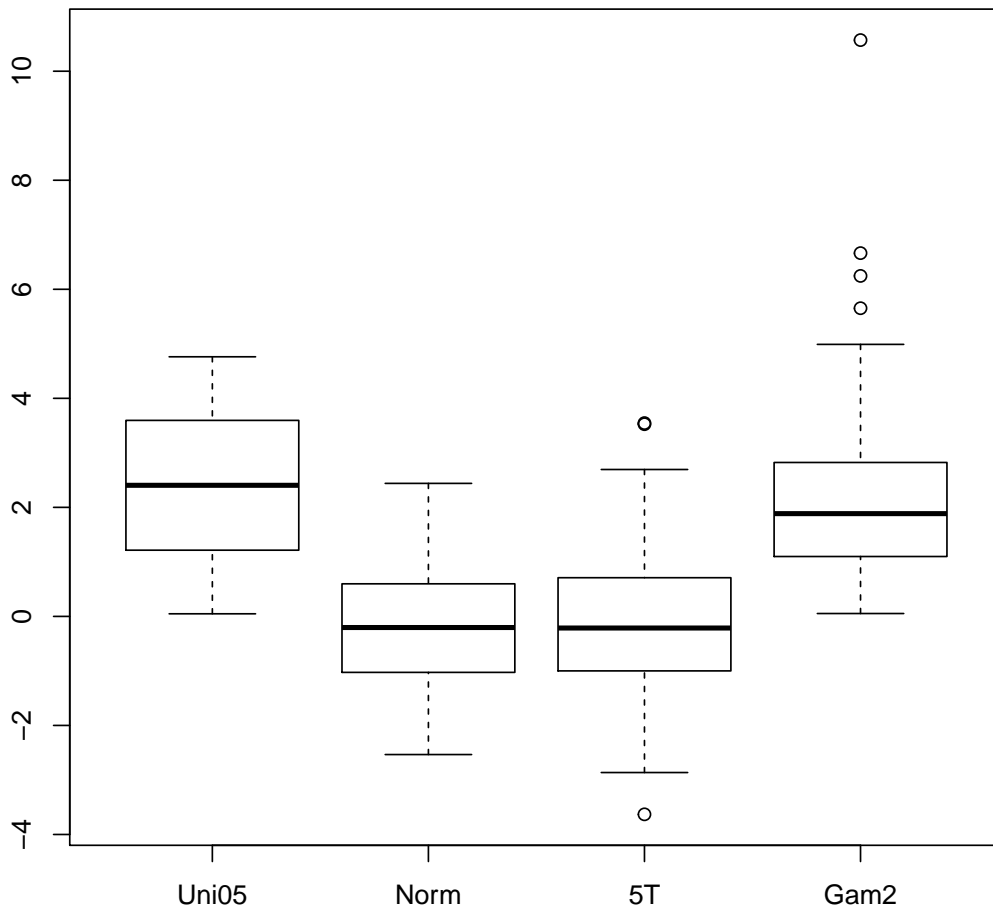


## 5.2 boxplot d'une matrice

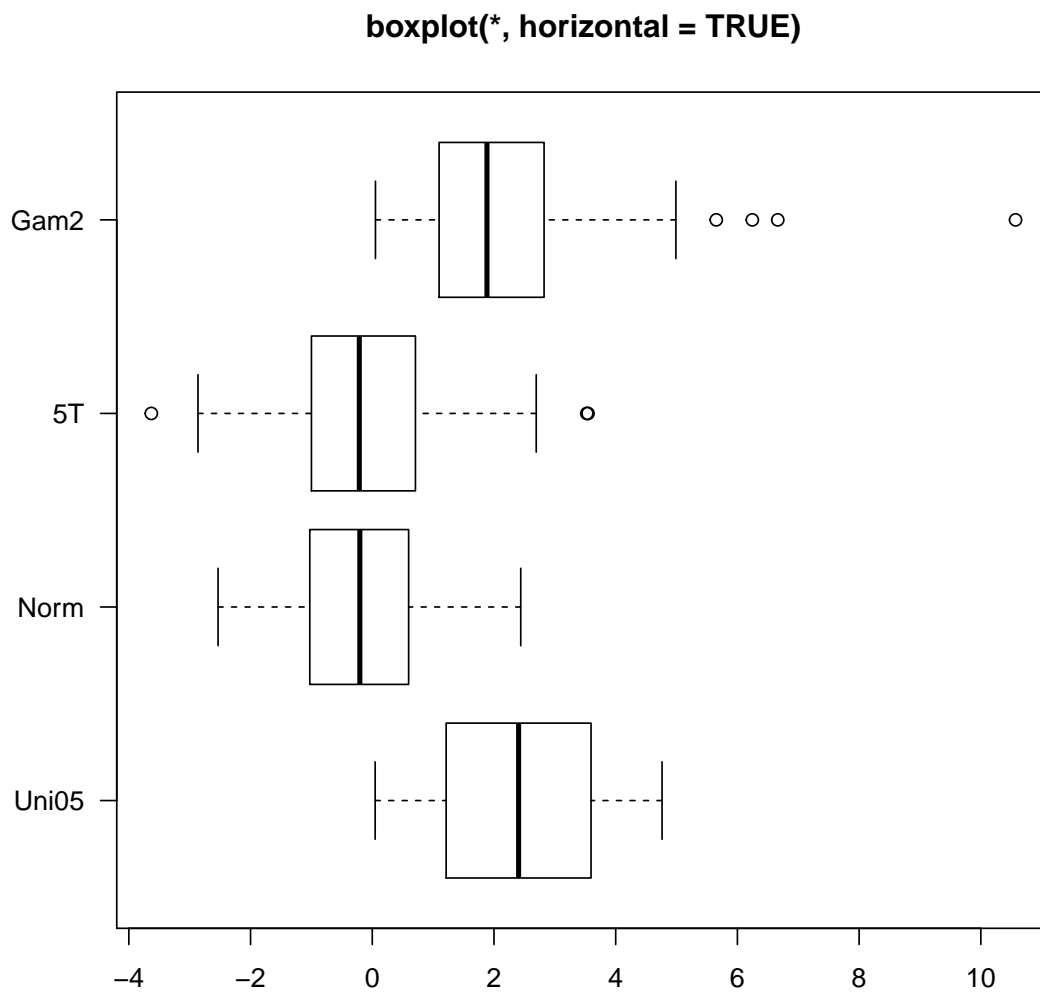
```
mat <- cbind(Uni05 = (1:100)/21, Norm = rnorm(100),
             `5T` = rt(100, df = 5), Gam2 = rgamma(100, shape = 2))
```

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

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



```
par(las=1)# all axis labels horizontal
boxplot(as.data.frame(mat), main = "boxplot(*, horizontal = TRUE)",
        horizontal = TRUE)
```

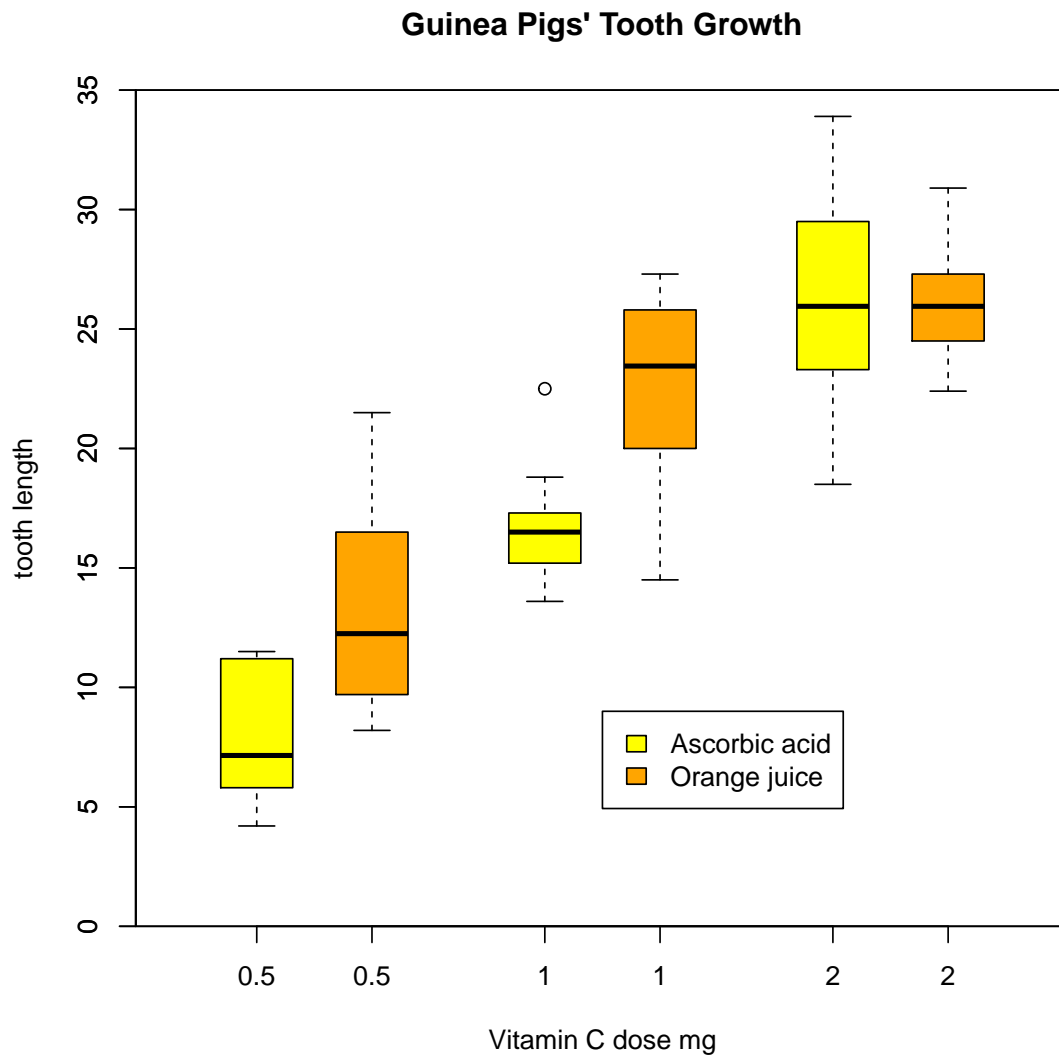


Using 'at = ' and adding boxplots – example idea by Roger Bivand :

```

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"))

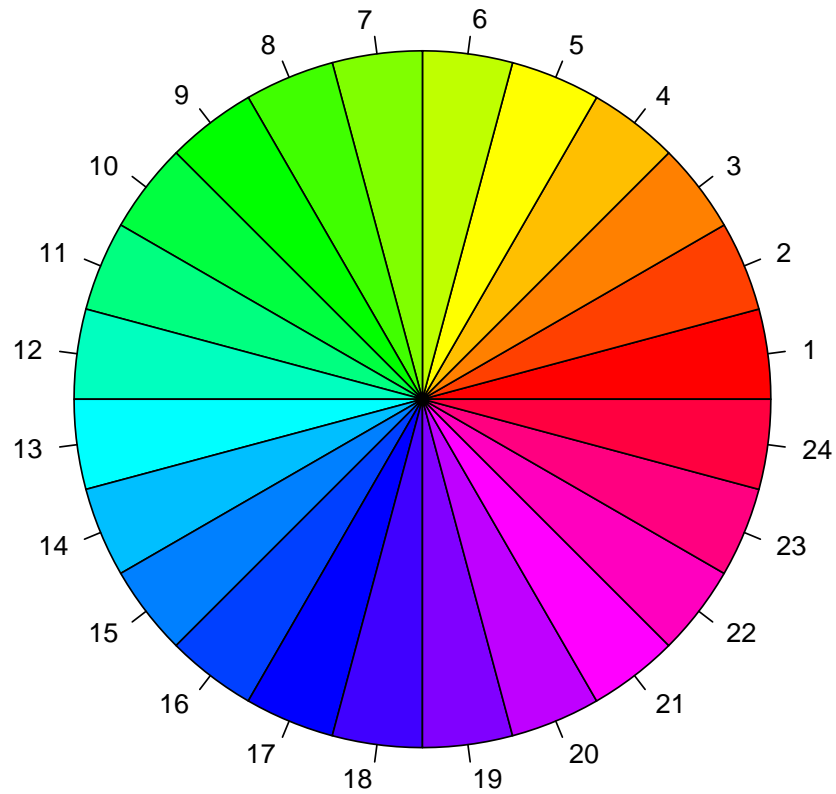
```



## 6 pie

```
require(grDevices)
```

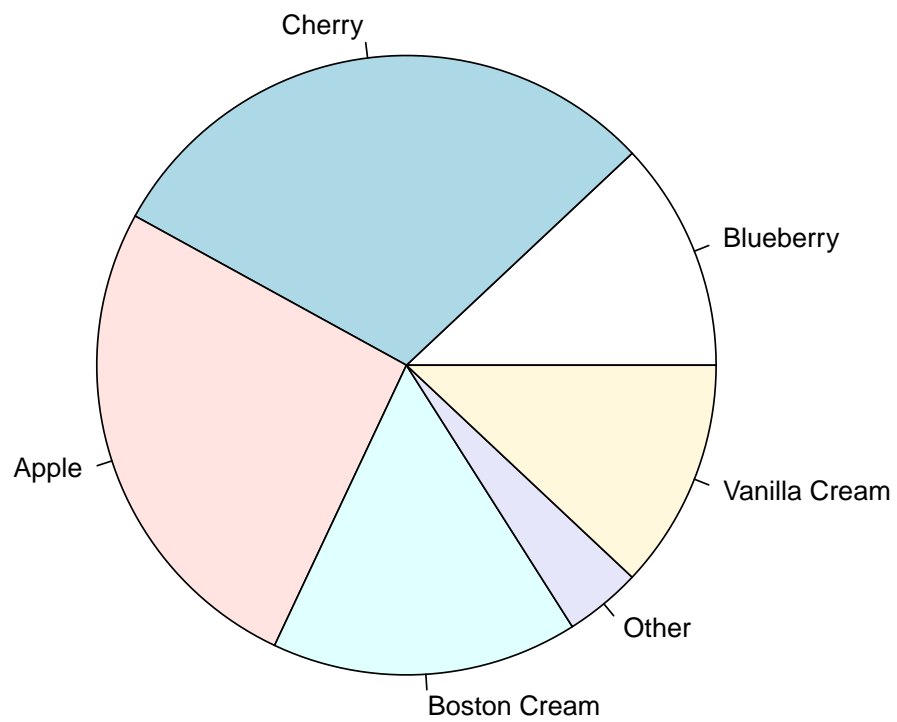
```
pie(rep(1, 24), col = rainbow(24), radius = 0.9)
```



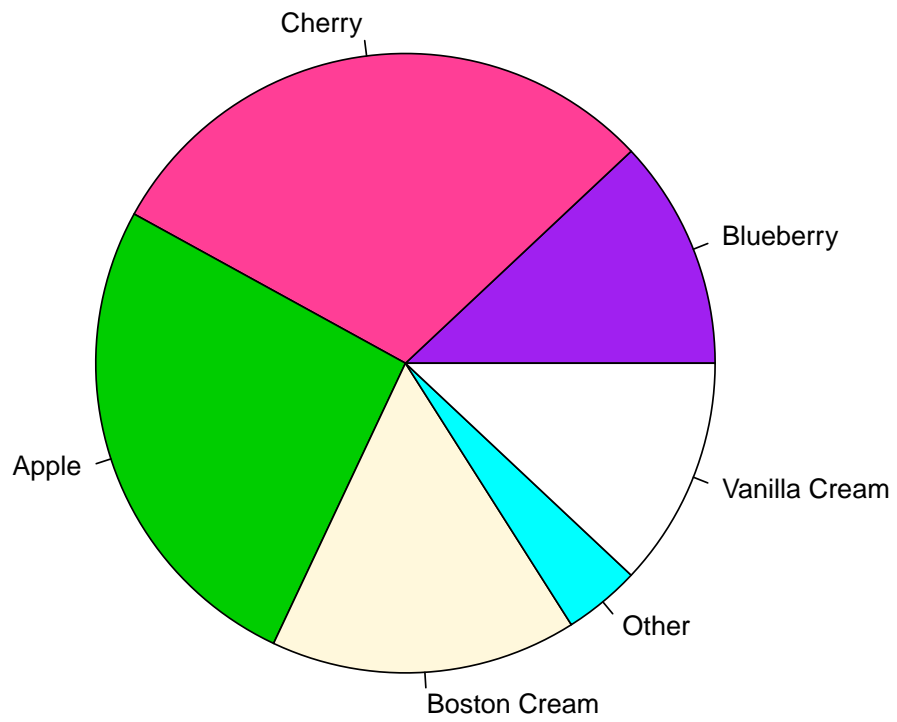
```
pie.sales <- c(0.12, 0.3, 0.26, 0.16, 0.04, 0.12)
names(pie.sales) <- c("Blueberry", "Cherry",
  "Apple", "Boston Cream", "Other", "Vanilla Cream")
```

```
pie(pie.sales) # default colours
```

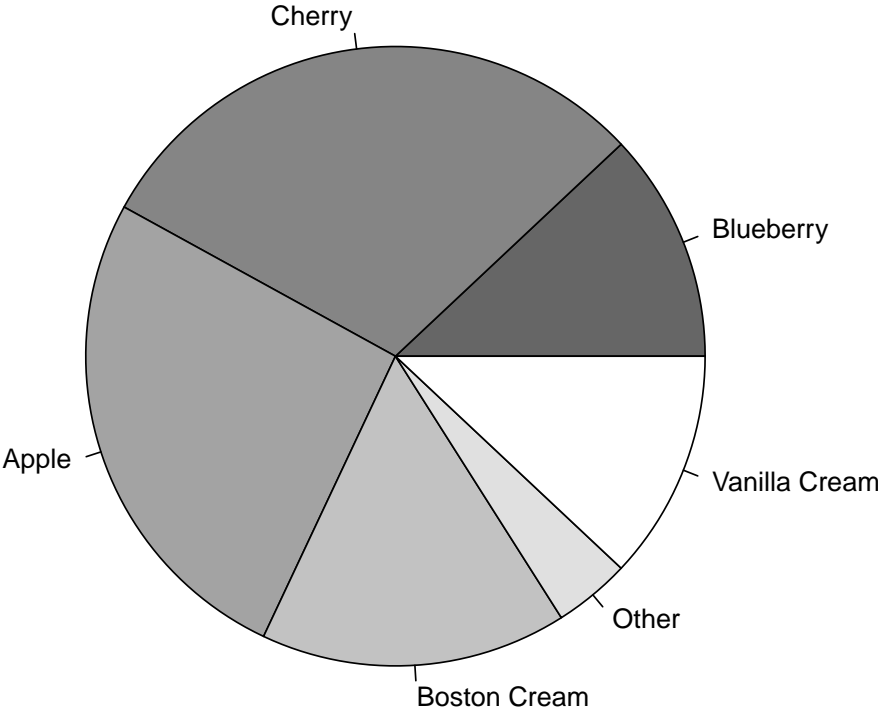




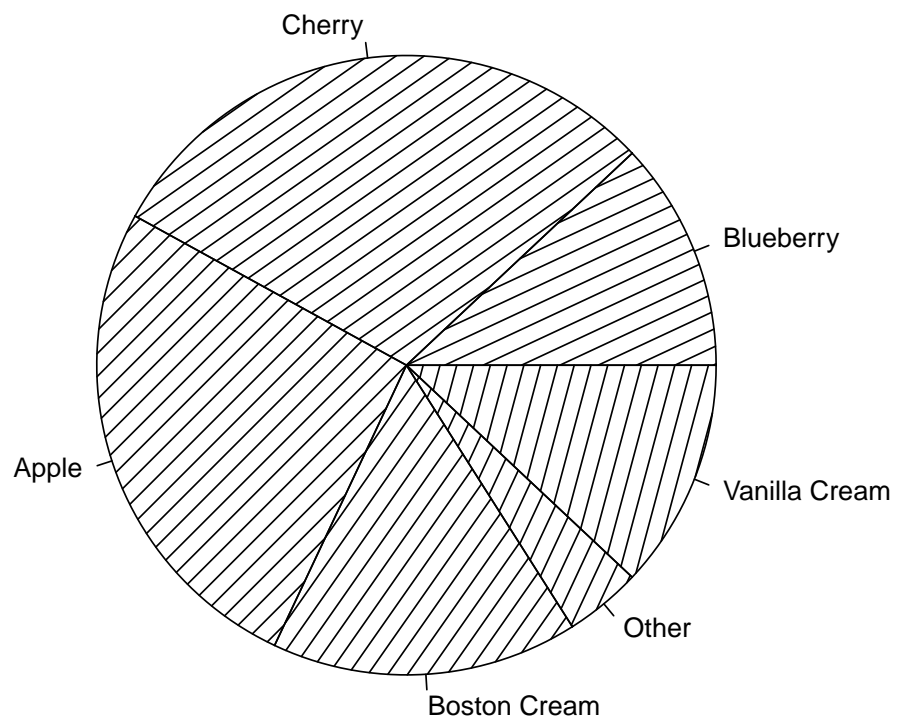
```
pie(pie.sales,  
    col = c("purple", "violetred1", "green3", "cornsilk", "cyan", "white"))
```



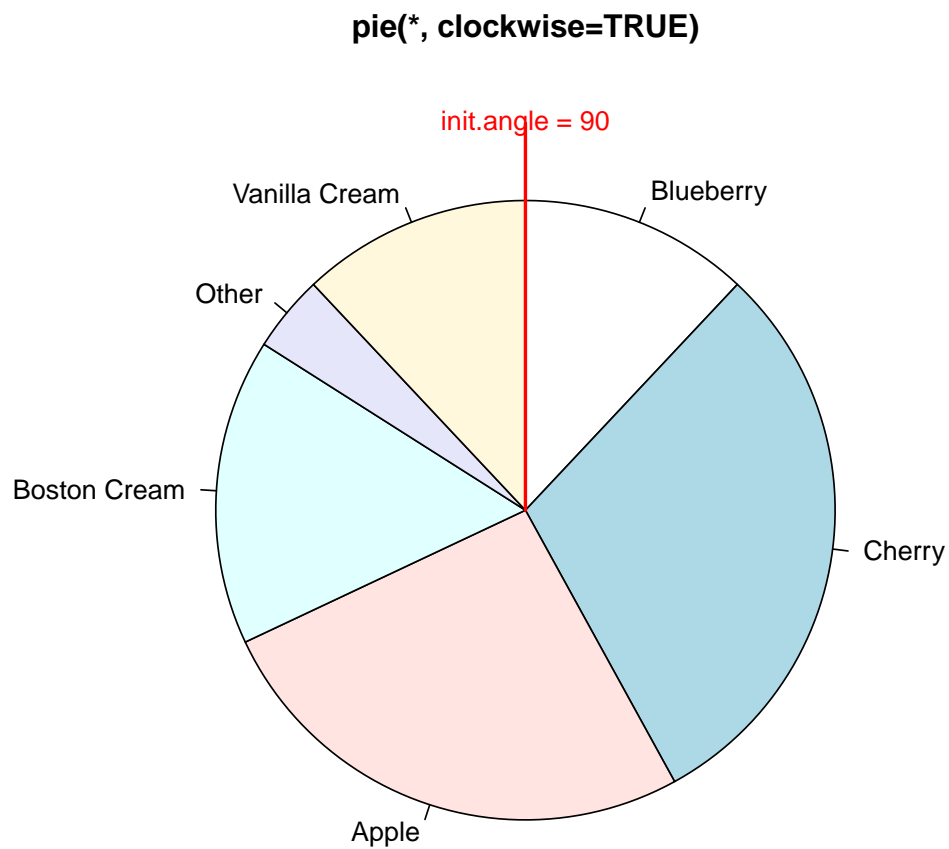
```
pie(pie.sales, col = gray(seq(0.4,1.0,length=6)))
```



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

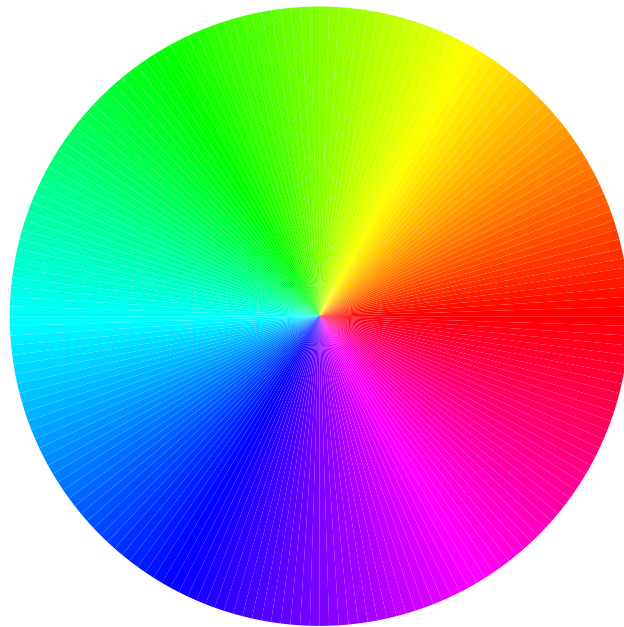


```
pie(pie.sales, clockwise=TRUE, main="pie(*, clockwise=TRUE)")
segments(0,0, 0,1, col= "red", lwd = 2)
text(0,1, "init.angle = 90", col= "red")
```



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

## Rainbow



## 7 Tableaux de contingence

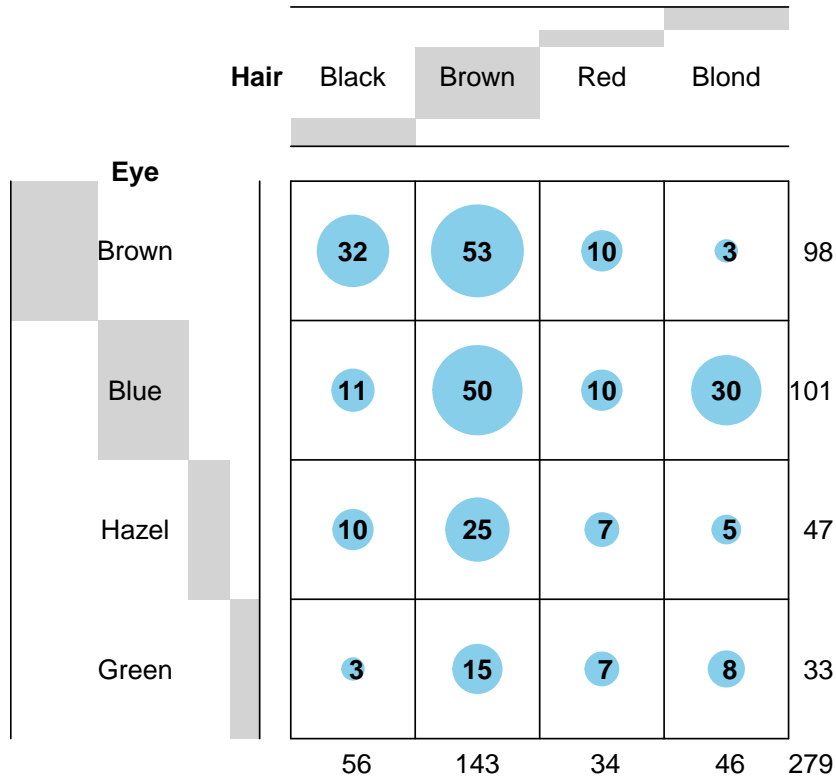
### 7.1 balloonplot

```
library(gdata)
## gdata: read.xls support for 'XLS' (Excel 97-2004) files ENABLED.
##
## gdata: read.xls support for 'XLSX' (Excel 2007+) files ENABLED.
##
## Attaching package: 'gdata'
## The following object is masked from 'package:stats':
##
##   nobs
```

```
## The following object is masked from 'package:utils':  
##  
##   object.size  
## The following object is masked from 'package:base':  
##  
##   startsWith  
  
library(gtools)  
library(gplots)  
  
##  
## Attaching package: 'gplots'  
## The following object is masked from 'package:stats':  
##  
##   lowess
```

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

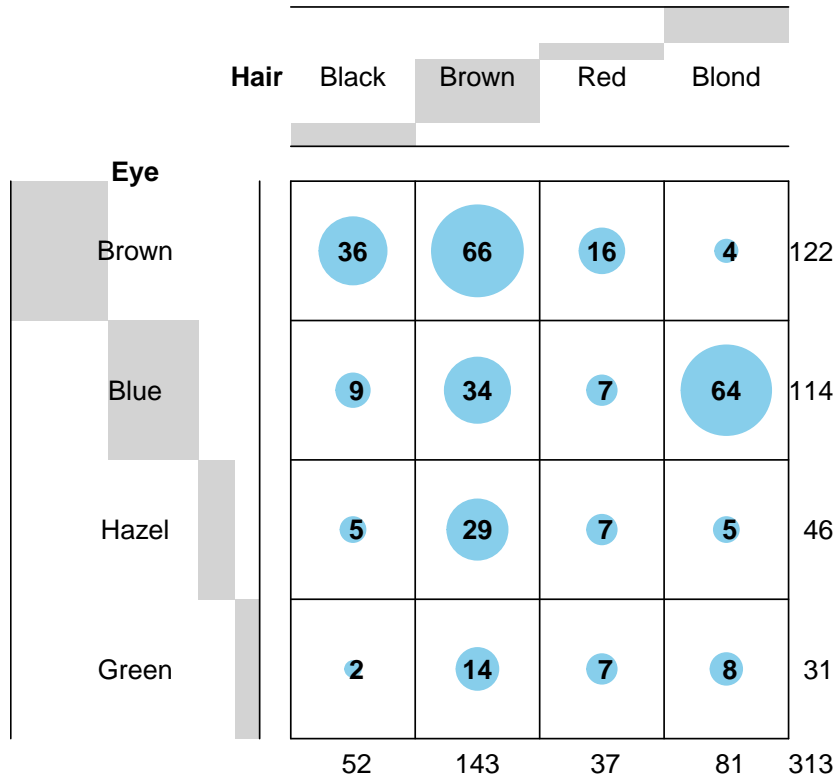
**Balloon Plot for x by y.  
Area is proportional to Freq.**



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



**Balloon Plot for x by y.**  
Area is proportional to Freq.

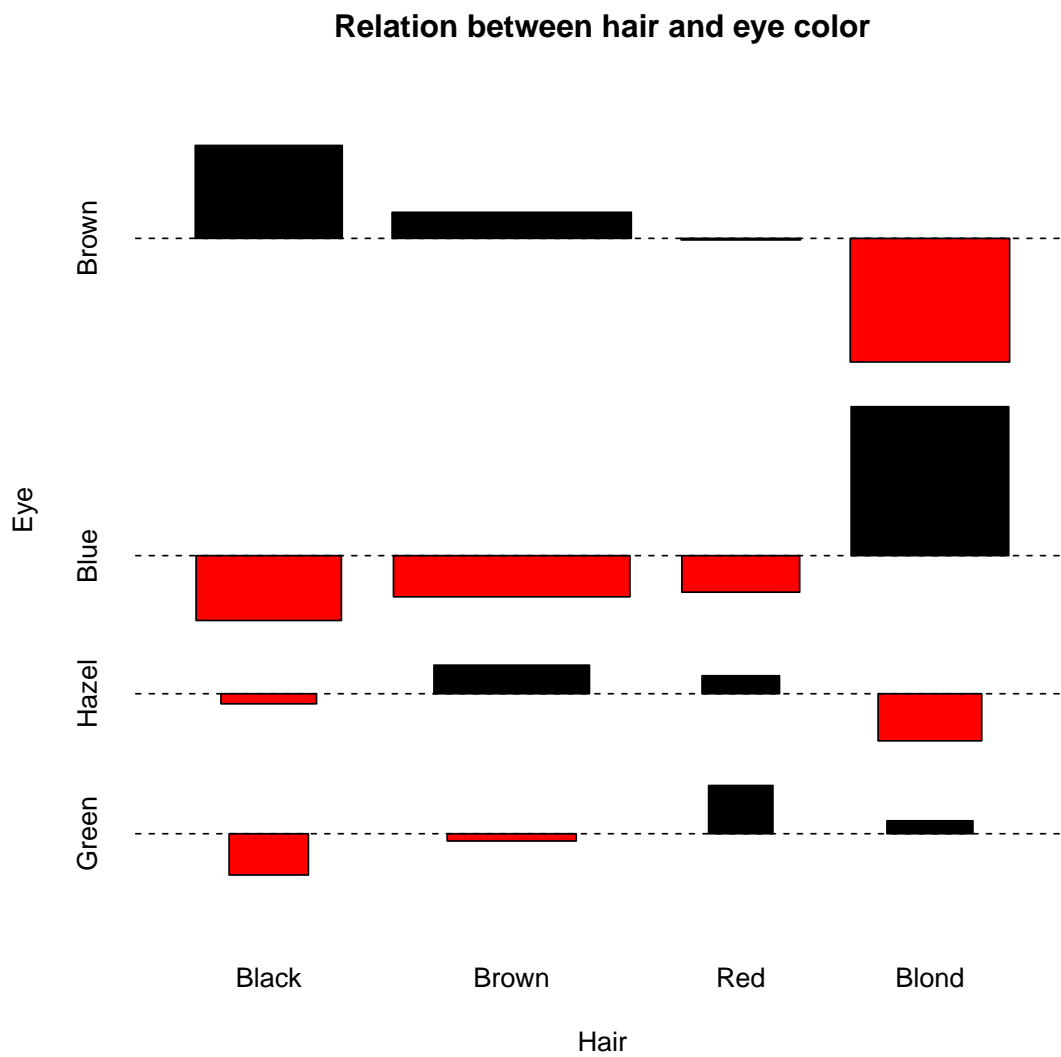


## 7.2 assocplot

## Aggregate over sex :

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

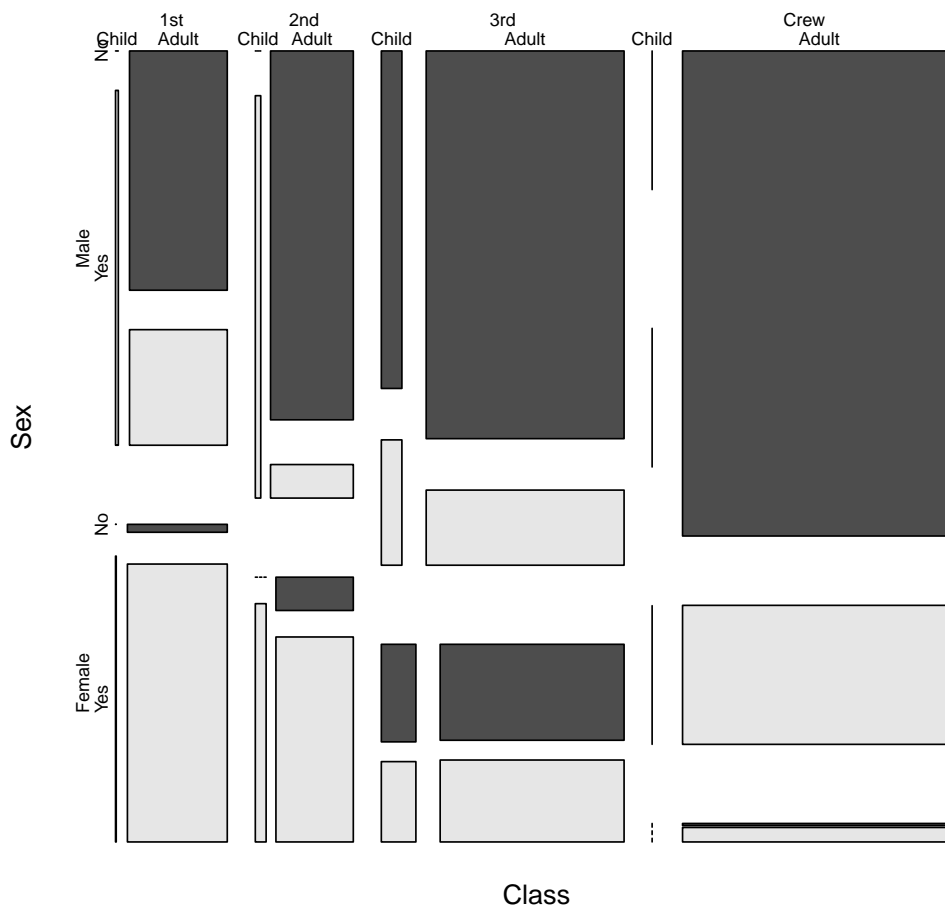
```
assocplot(x, main = "Relation between hair and eye color")
```



### 7.3 mosaicplot

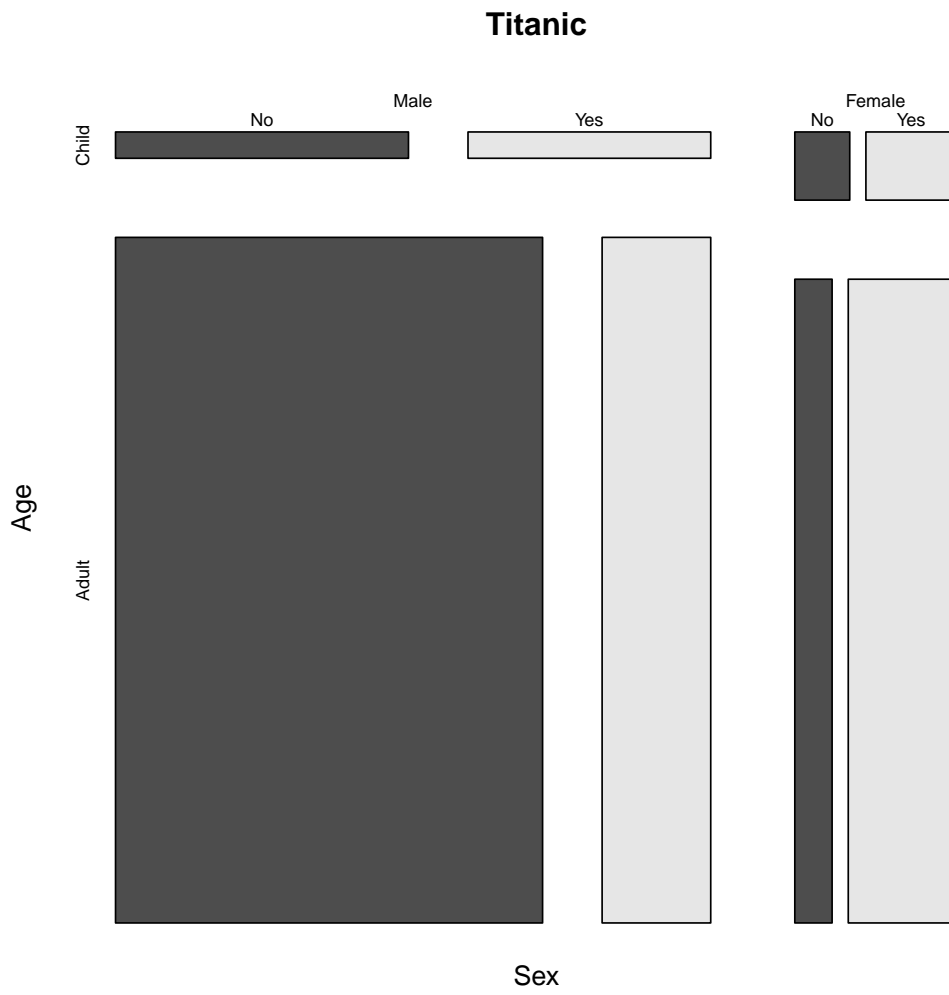
```
mosaicplot(Titanic, main = "Survival on the Titanic", color = TRUE)
```

### Survival on the Titanic

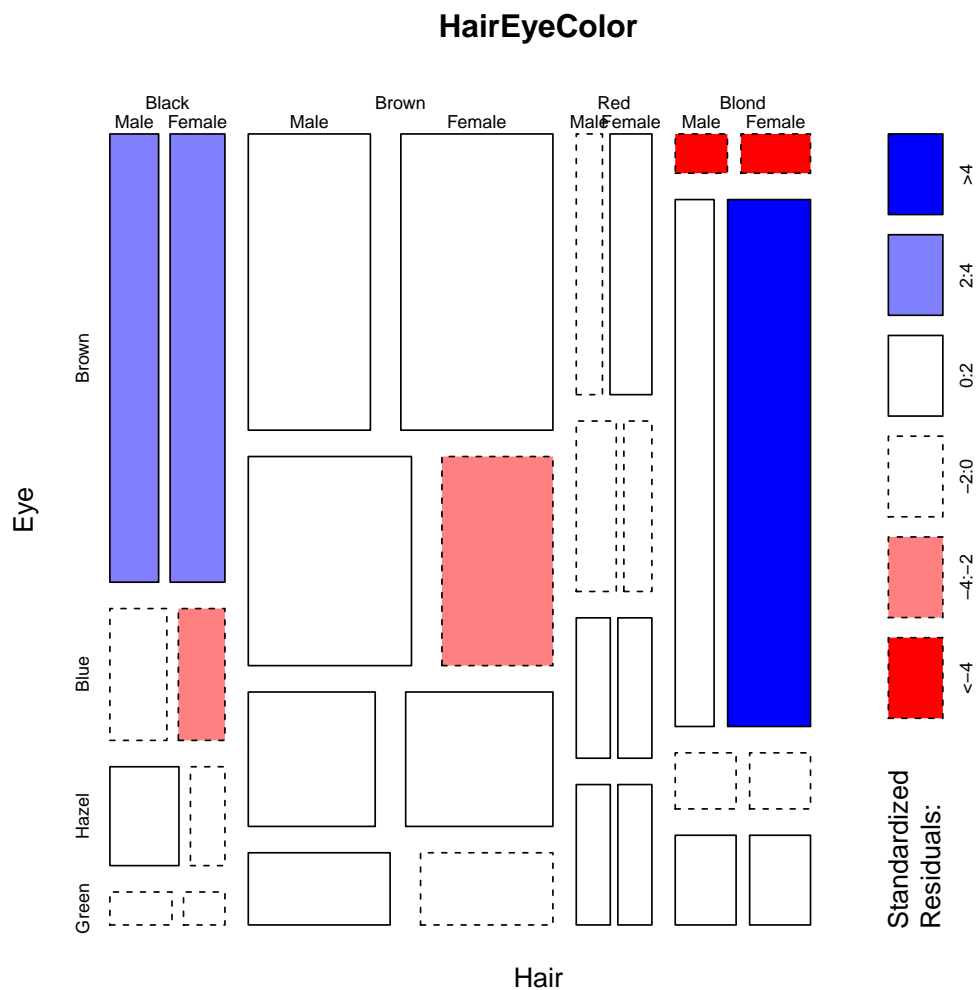


## Formula interface for tabulated data :

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



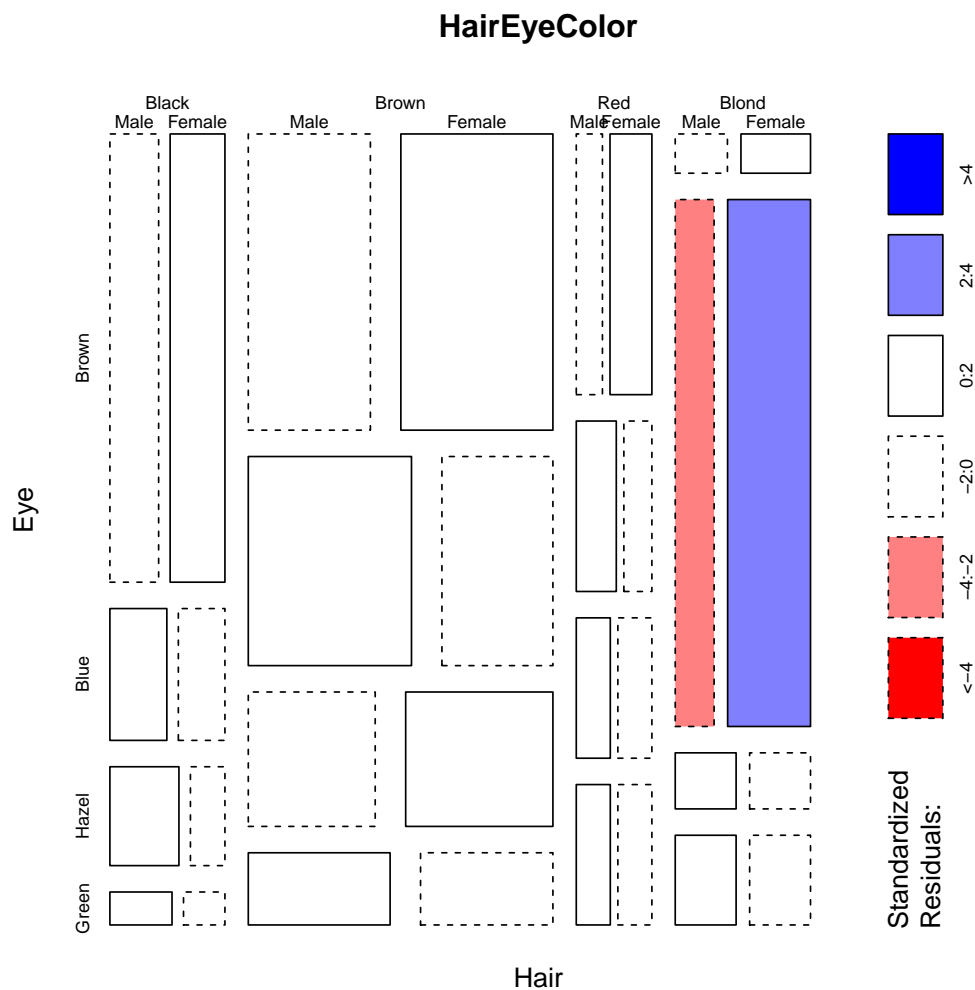
```
mosaicplot(HairEyeColor, shade = TRUE)
```



## 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))
```

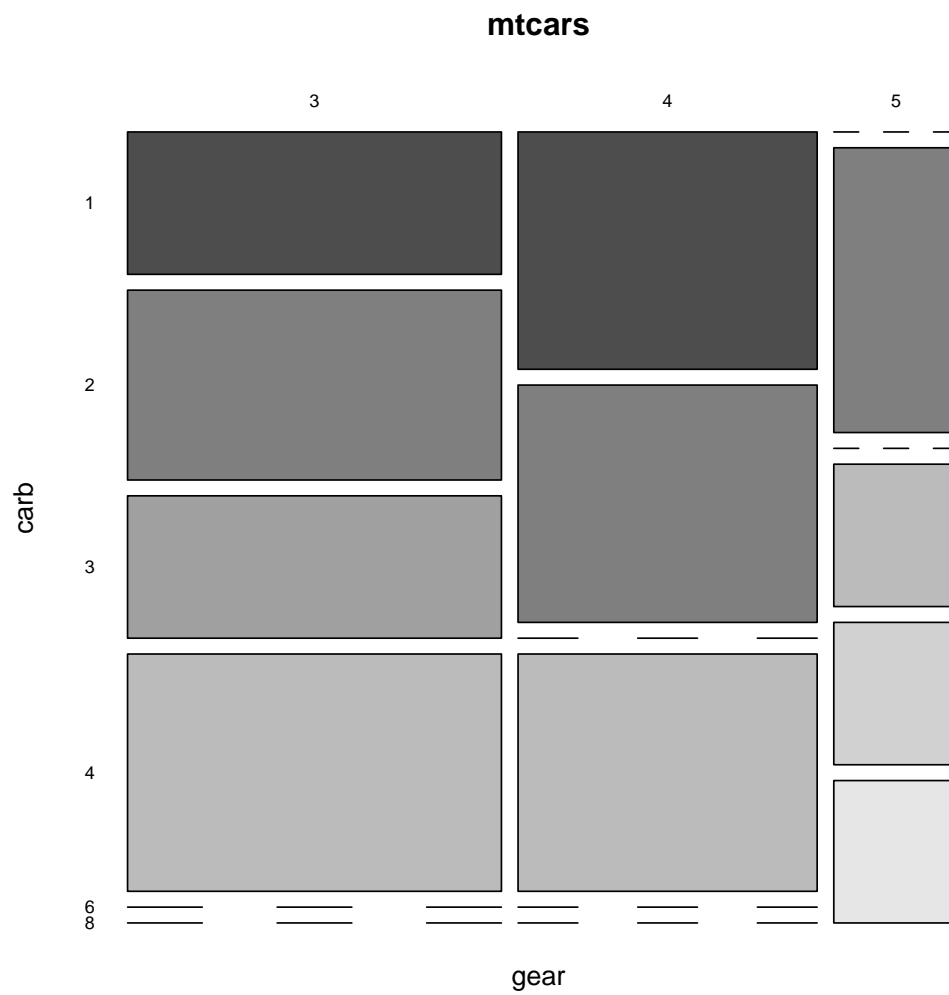


## Model of joint independence of sex from hair and eye color. Males ## are underrepresented among people with brown hair and eyes, and are ## overrepresented among people with brown hair and blue eyes. ## The corresponding model is :

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

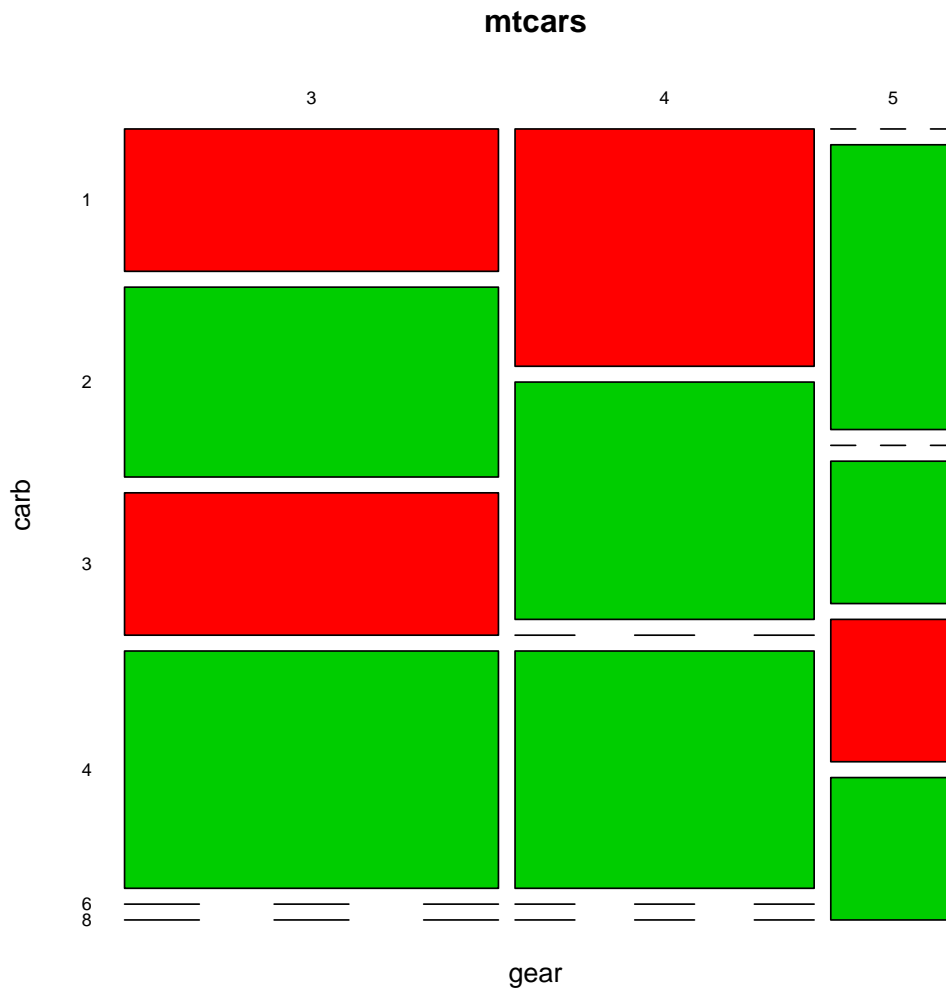
## Formula interface for raw data : visualize cross-tabulation of numbers ## of gears and carburettors in Motor Trend car data.

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



# color recycling

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



## 7.4 splineplot

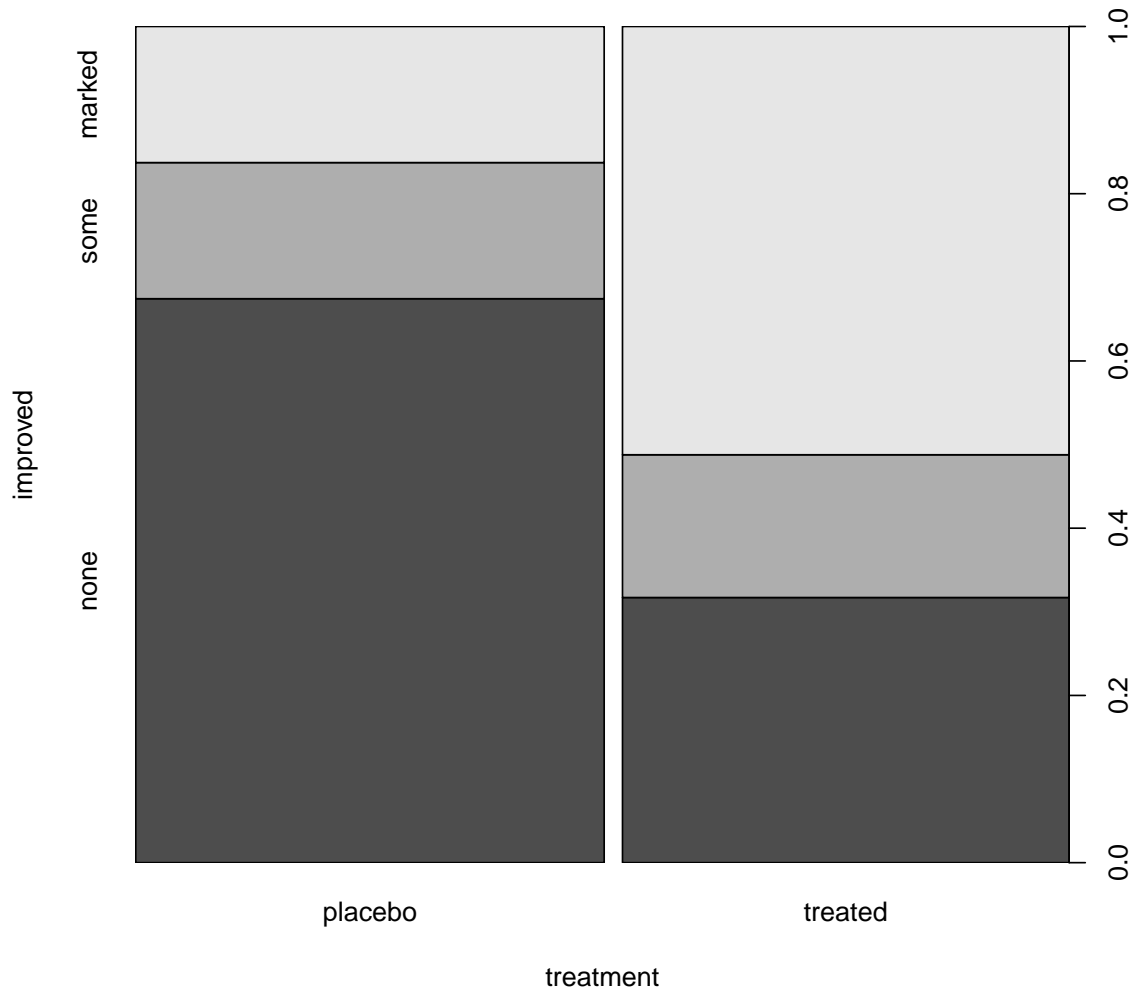
## treatment and improvement of patients with rheumatoid arthritis

```
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)

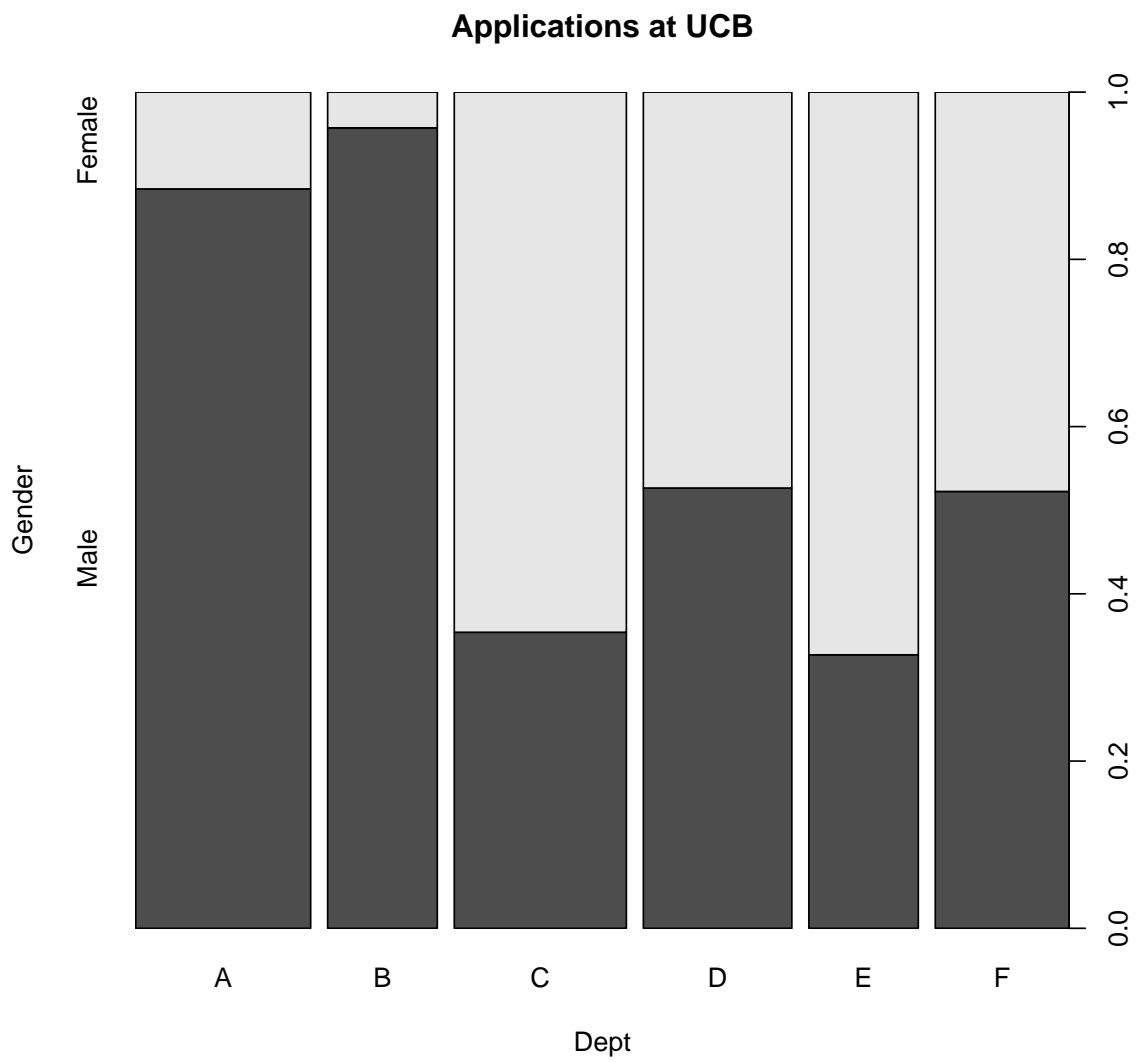


```
(spineplot(improved ~ treatment))
```

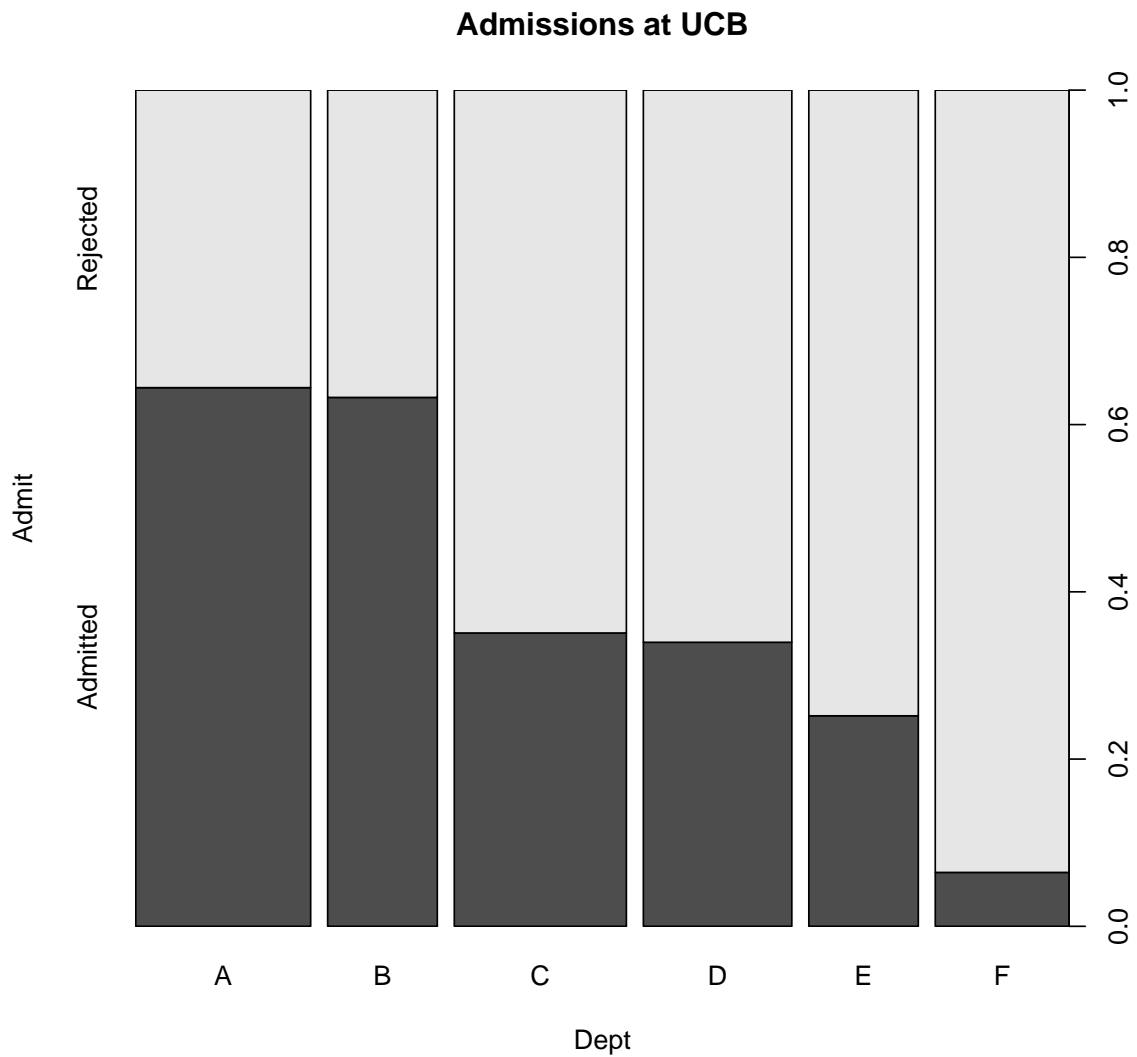


## applications and admissions by department at UC Berkeley ## (two-way tables)

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



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

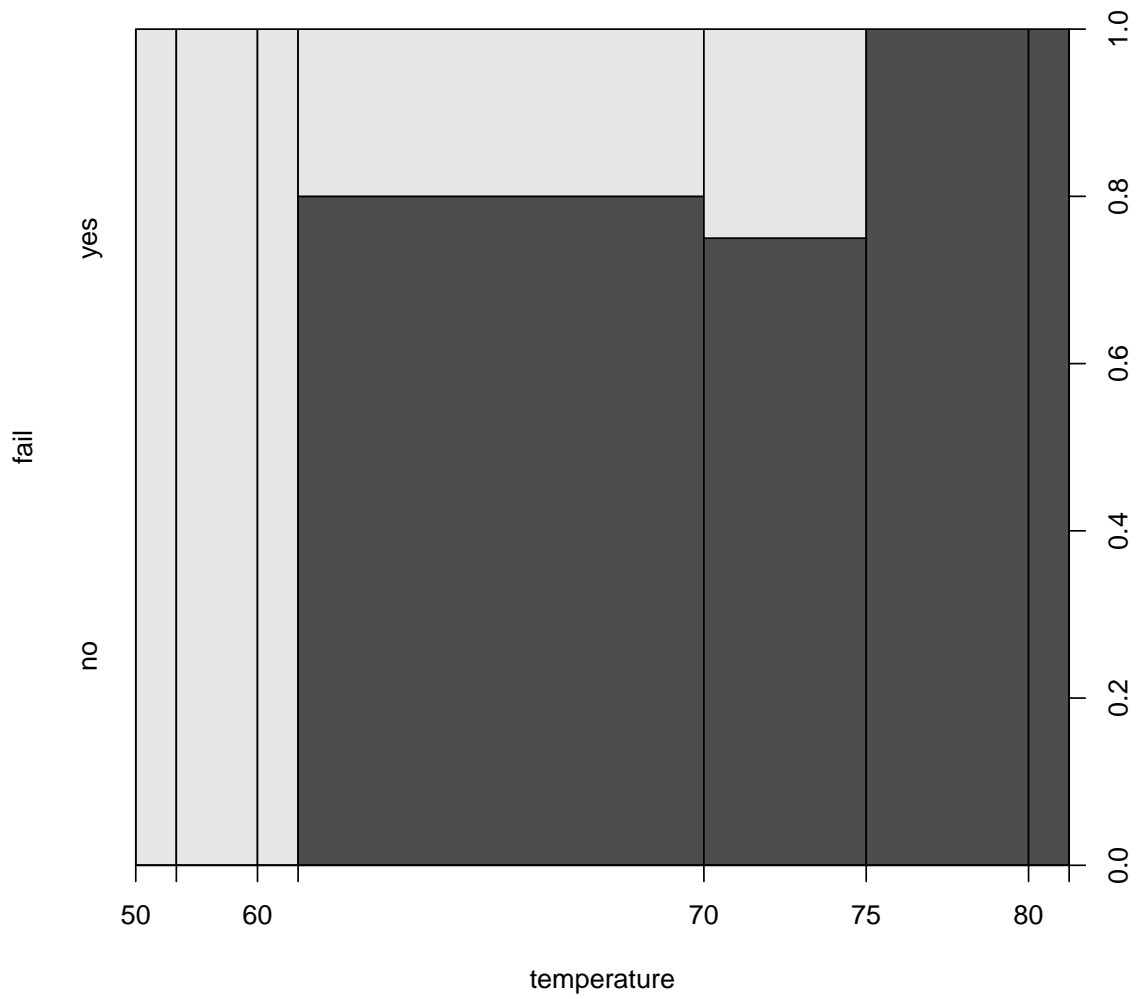


## NASA space shuttle o-ring failures

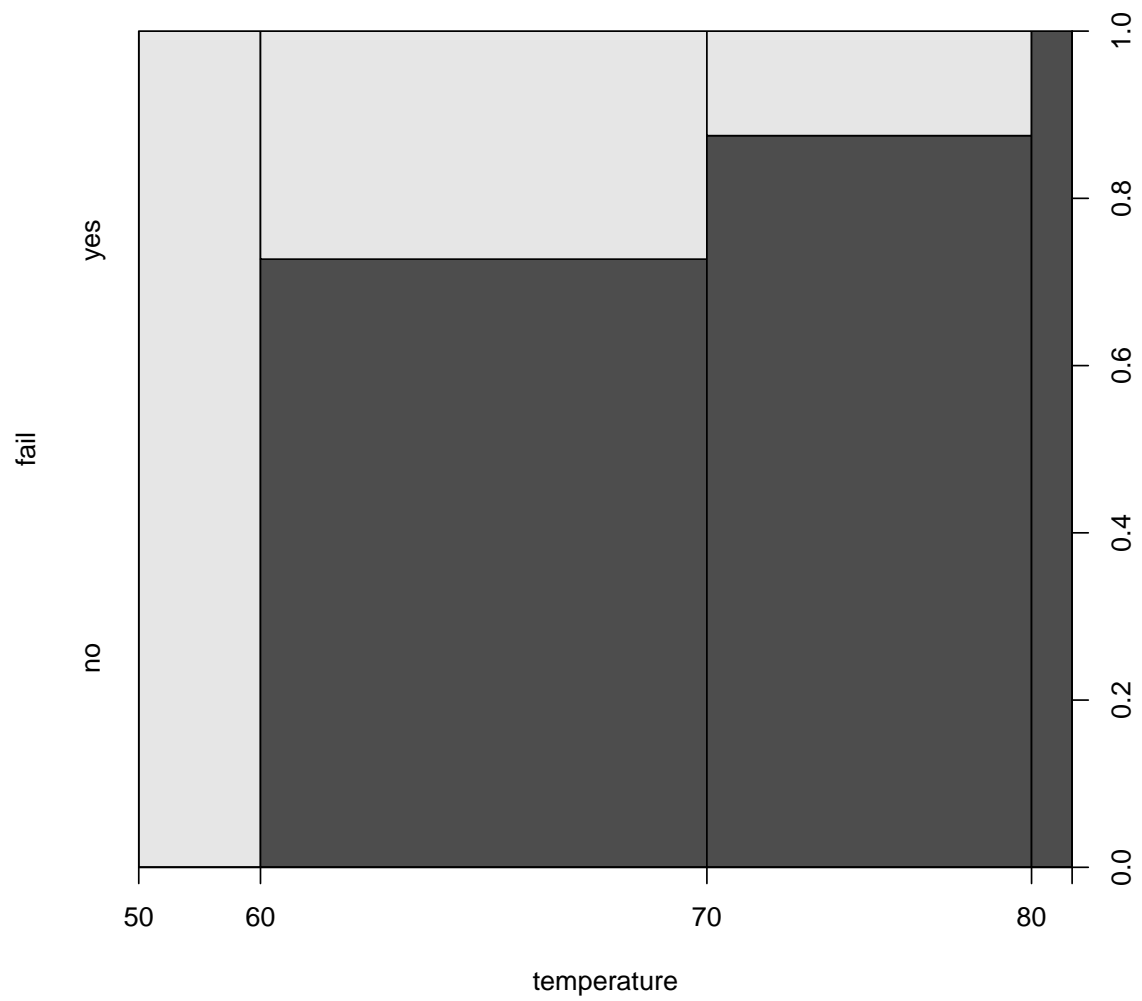
```
fail <- factor(c(2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1,
                1, 1, 1, 2, 1, 1, 1, 1, 1),
              levels = c(1, 2), labels = c("no", "yes"))
temperature <- c(53, 57, 58, 63, 66, 67, 67, 67, 68, 69, 70, 70,
                70, 70, 72, 73, 75, 75, 76, 76, 78, 79, 81)
```

## (dependence on a numerical variable)

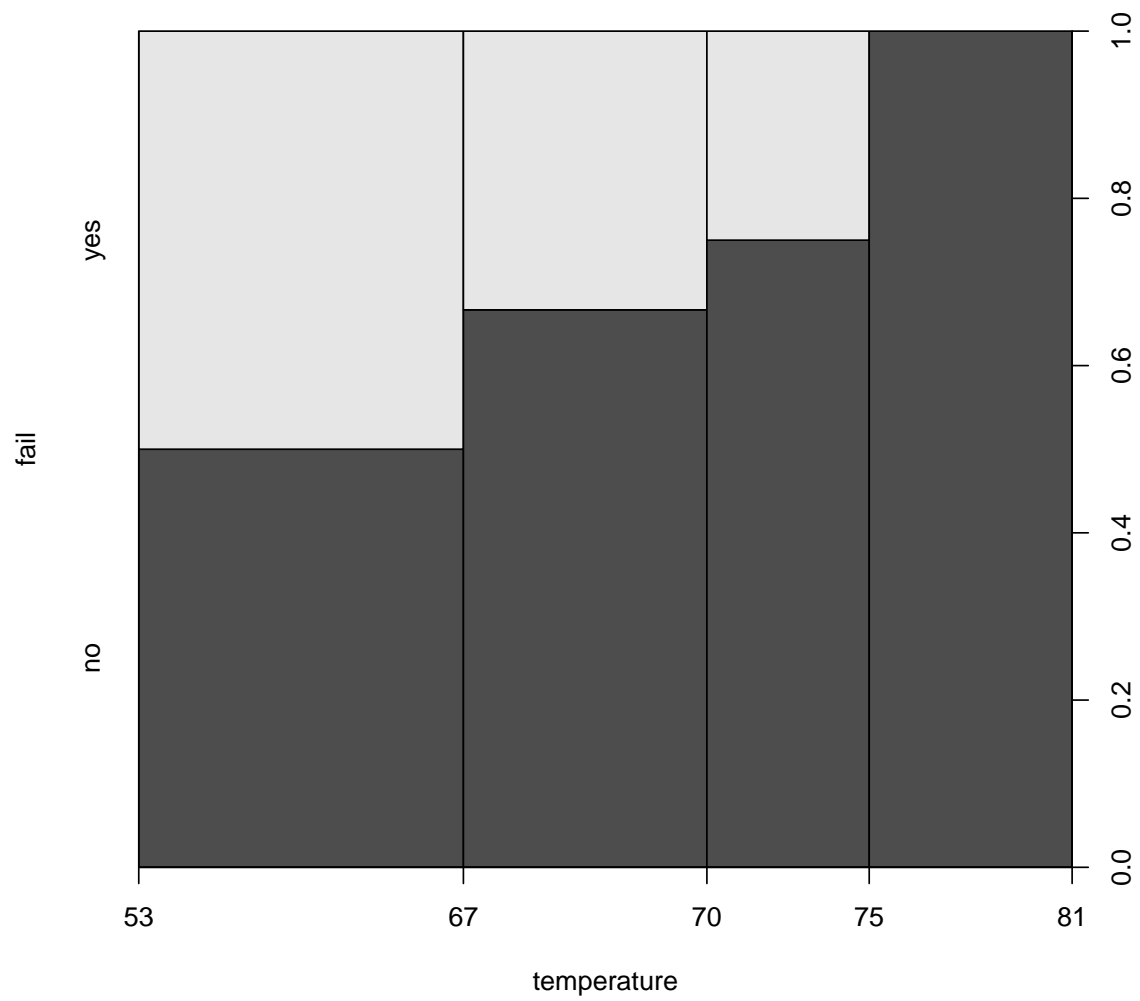
```
(spineplot(fail ~ temperature))
```



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

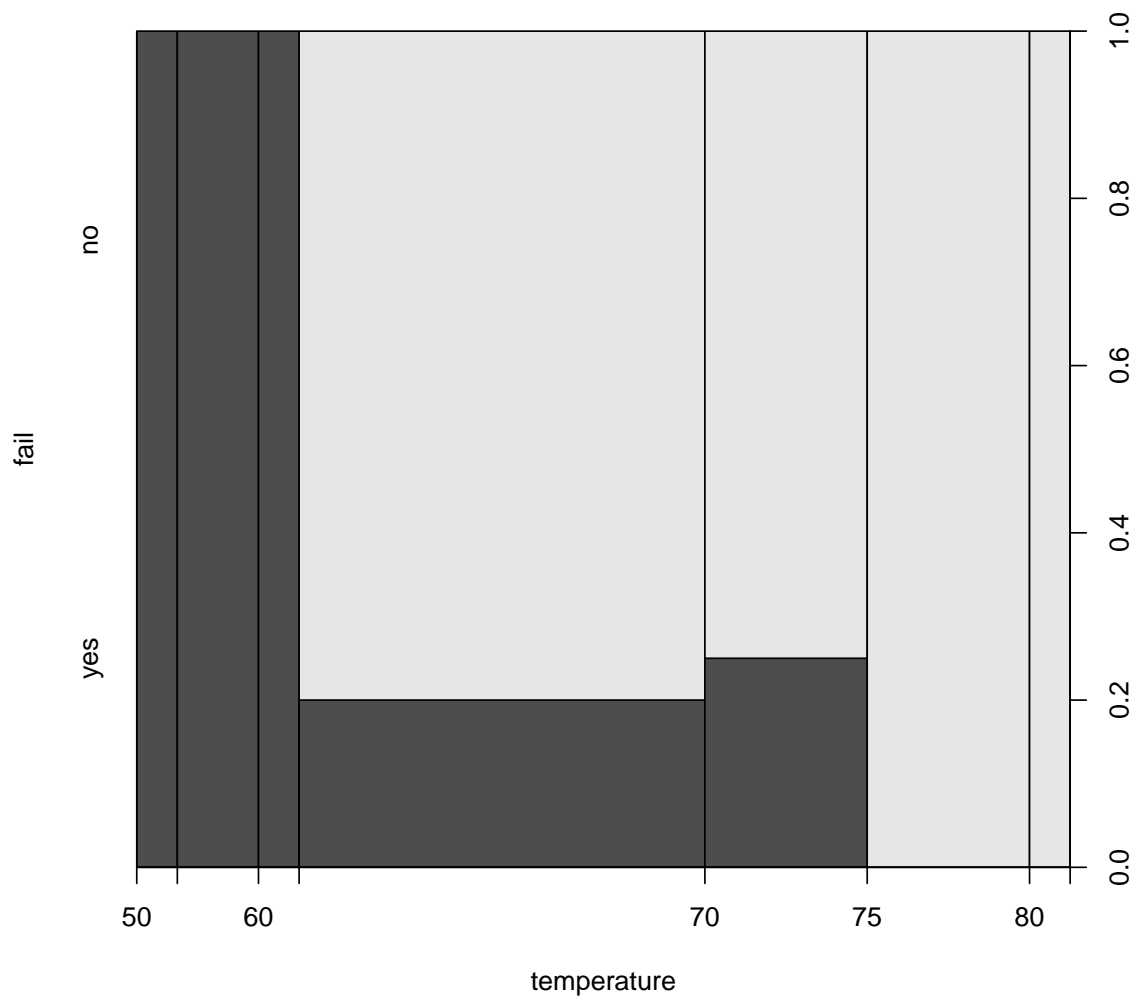


```
(spineplot(fail ~ temperature, breaks = quantile(temperature)))
```



## highlighting for failures

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

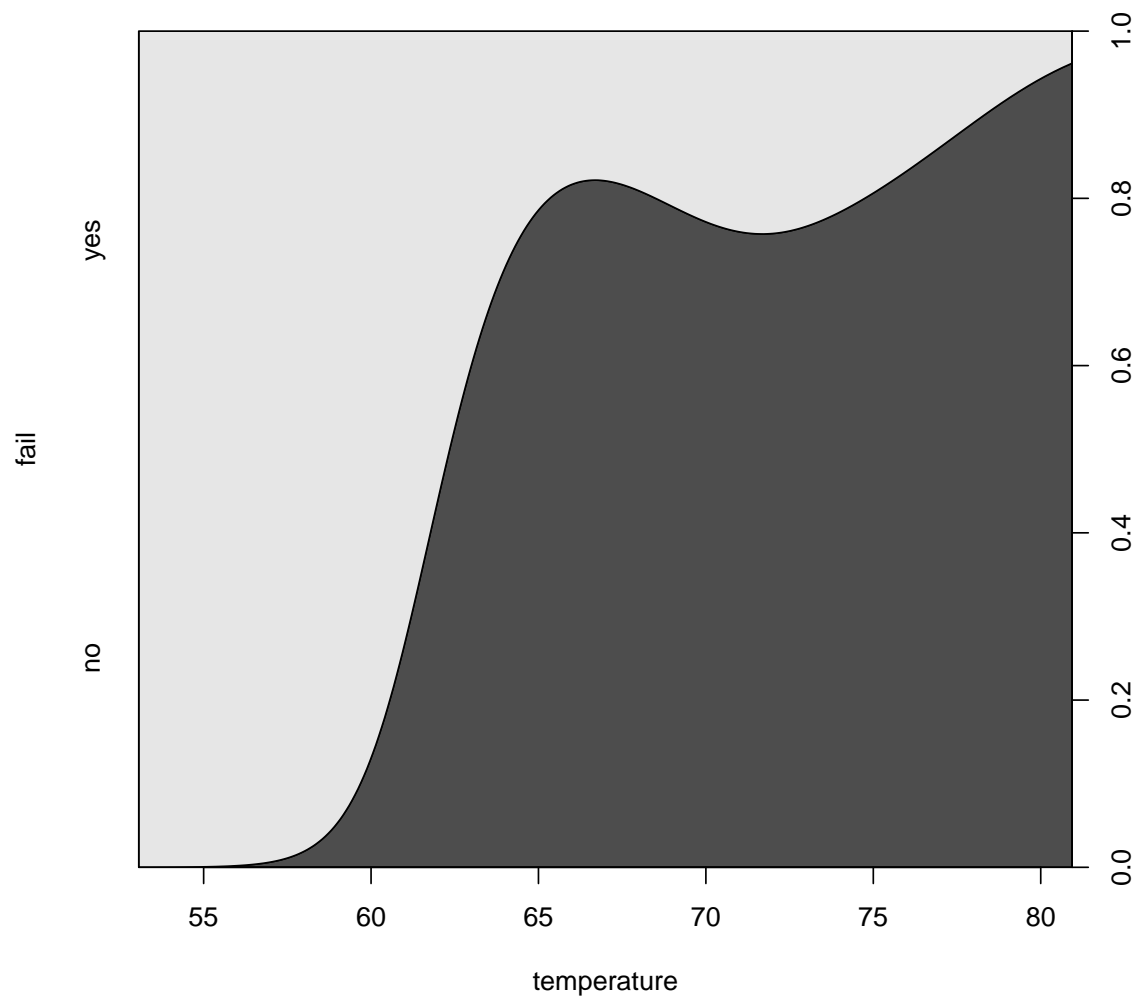


## 7.5 cdplot (Conditional Density Plots)

NASA space shuttle o-ring failures

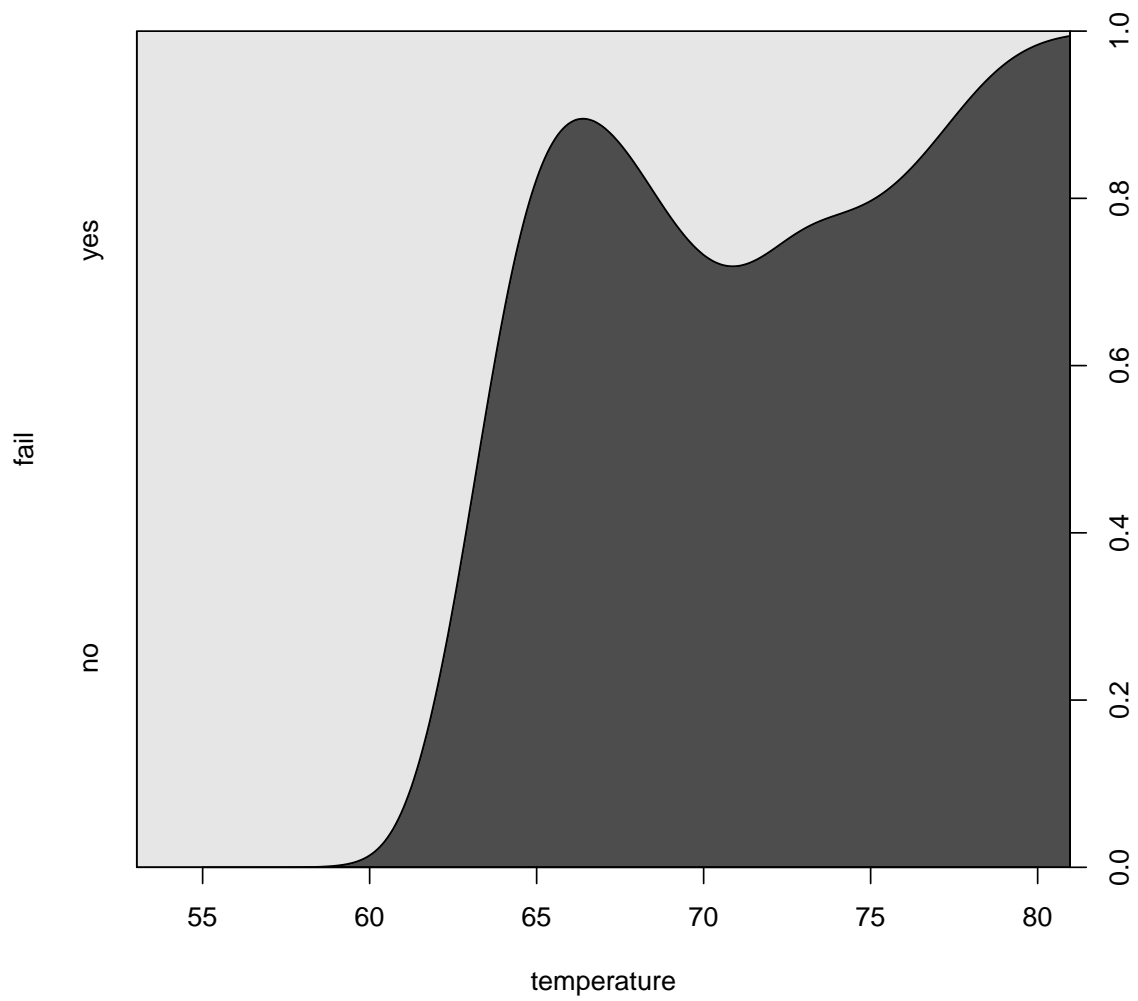
```
fail <- factor(c(2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1, 1,
                1, 2, 1, 1, 1, 1, 1),
              levels = 1:2, labels = c("no", "yes"))
temperature <- c(53, 57, 58, 63, 66, 67, 67, 67, 68, 69, 70, 70,
                70, 70, 72, 73, 75, 75, 76, 76, 78, 79, 81)
```

```
cdplot(fail ~ temperature)
```

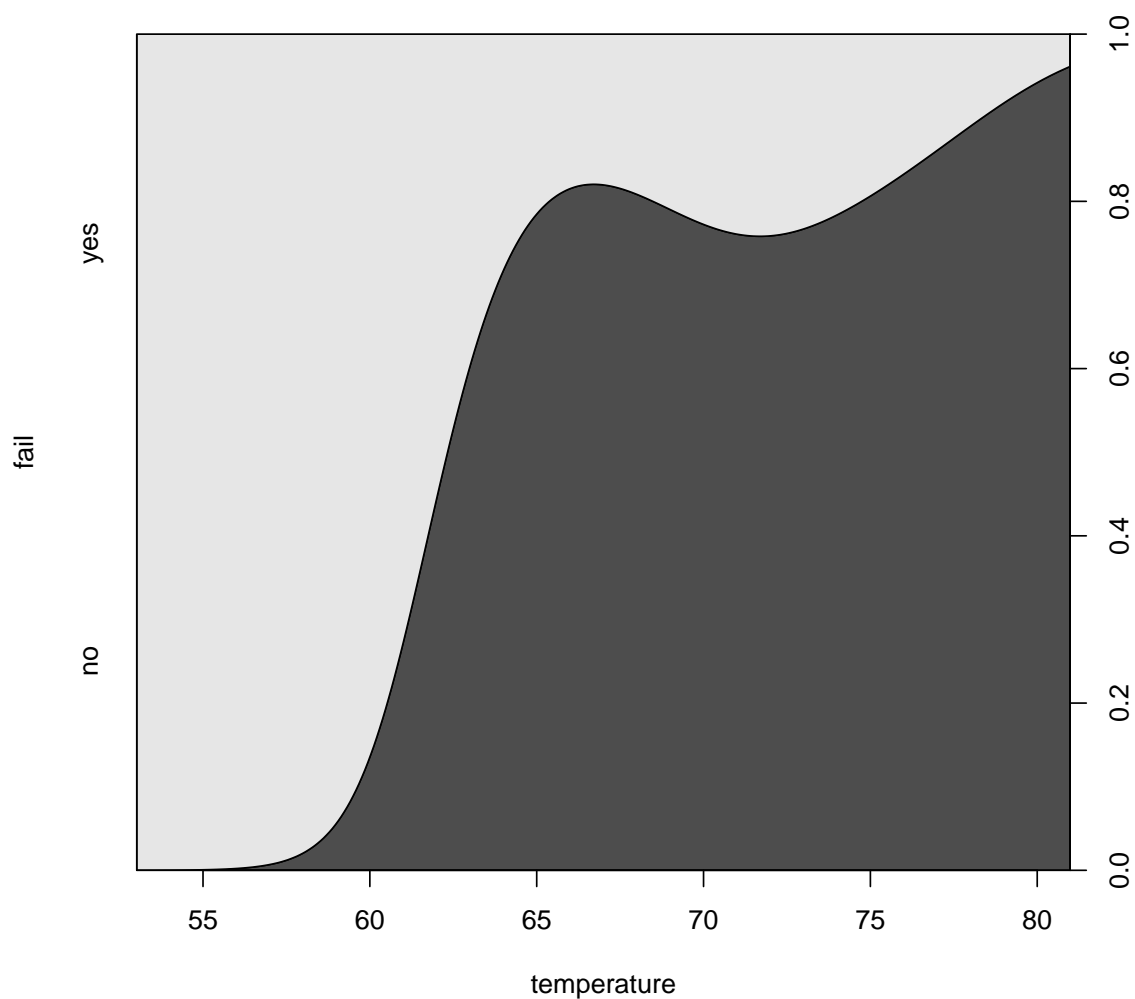


```
cdplot(fail ~ temperature, bw = 2)
```



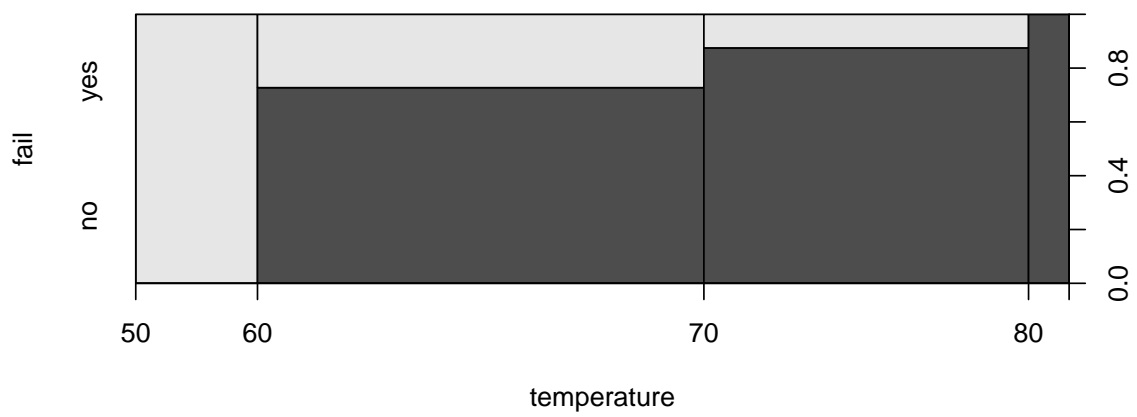
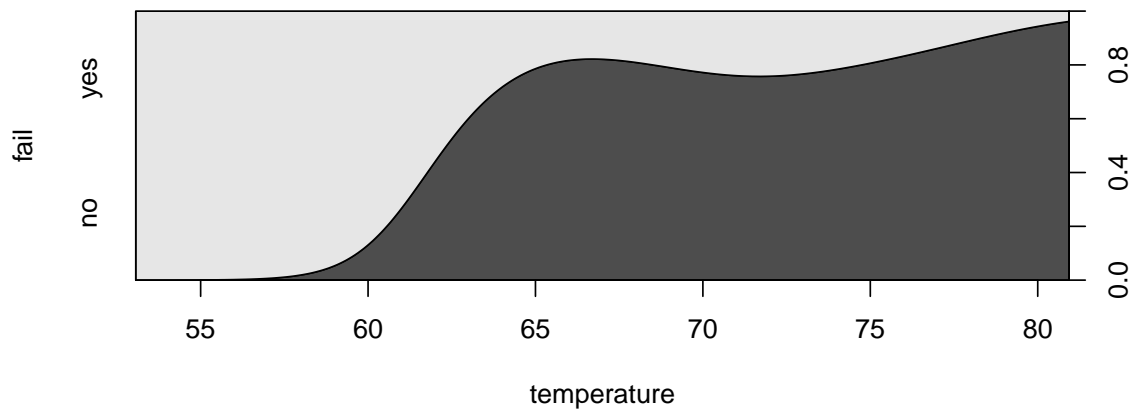


```
cdplot(fail ~ temperature, bw = "SJ")
```



compare with spinogram on the same graph

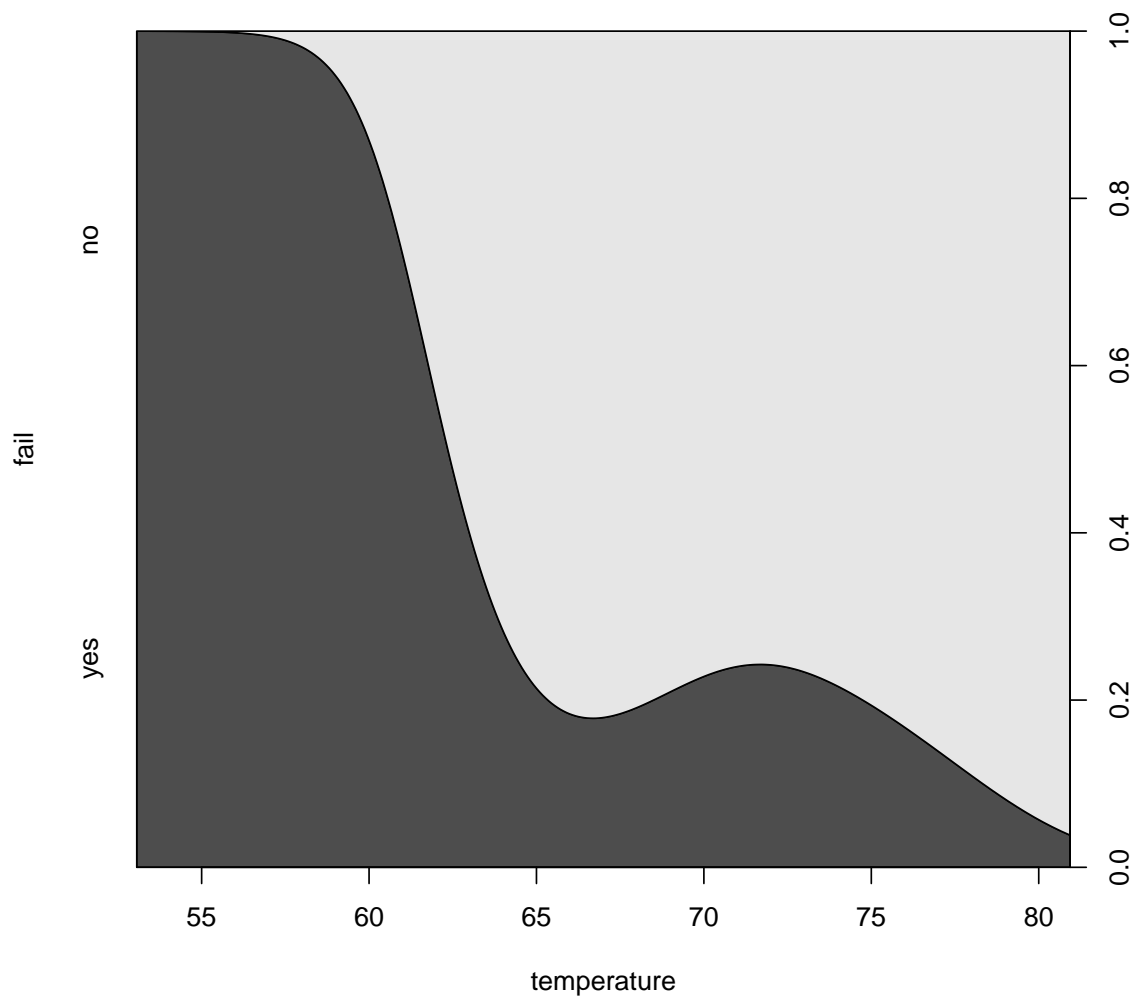
```
layout(1:2)
cdplot(fail ~ temperature)
(spineplot(fail ~ temperature, breaks = 3))
```



```
layout(1)
```

highlighting for failures

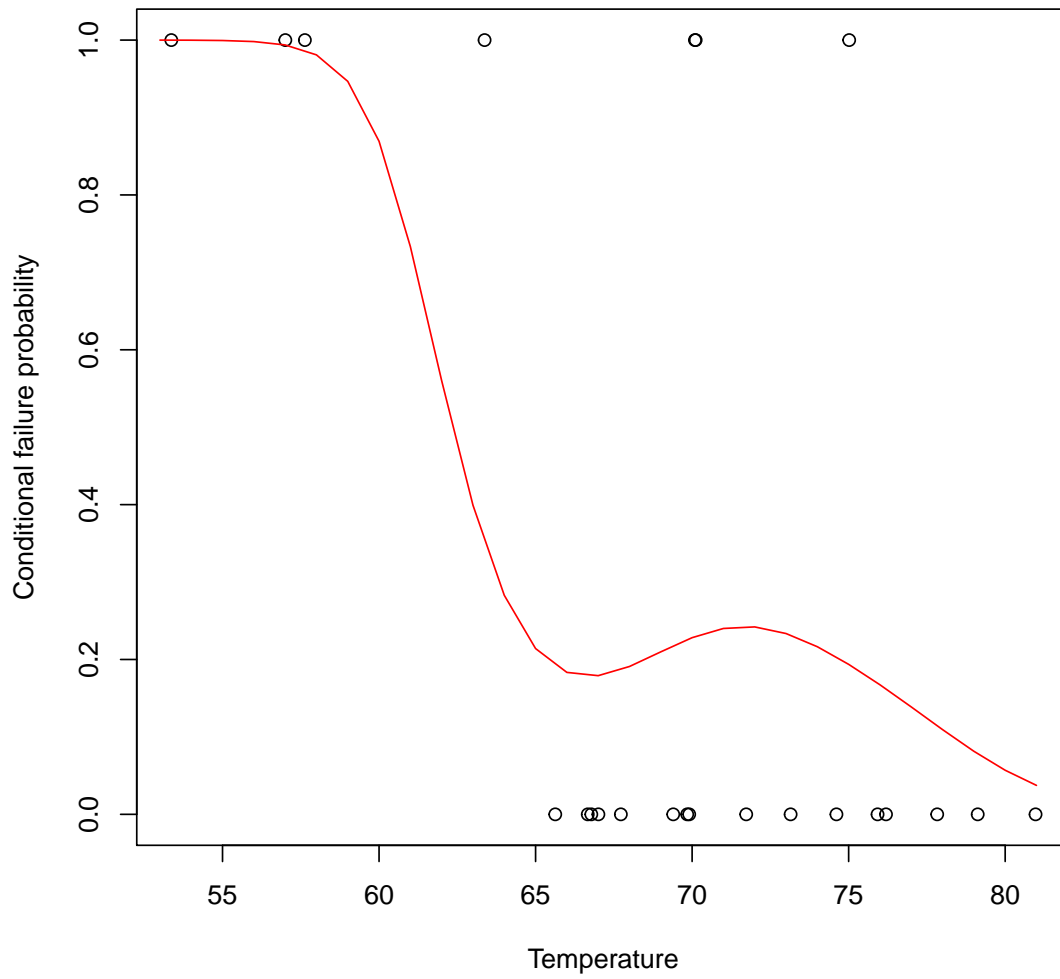
```
cdplot(fail ~ temperature, ylevels = 2:1)
```



scatter plot with conditional density

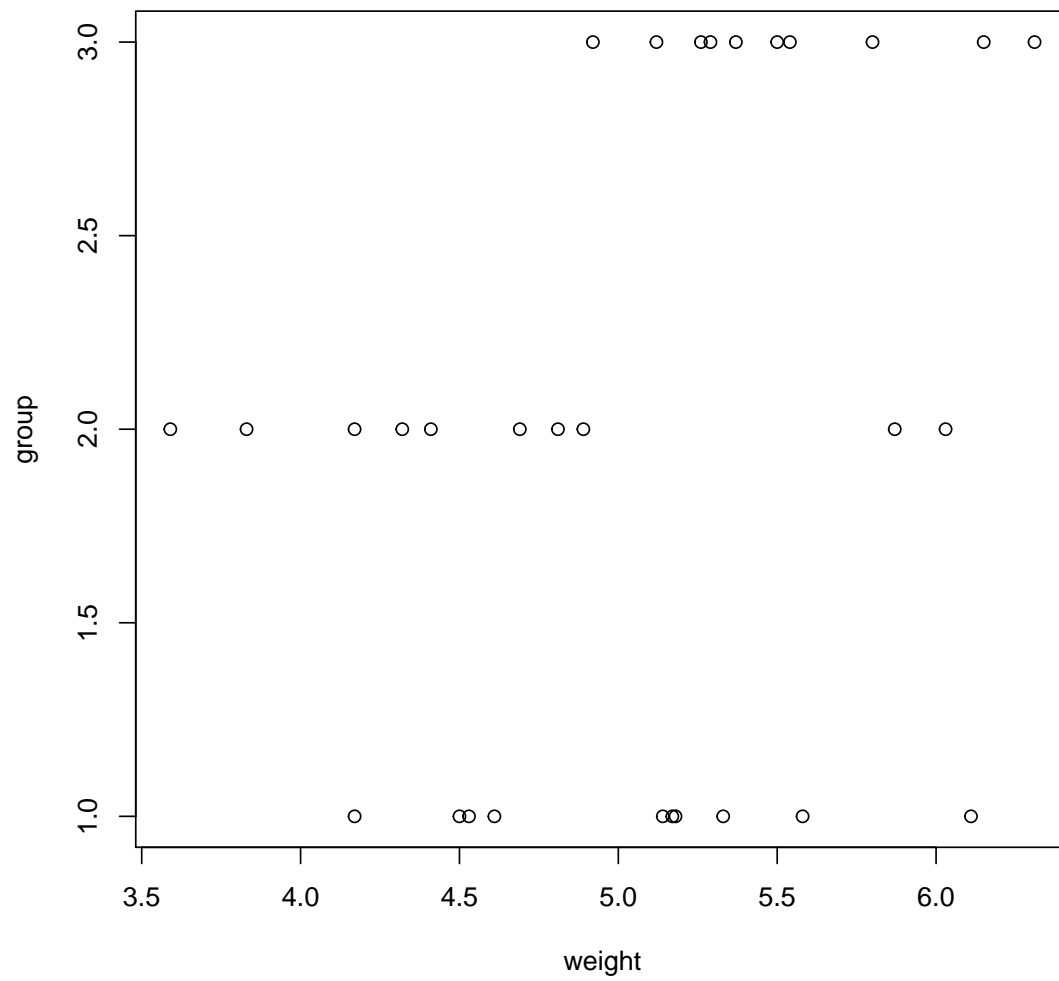
```
cdens <- cdplot(fail ~ temperature, plot = FALSE)
```

```
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)
```

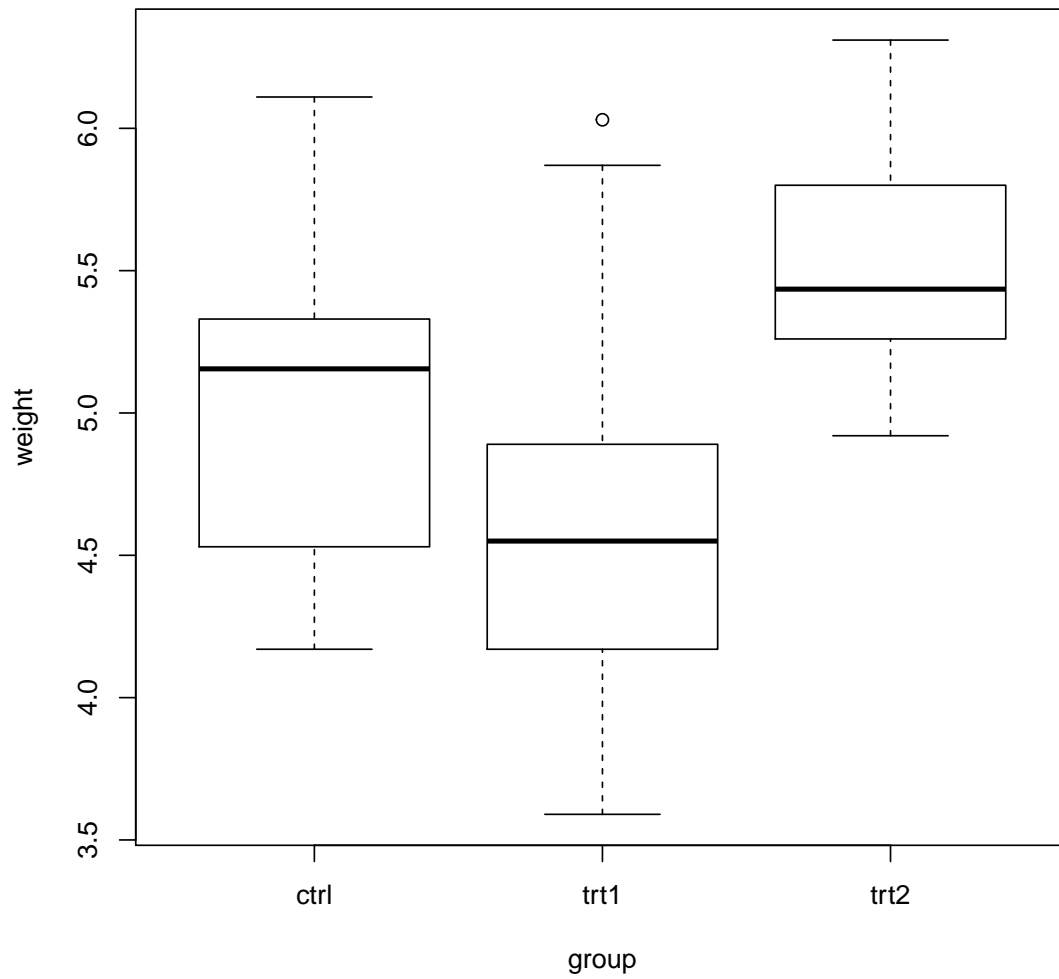


## 8 Plot factor variables

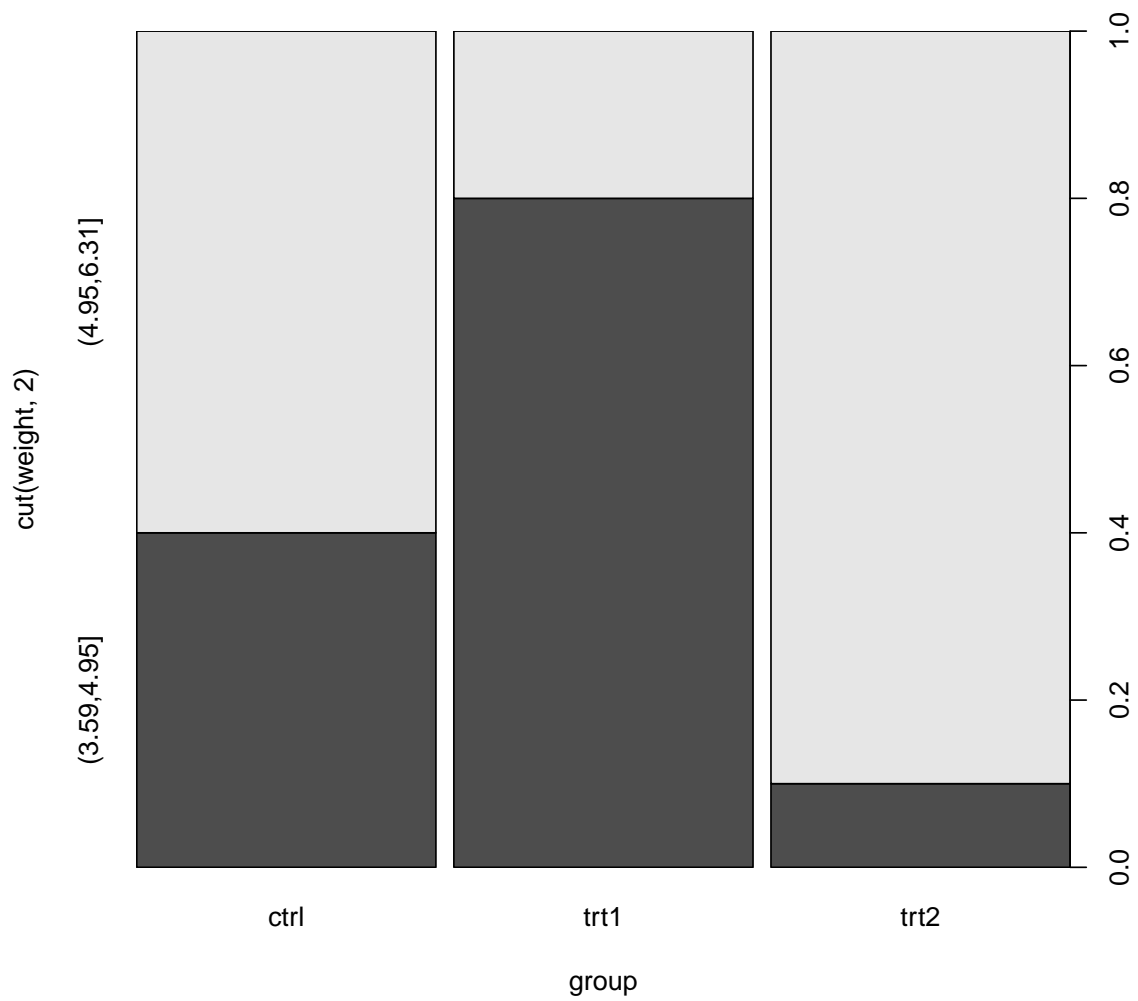
```
require(grDevices)
plot(PlantGrowth) # -> plot.data.frame
```



```
plot(weight ~ group, data = PlantGrowth) # numeric vector ~ factor
```



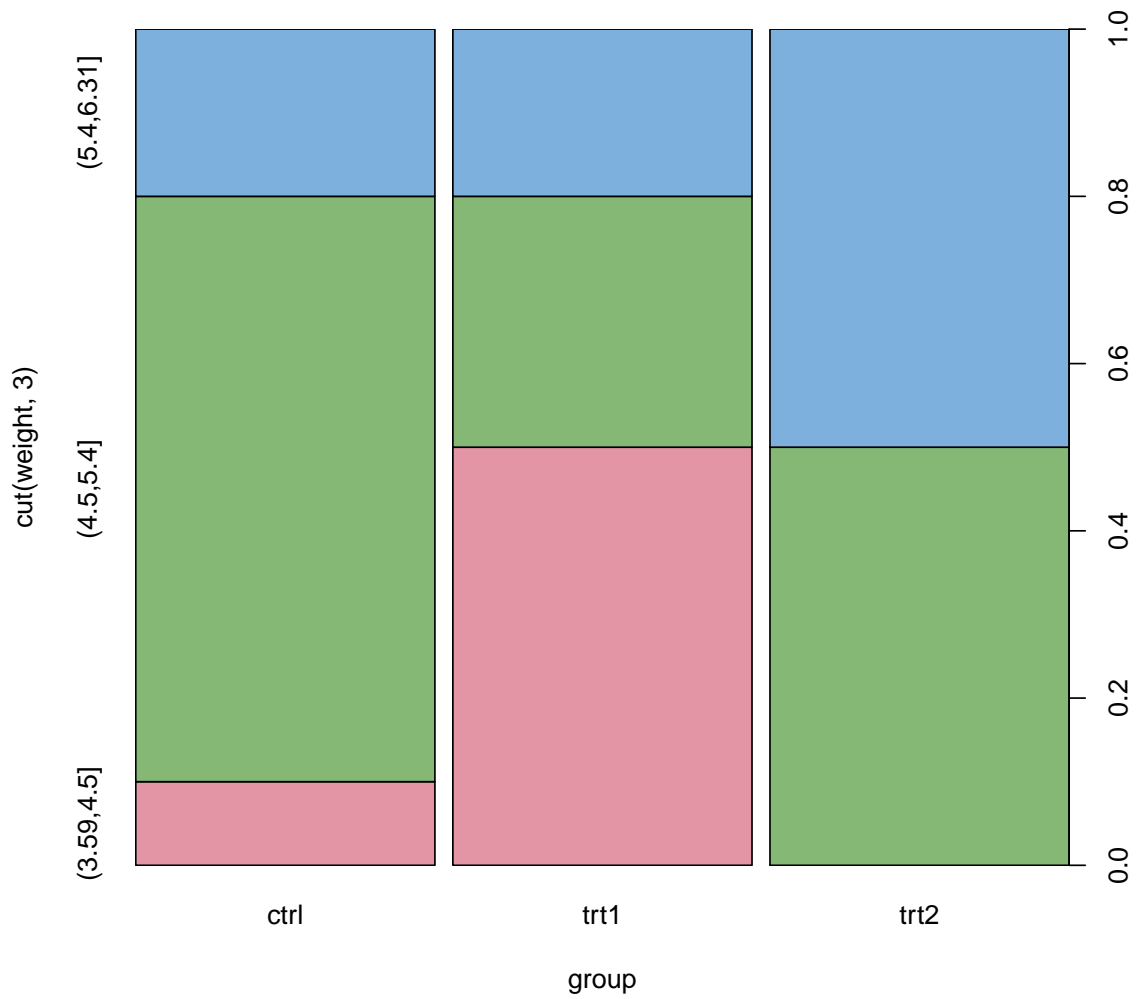
```
plot(cut(weight, 2) ~ group, data = PlantGrowth) # factor ~ factor
```



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

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

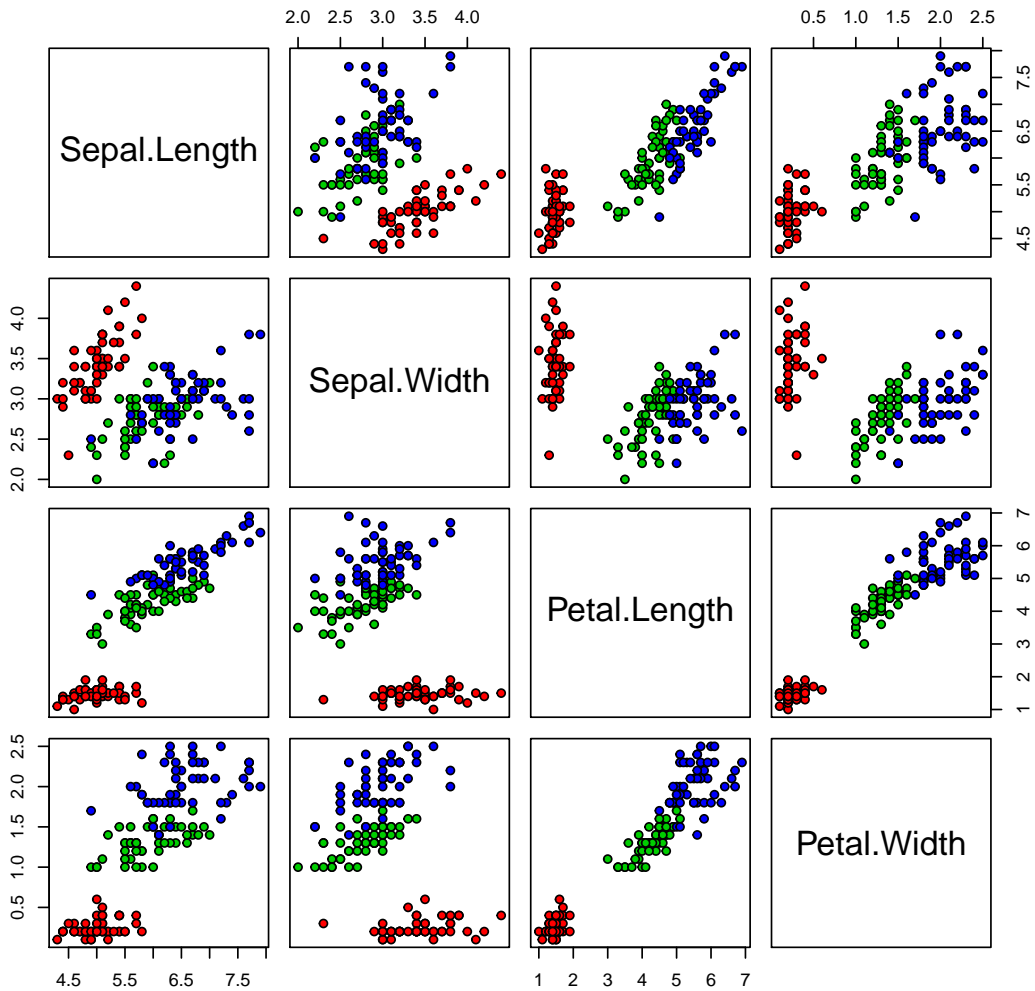




## 9 Matrix plot

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

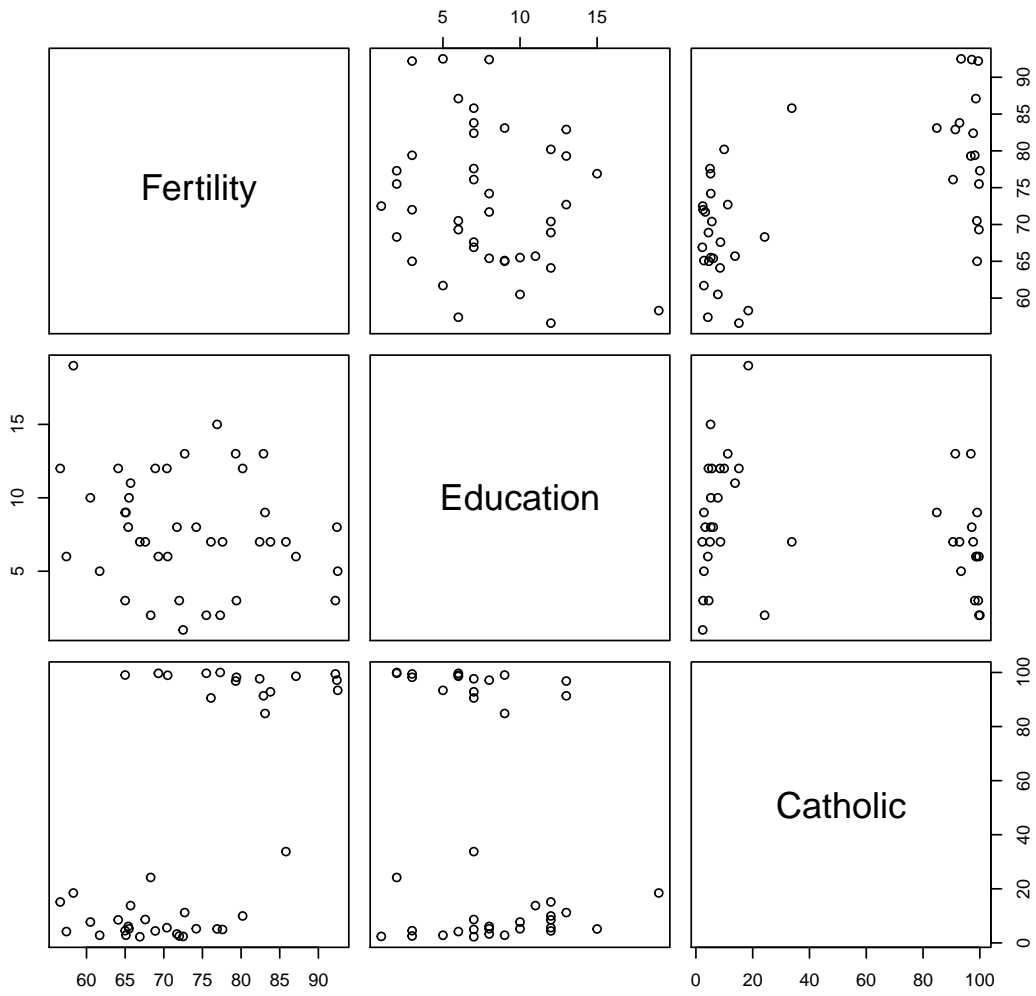
## Anderson's Iris Data -- 3 species



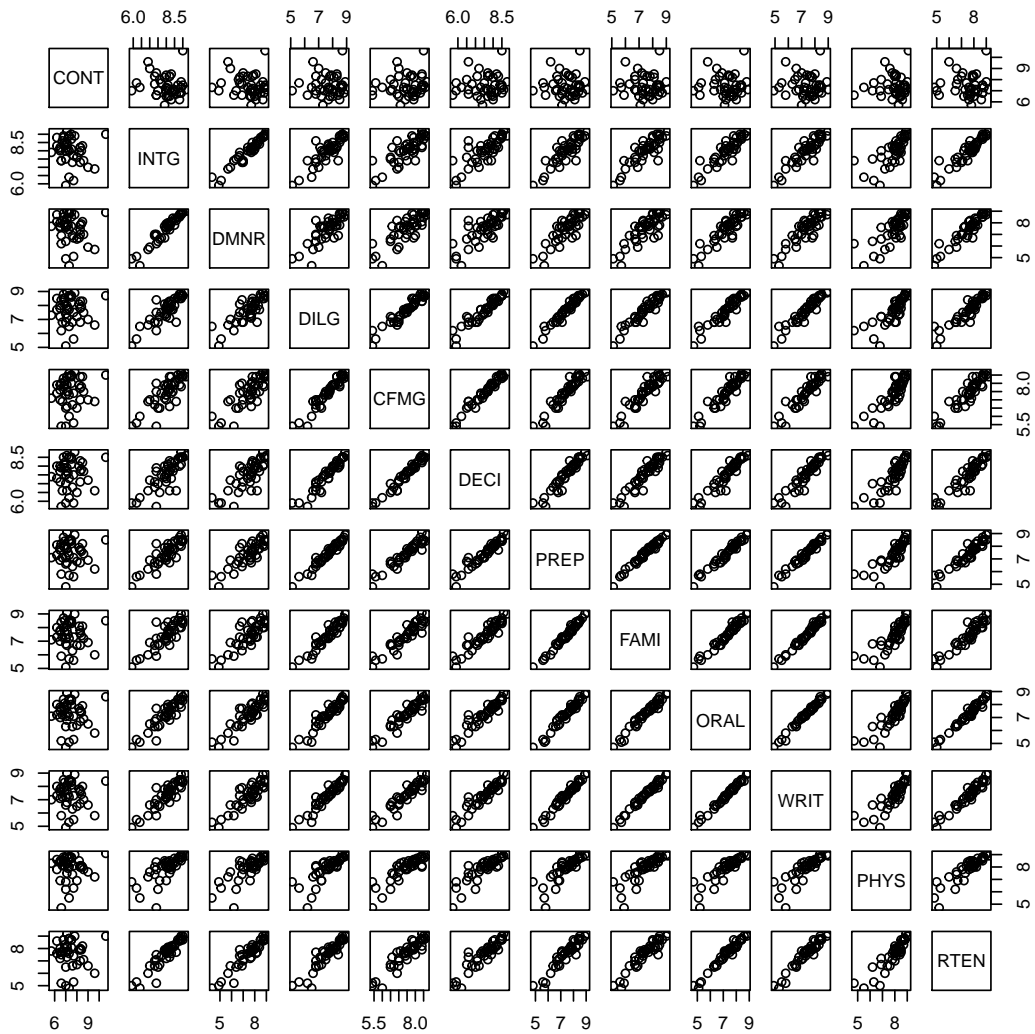
formula method

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

Swiss data, Education < 20



```
pairs(USJudgeRatings)
```



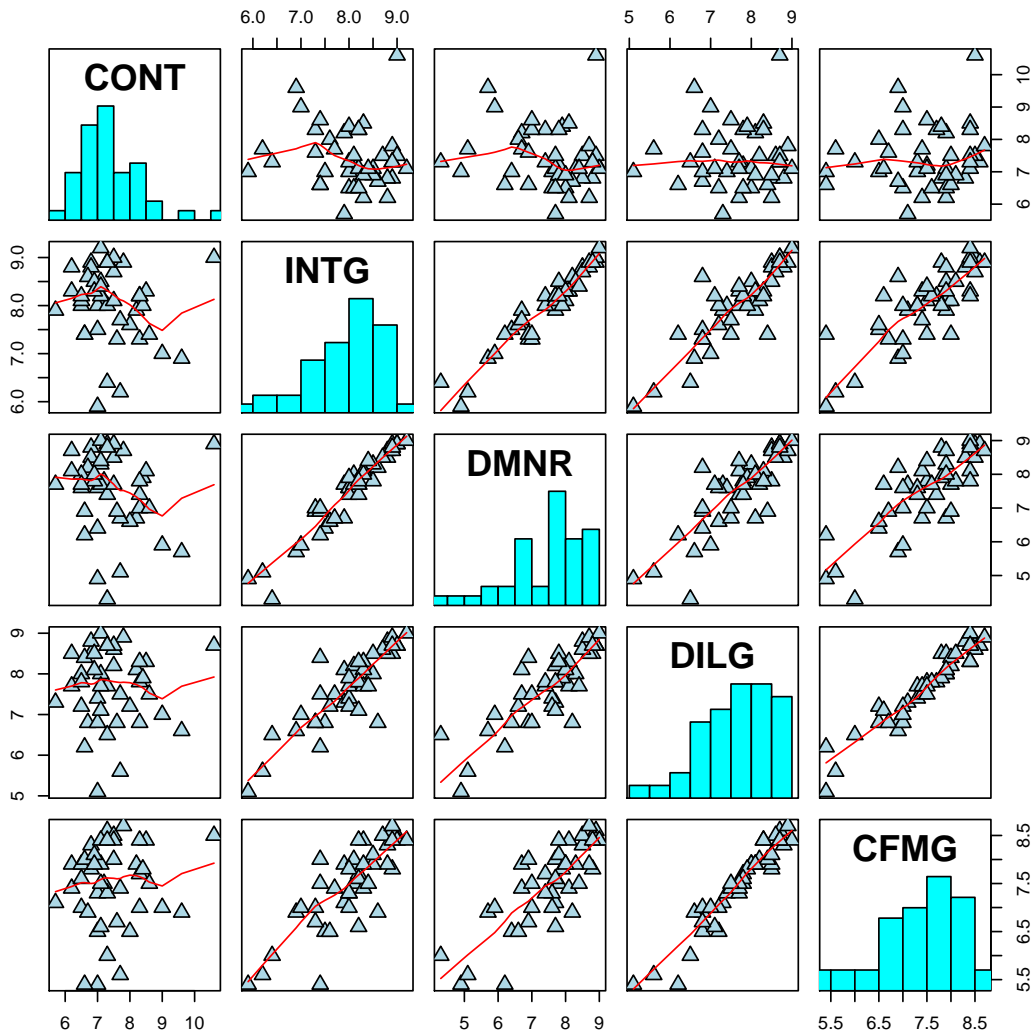
put histograms on the diagonal

```
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="")
}

```

```

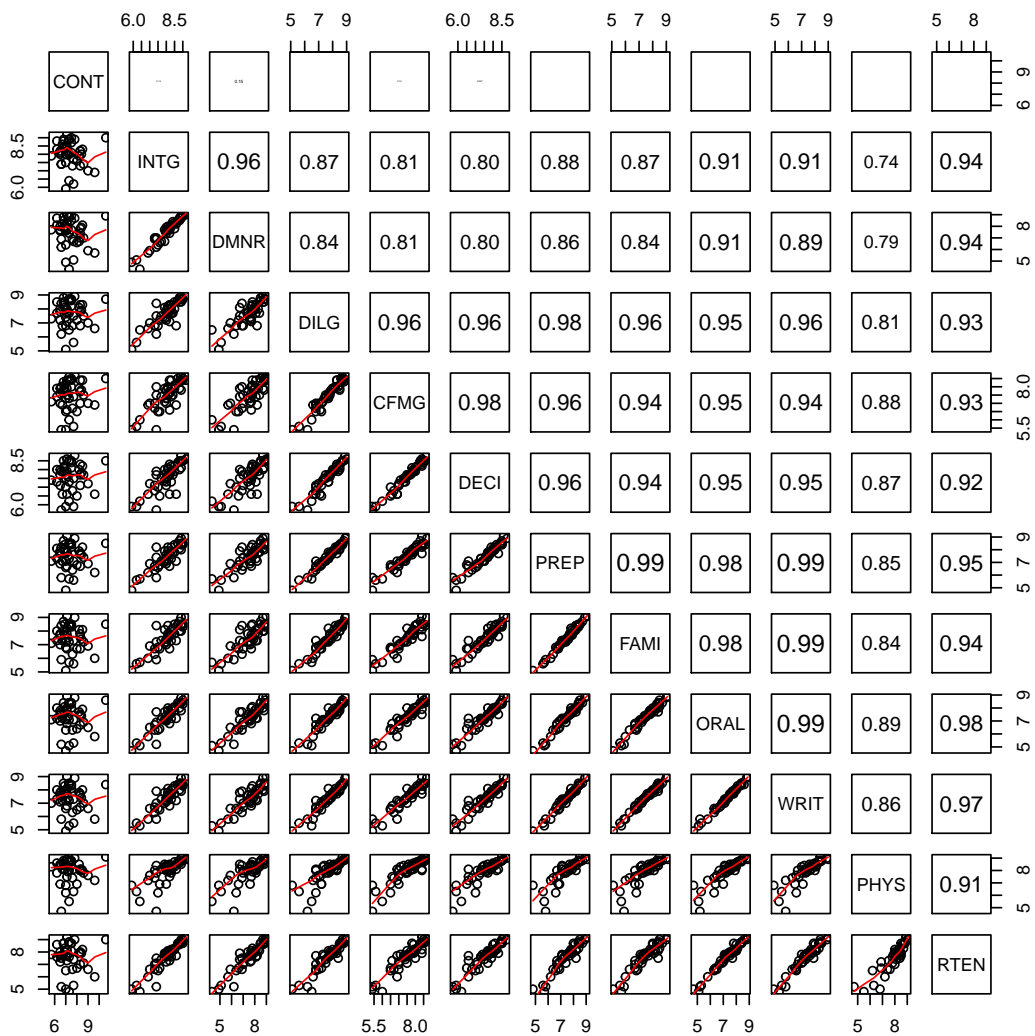
if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
text(0.5, 0.5, txt, cex = cex.cor * r)
}

```

```

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

```

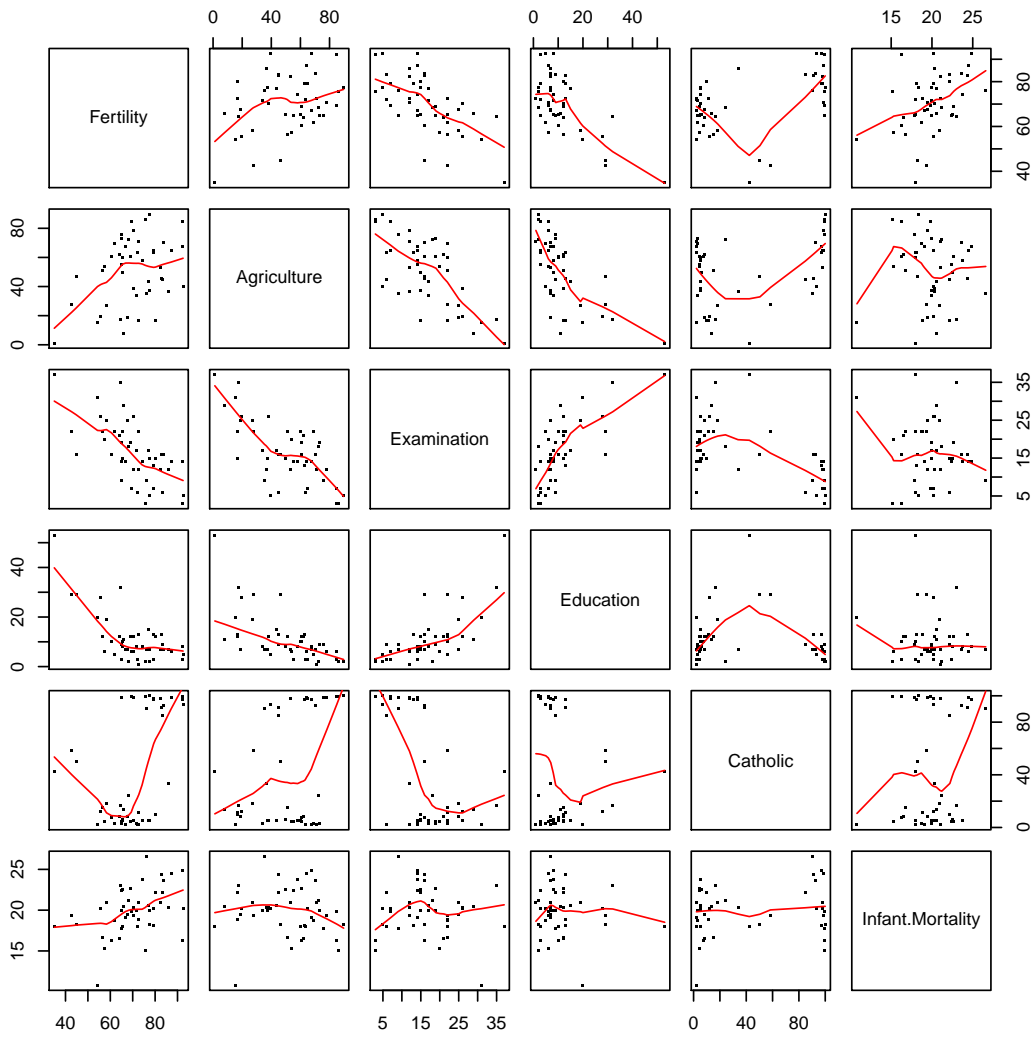


## 10 Simplepanel plots

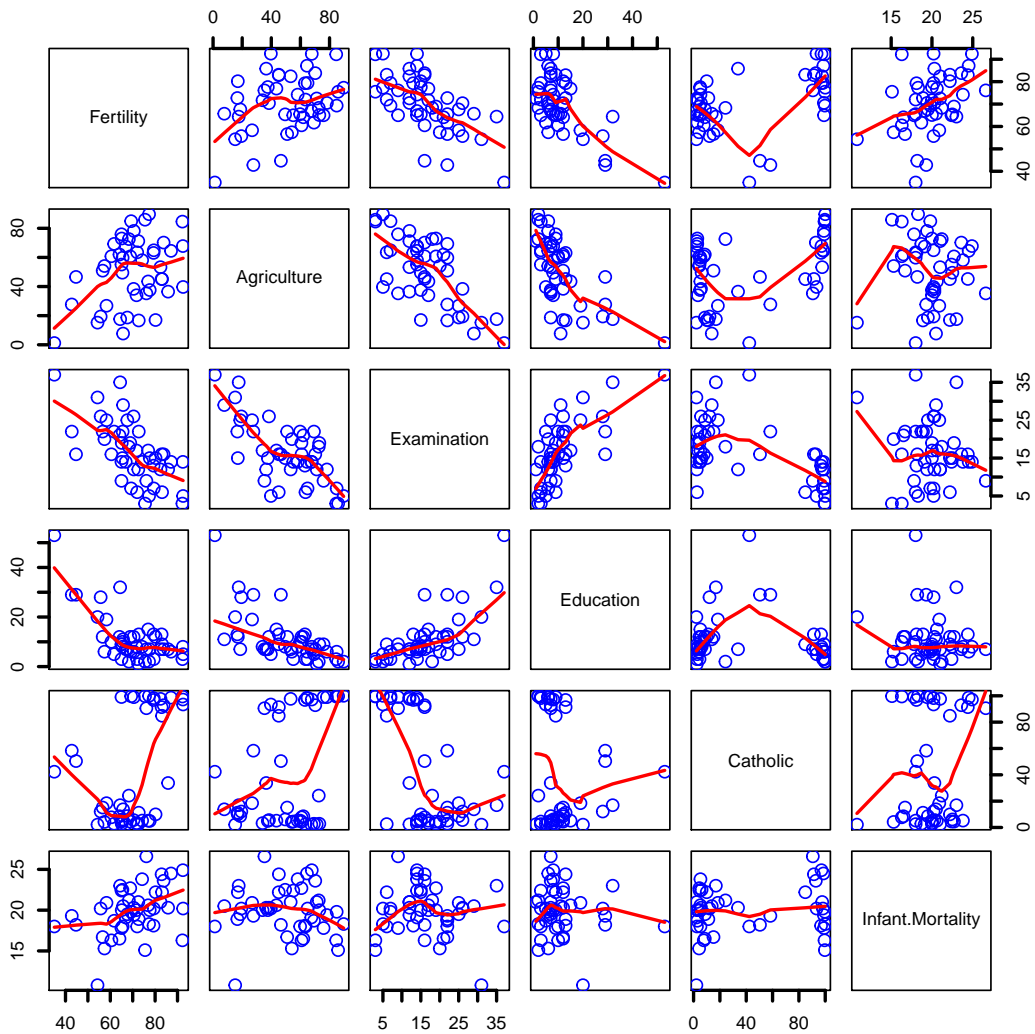
```

pairs(swiss, panel = panel.smooth, pch = ".") # emphasize the smooths

```



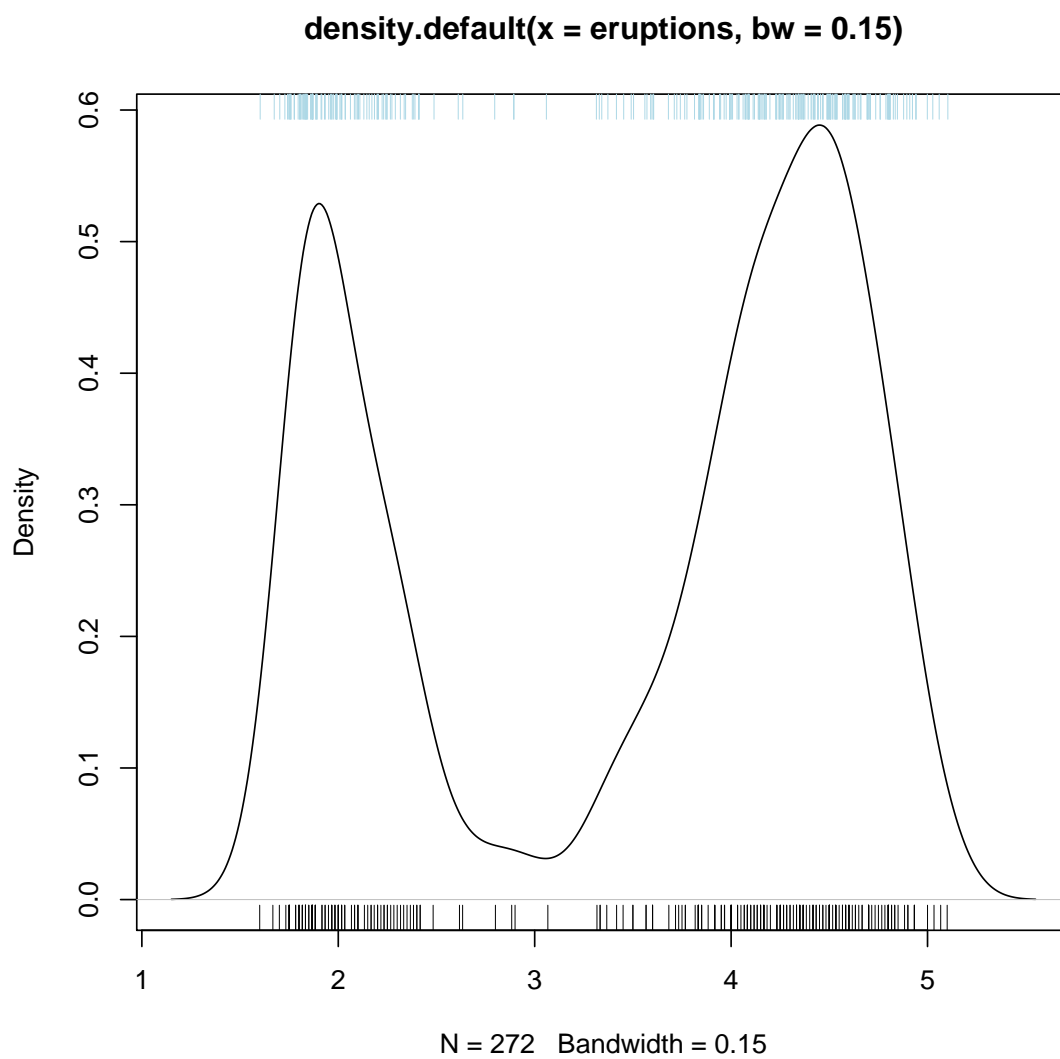
```
pairs(swiss, panel = panel.smooth, lwd = 2, cex = 1.5, col = "blue") # hmm...
```



## 11 jitter

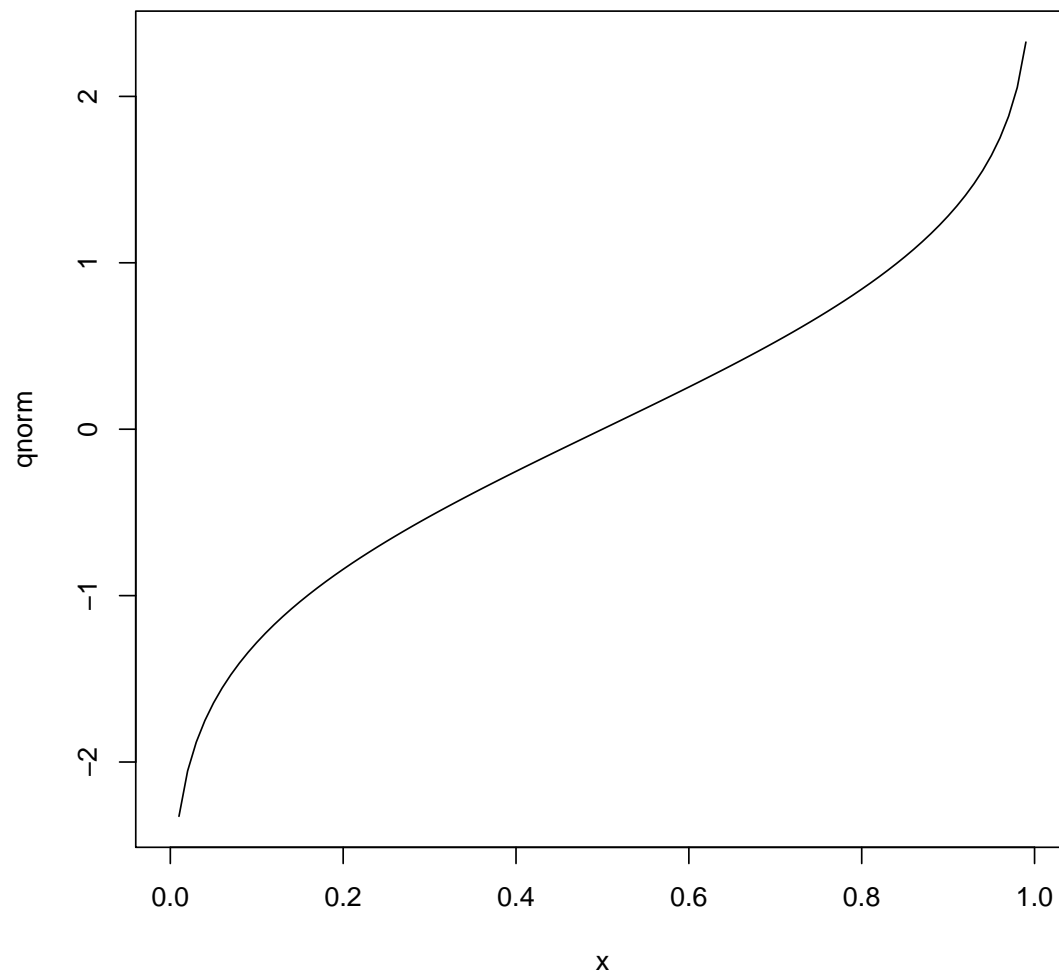
```
require(stats) # both 'density' and its default method
with(faithful, {
  plot(density(eruptions, bw = 0.15))
  rug(eruptions)
  rug(jitter(eruptions, amount = 0.01), side = 3, col = "light blue")
})
```



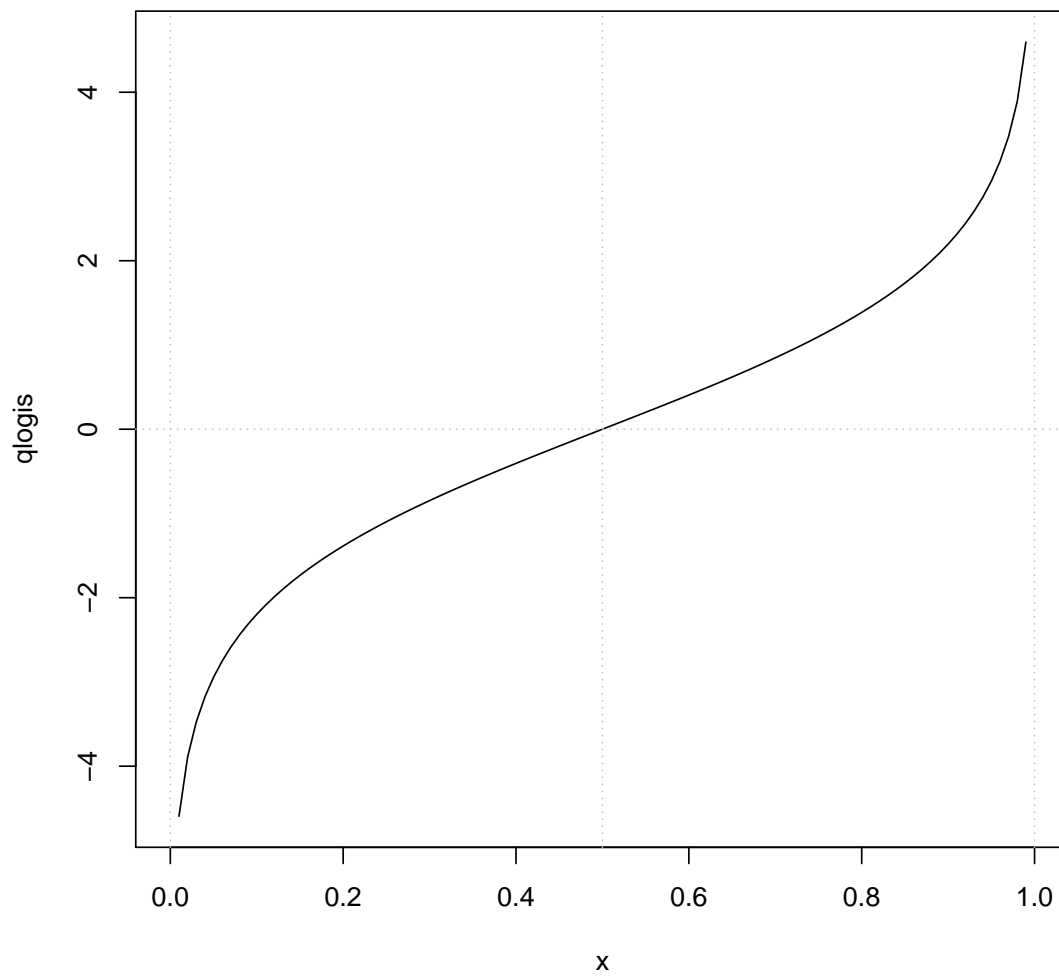


## 12 curves

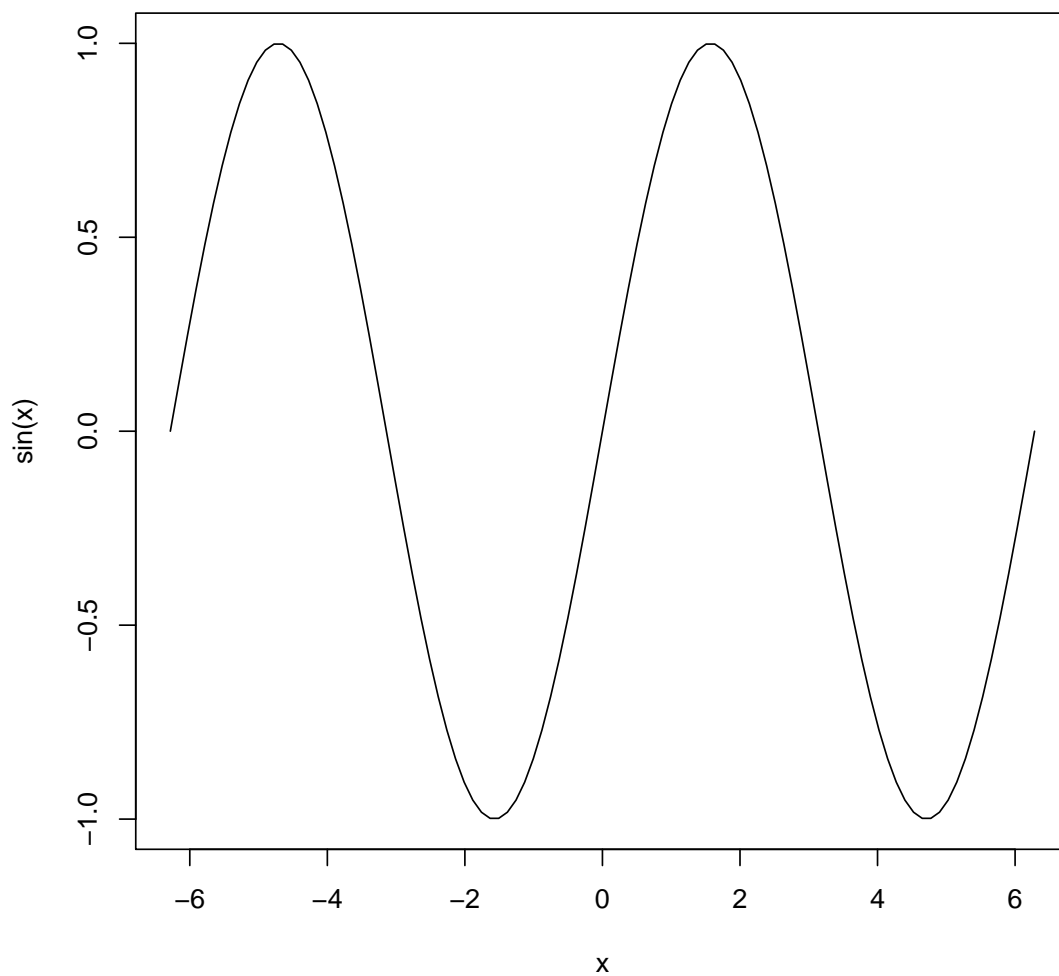
```
plot(qnorm)
```



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

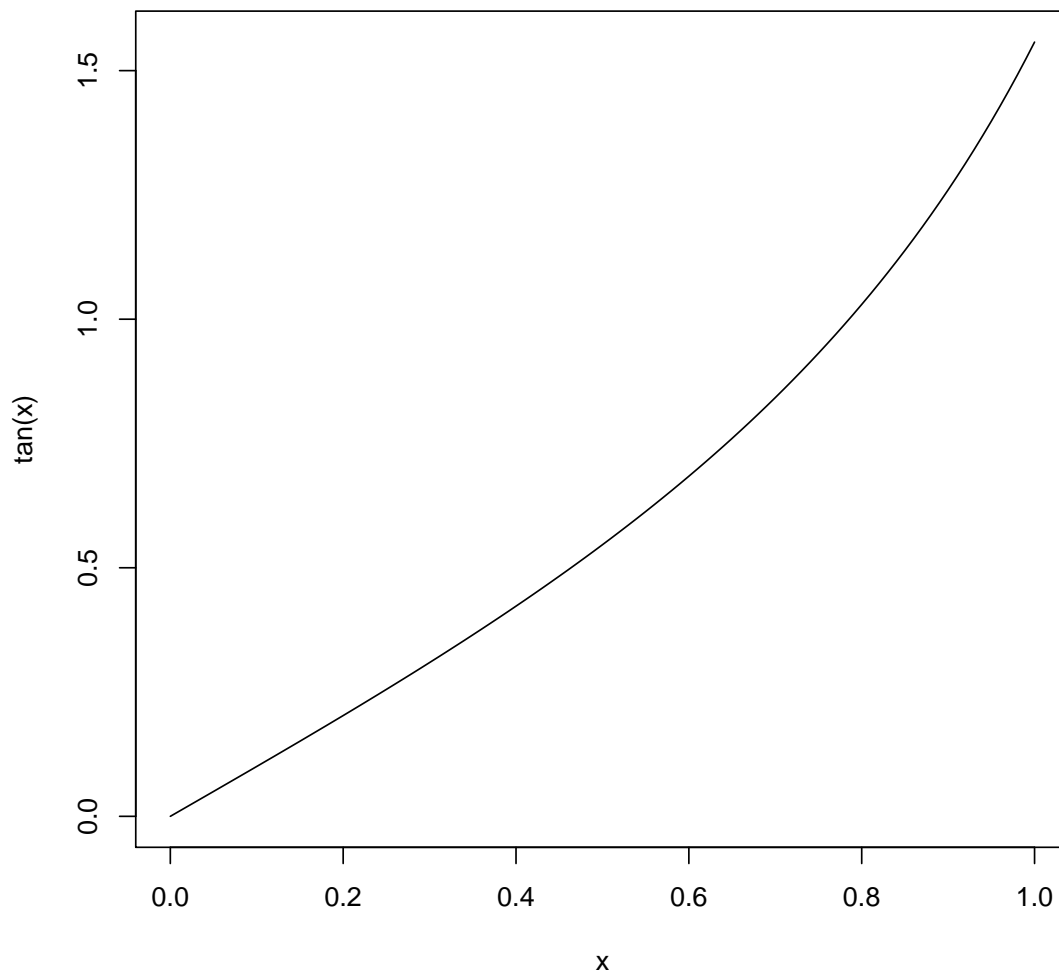
**The Inverse Logit : qlogis()**

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



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

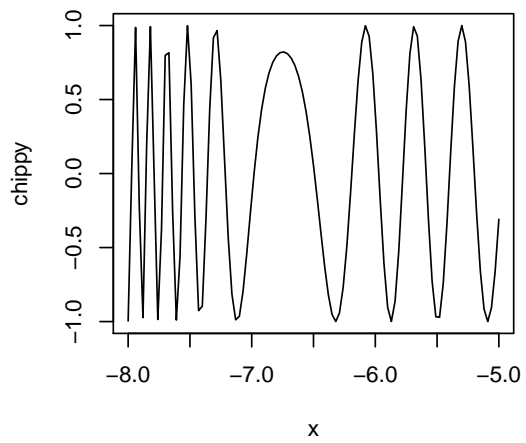
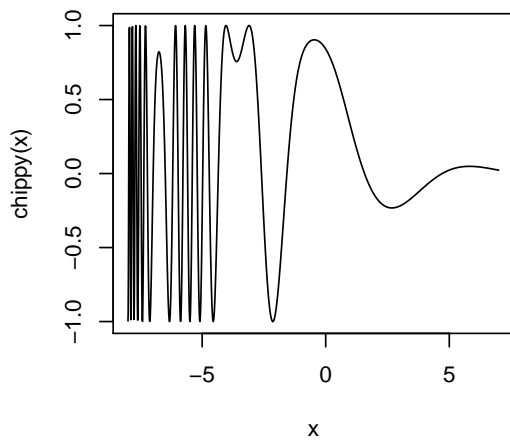
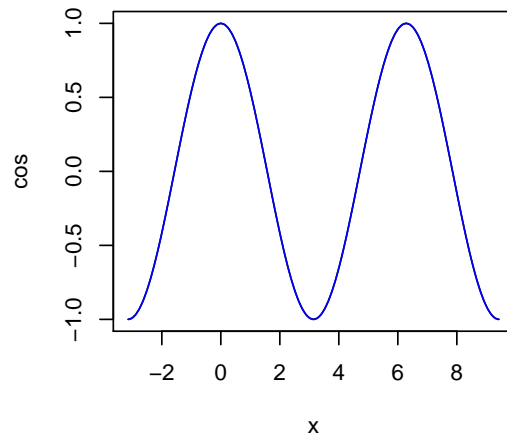
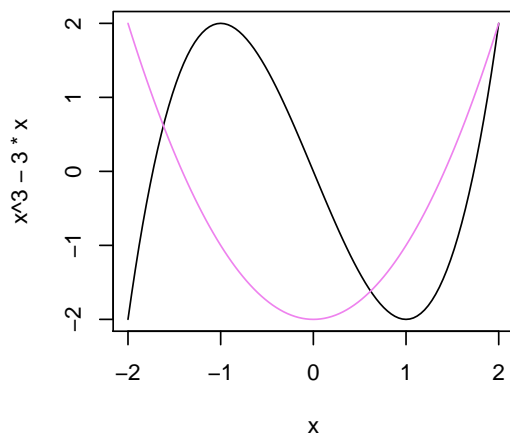
**curve(tan) --> same x-scale as previous plot**



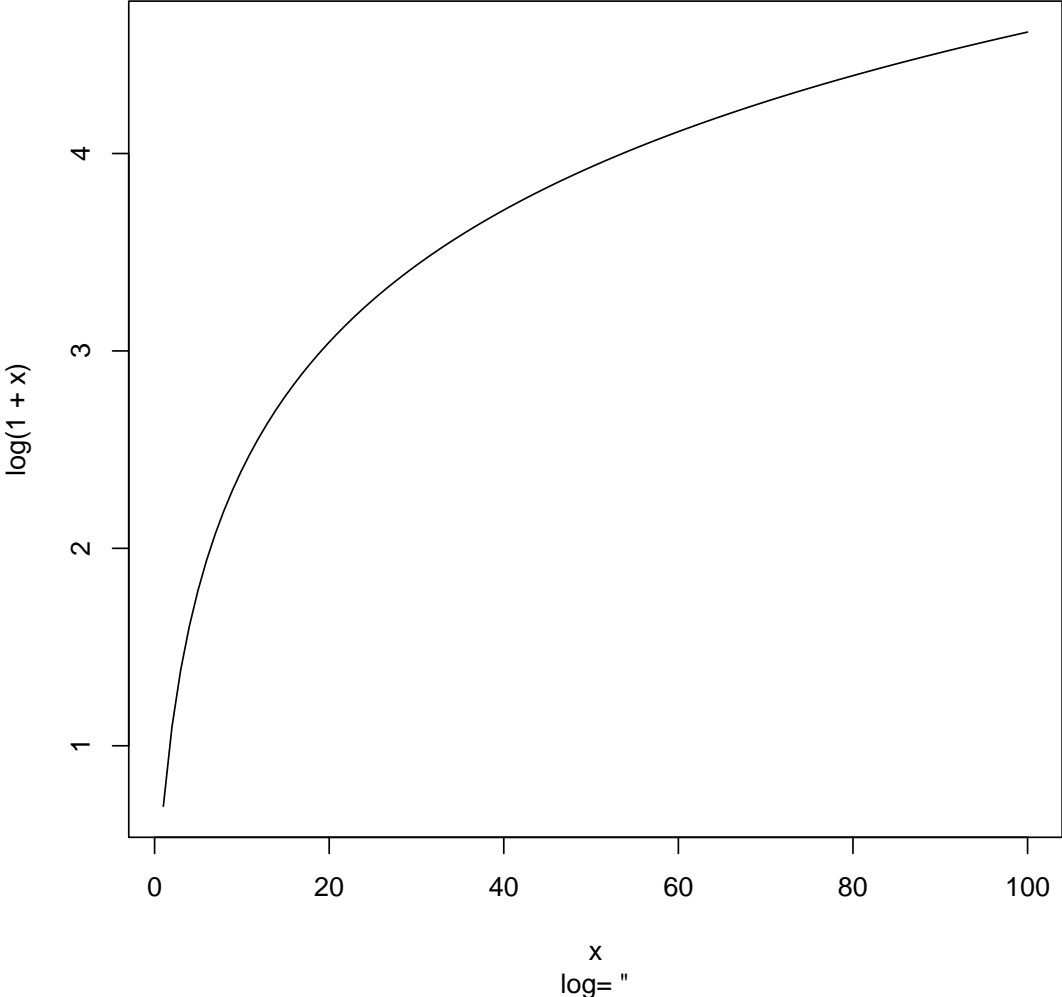
```
op <- par(mfrow=c(2,2))
curve(x^3-3*x, -2, 2)
curve(x^2-2, add = TRUE, col = "violet")

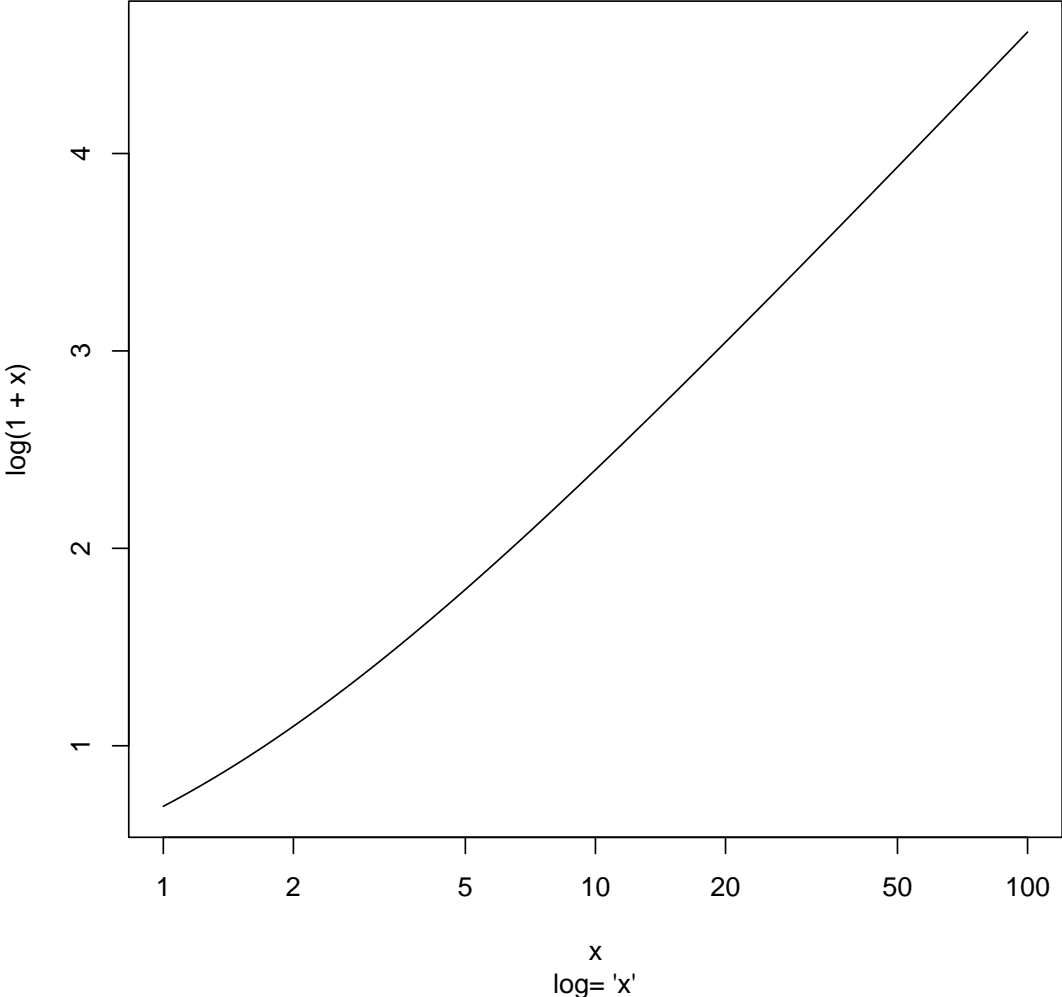
plot(cos, -pi, 3*pi)
plot(cos, xlim = c(-pi,3*pi), n = 1001, col = "blue", add=TRUE)

chippy <- function(x) sin(cos(x)*exp(-x/2))
curve(chippy, -8, 7, n=2001)
plot (chippy, -8, -5)
```

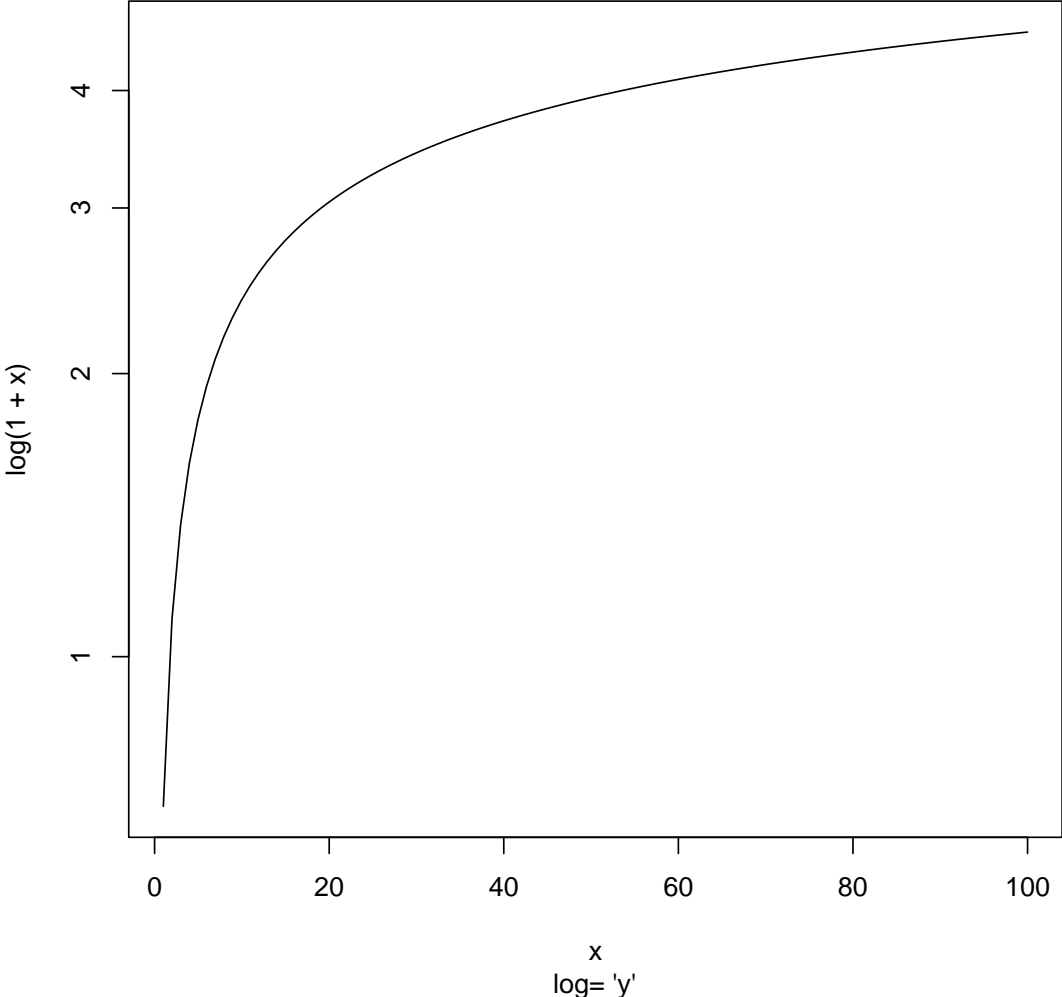


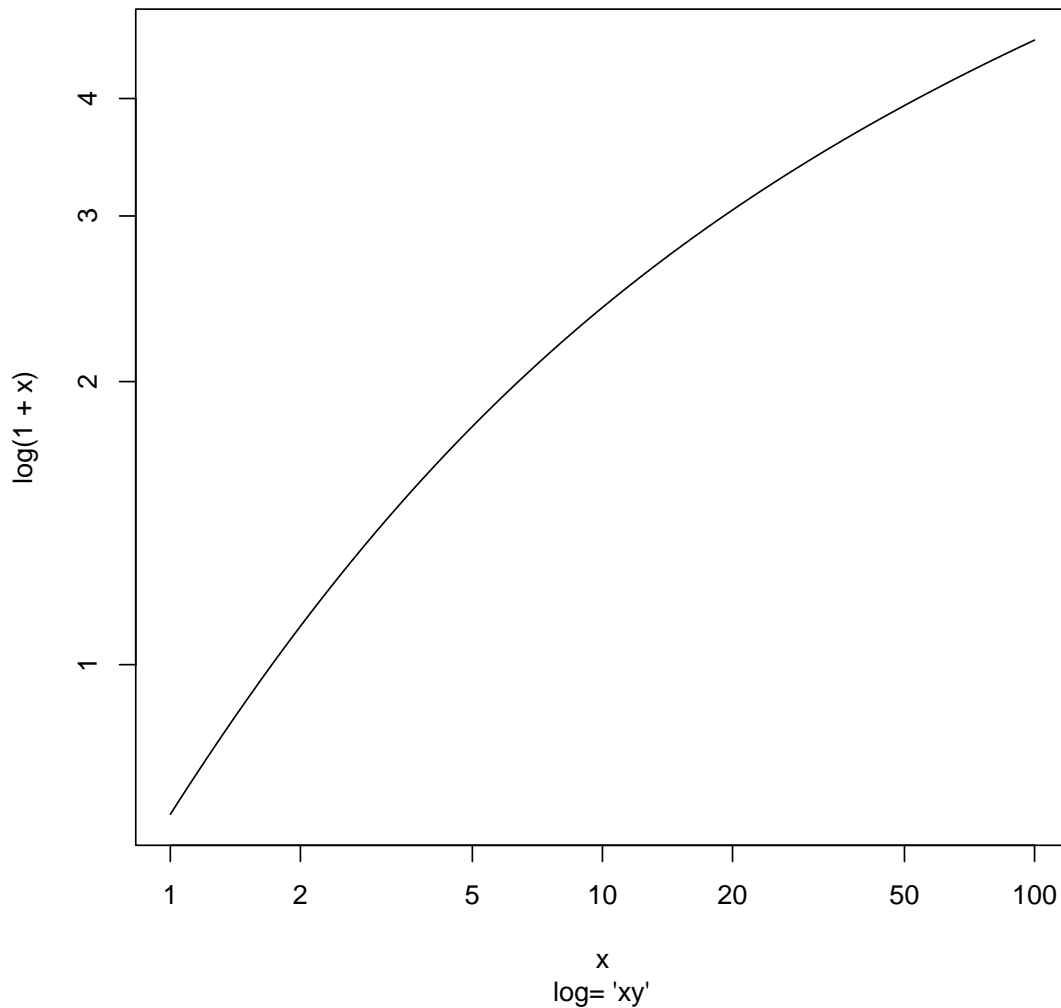
```
for(ll in c("", "x", "y", "xy"))  
  curve(log(1+x), 1, 100, log=ll, sub=paste("log= ", ll, "", sep=""))
```











```
par(op)
```

### 13 loess (regression non-paramétrique)

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

## $fit
##      1      2      3      4      5      6      7
## 7.797353 10.002308 12.499786 15.281082 18.446568 21.865315 25.517015
##      8      9     10     11     12     13     14
## 29.350386 33.230660 37.167935 41.205226 45.055736 48.355889 49.824812
```

```
##      15      16      17      18      19      20      21
## 51.986702 56.461318 61.959729 68.569313 76.316068 85.212121 95.324047
##      22      23      24      25      26
##      NA      NA      NA      NA      NA
##
## $se.fit
##      1      2      3      4      5      6      7      8
## 7.568120 5.945831 4.990827 4.545284 4.308639 4.115049 3.789542 3.716231
##      9      10     11     12     13     14     15     16
## 3.776947 4.091747 4.709568 4.245427 4.035929 3.753410 4.004705 4.043190
##      17     18     19     20     21     22     23     24
## 4.026105 4.074664 4.570818 5.954217 8.302014      NA      NA      NA
##      25     26
##      NA      NA
##
## $residual.scale
## [1] 15.29496
##
## $df
## [1] 44.6179
```

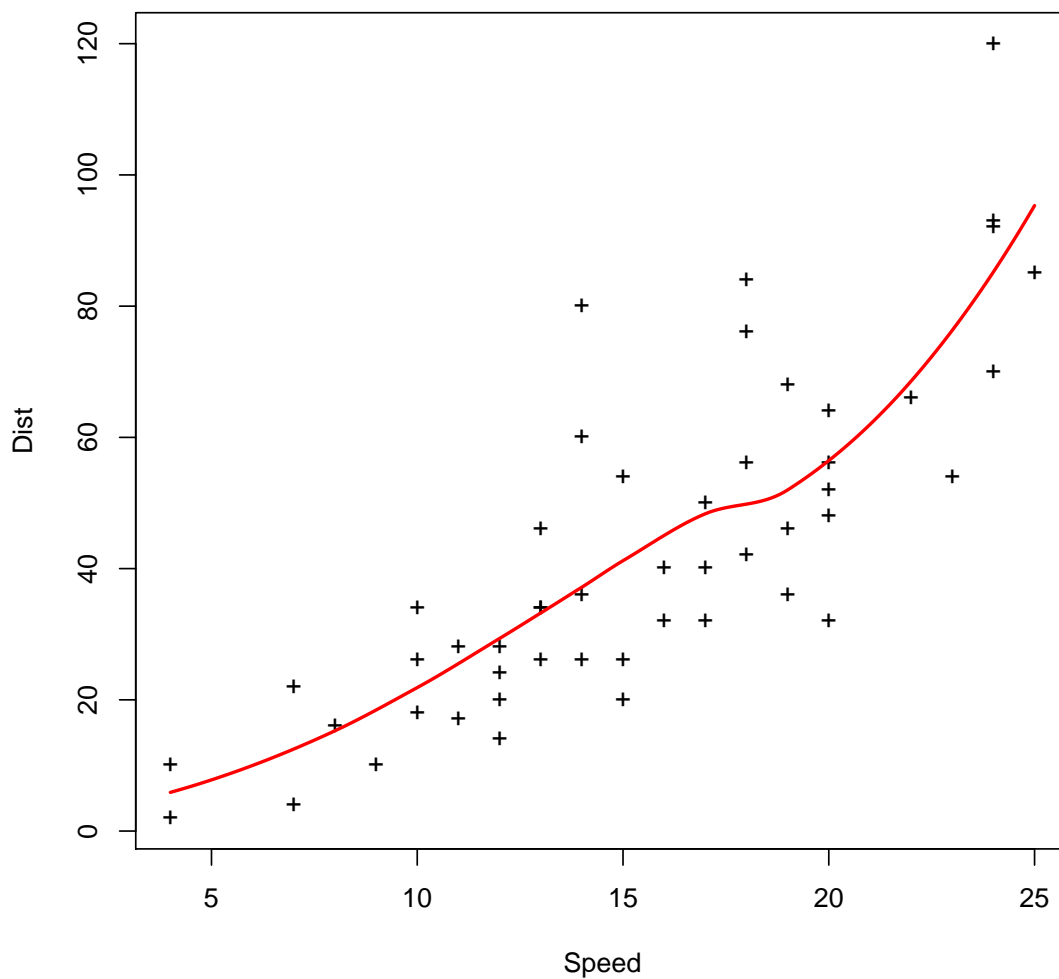
to allow extrapolation

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

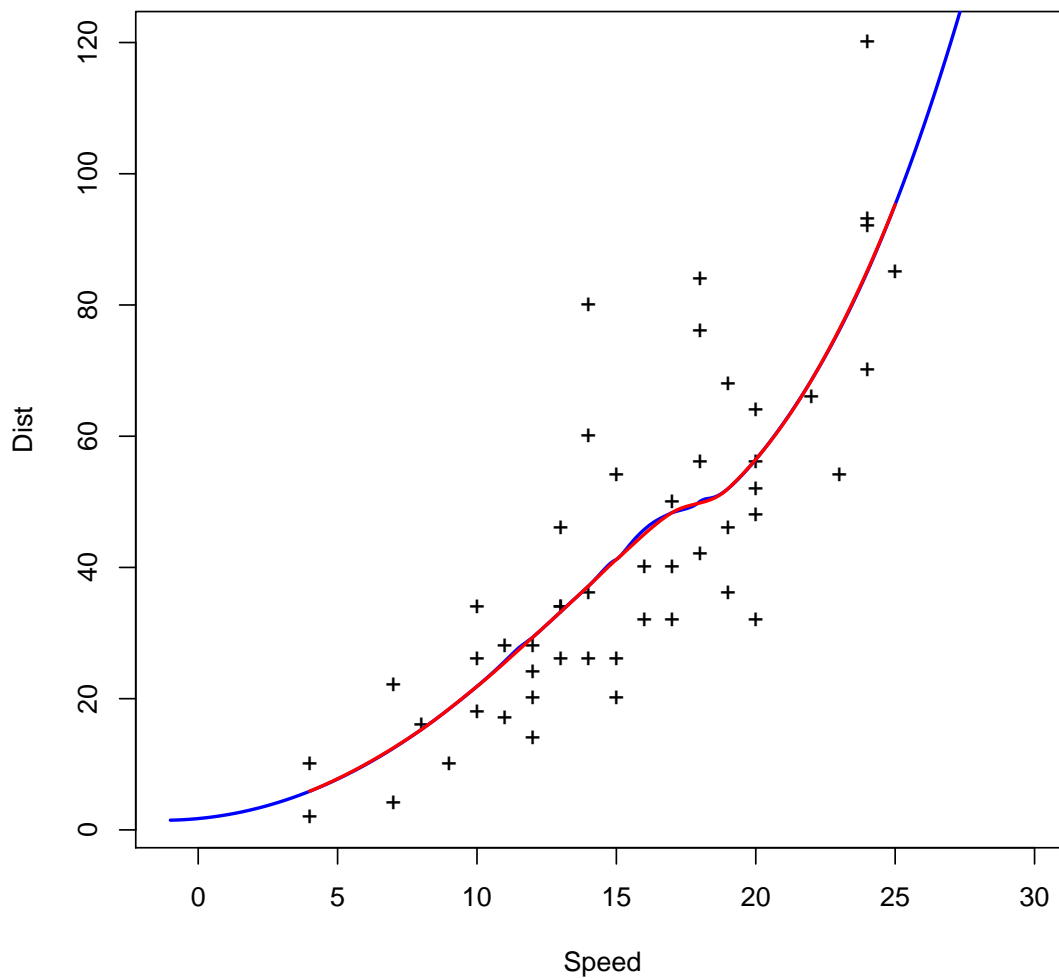
## $fit
##      1      2      3      4      5      6
## 7.741006 9.926596 12.442424 15.281082 18.425712 21.865315
##      7      8      9     10     11     12
## 25.713413 29.350386 33.230660 37.167935 41.205226 45.781544
##     13     14     15     16     17     18
## 48.355889 50.067148 51.986702 56.445263 62.025404 68.569313
##     19     20     21     22     23     24
## 76.193111 85.053364 95.300523 106.974661 120.092581 134.665851
##     25     26
## 150.698545 168.190283
##
## $se.fit
##      1      2      3      4      5      6      7
## 7.565991 5.959097 5.012013 4.550013 4.321596 4.119331 3.939804
##      8      9     10     11     12     13     14
## 3.720098 3.780877 4.096004 4.714469 4.398936 4.040129 4.184257
##     15     16     17     18     19     20     21
```

```
## 4.008873 4.061865 4.033998 4.078904 4.584606 5.952480 8.306901
##      22      23      24      25      26
## 11.601911 15.792480 20.864660 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
```

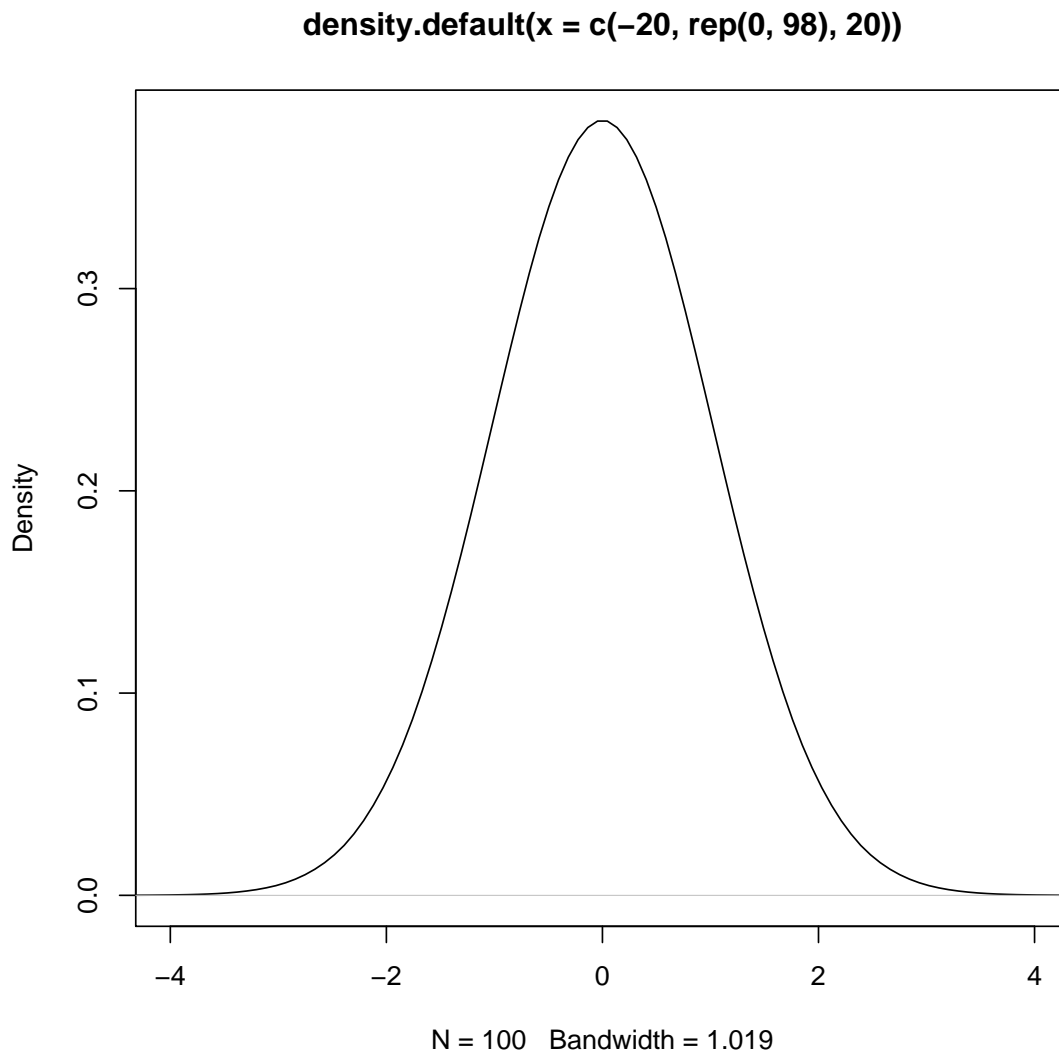


```
plot(cars.lo2, xlab="Speed", ylab="Dist", pch="+", xlim=c(min(cars$speed)-5,max(car
lines(seq(min(cars$speed)-5, max(cars$speed)+5, 0.1),predict(cars.lo2, data.frame(s
lines(seq(min(cars$speed)-5, max(cars$speed)+5, 0.1),predict(cars.lo, data.frame(sp
```



## 14 density estimation

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

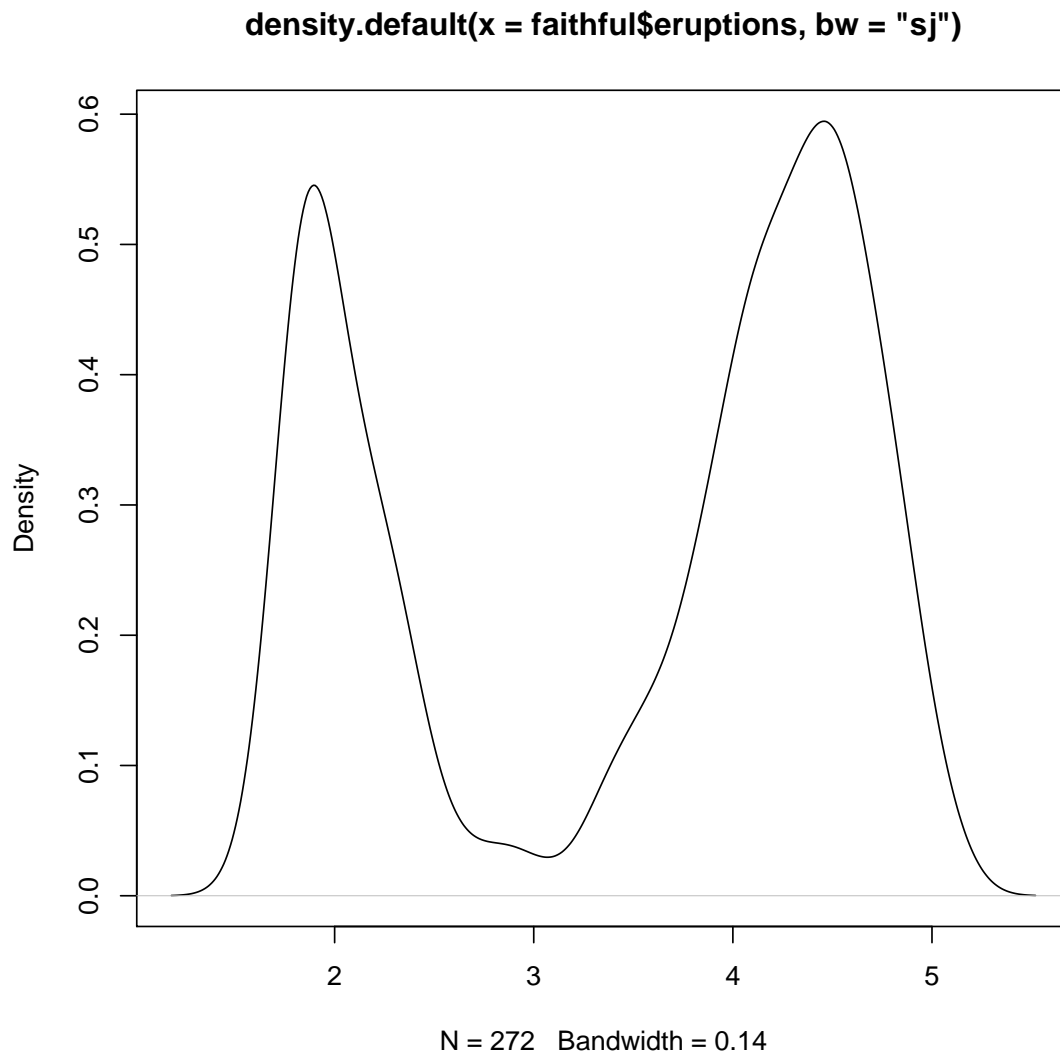


```
# The Old Faithful geyser data
d <- density(faithful$eruptions, bw = "sj")
d

##
## 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
```

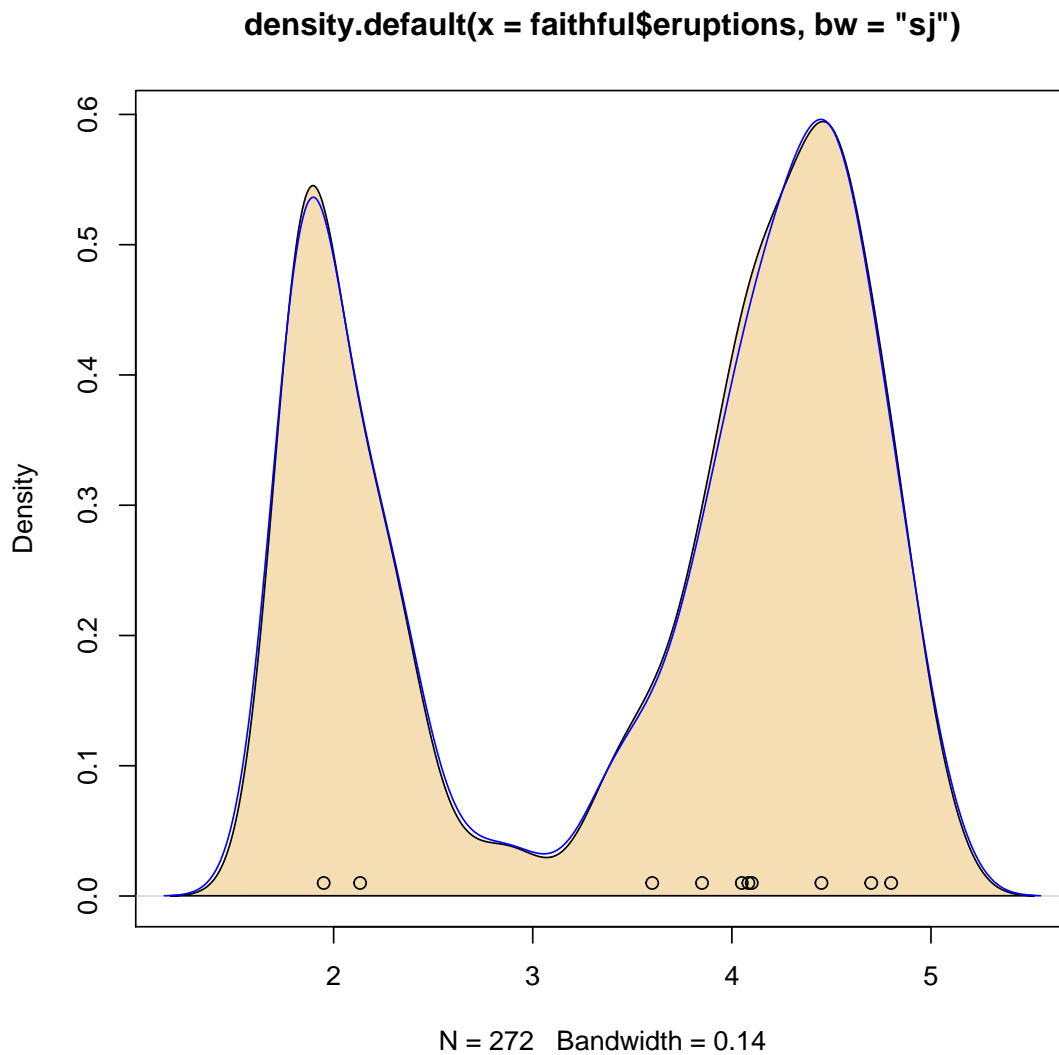
```
## Mean :3.350 Mean :0.2301726
## 3rd Qu.:4.435 3rd Qu.:0.4134348
## Max. :5.520 Max. :0.5945634
```

```
plot(d)
```



```
plot(d, type = "n")
polygon(d, col = "wheat")
## Missing values:
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))
```



```
## Weighted observations:
fe <- sort(faithful$eruptions) # has quite a few non-unique values
## use 'counts / n' as weights:
dw <- density(unique(fe), weights = table(fe)/length(fe), bw = d$bw)
utils::str(dw) ## smaller n: only 126, but identical estimate:

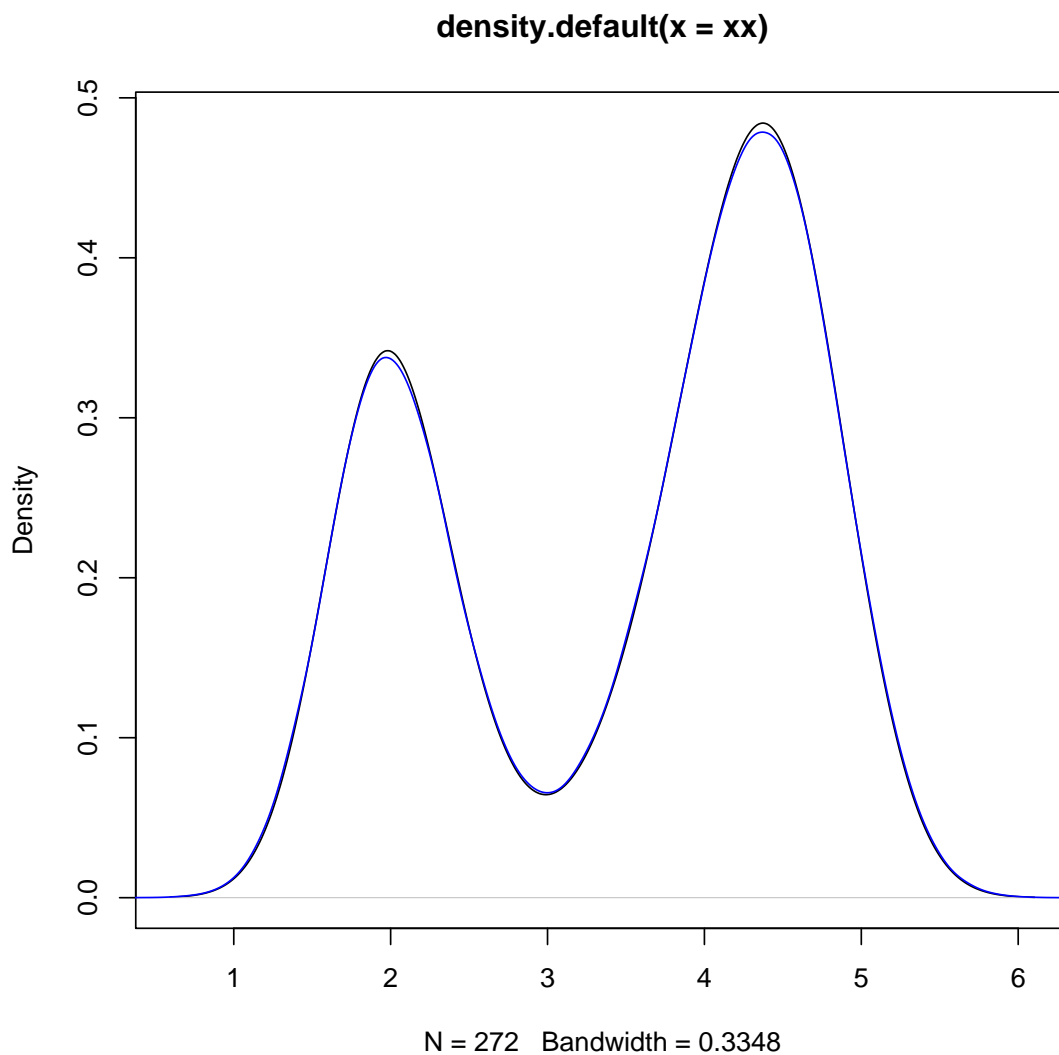
## List of 7
## $ x      : num [1:512] 1.18 1.19 1.2 1.21 1.21 ...
## $ y      : num [1:512] 0.000183 0.000223 0.00027 0.000328 0.000397 ...
```



```
## $ bw      : num 0.14
## $ n      : int 126
## $ call    : language density.default(x = unique(fe), bw = d$bw, weights = tabl
## $ data.name: chr "unique(fe)"
## $ has.na  : logi FALSE
## - attr(*, "class")= chr "density"

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

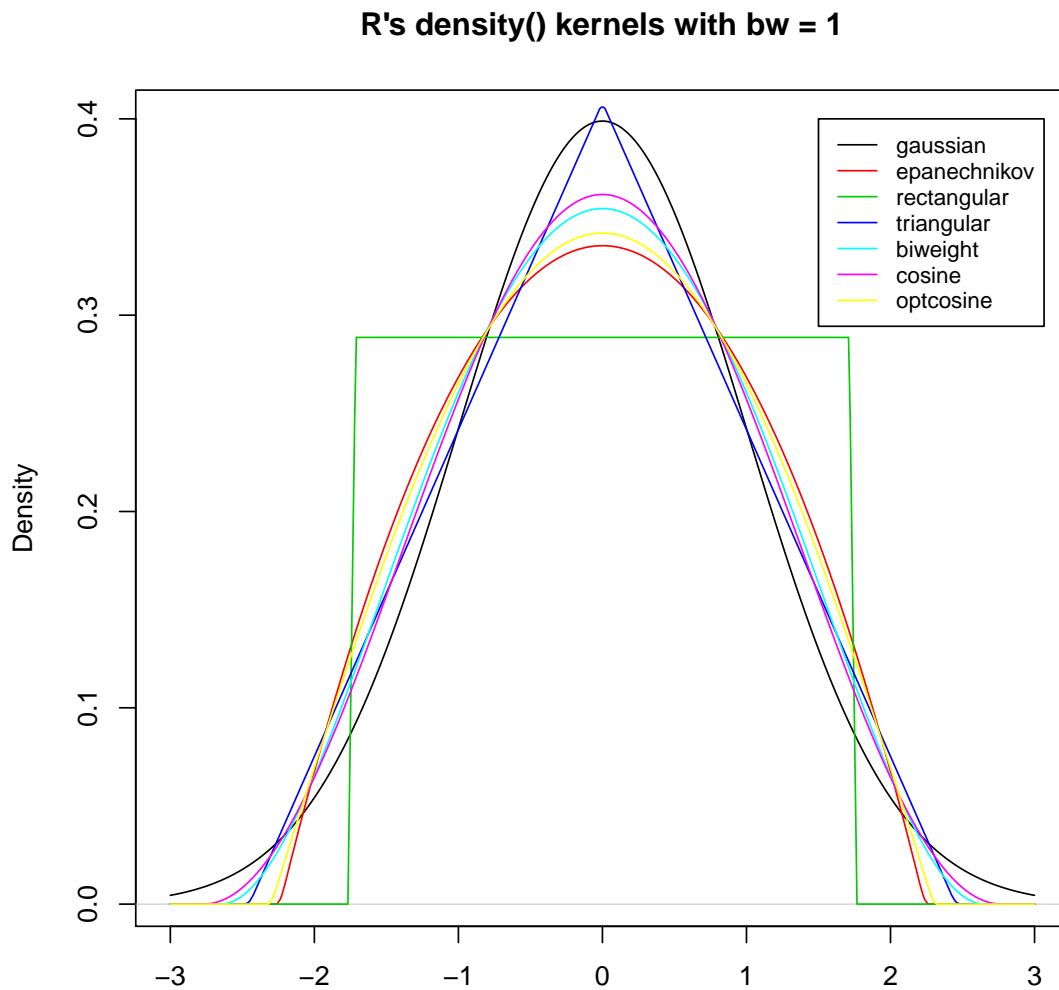
```
## simulation from a density() fit:
# a kernel density fit is an equally-weighted mixture.
fit <- density(xx)
N <- 1e6
x.new <- rnorm(N, sample(xx, size = N, replace = TRUE), fit$bw)
plot(fit)
lines(density(x.new), col="blue")
```



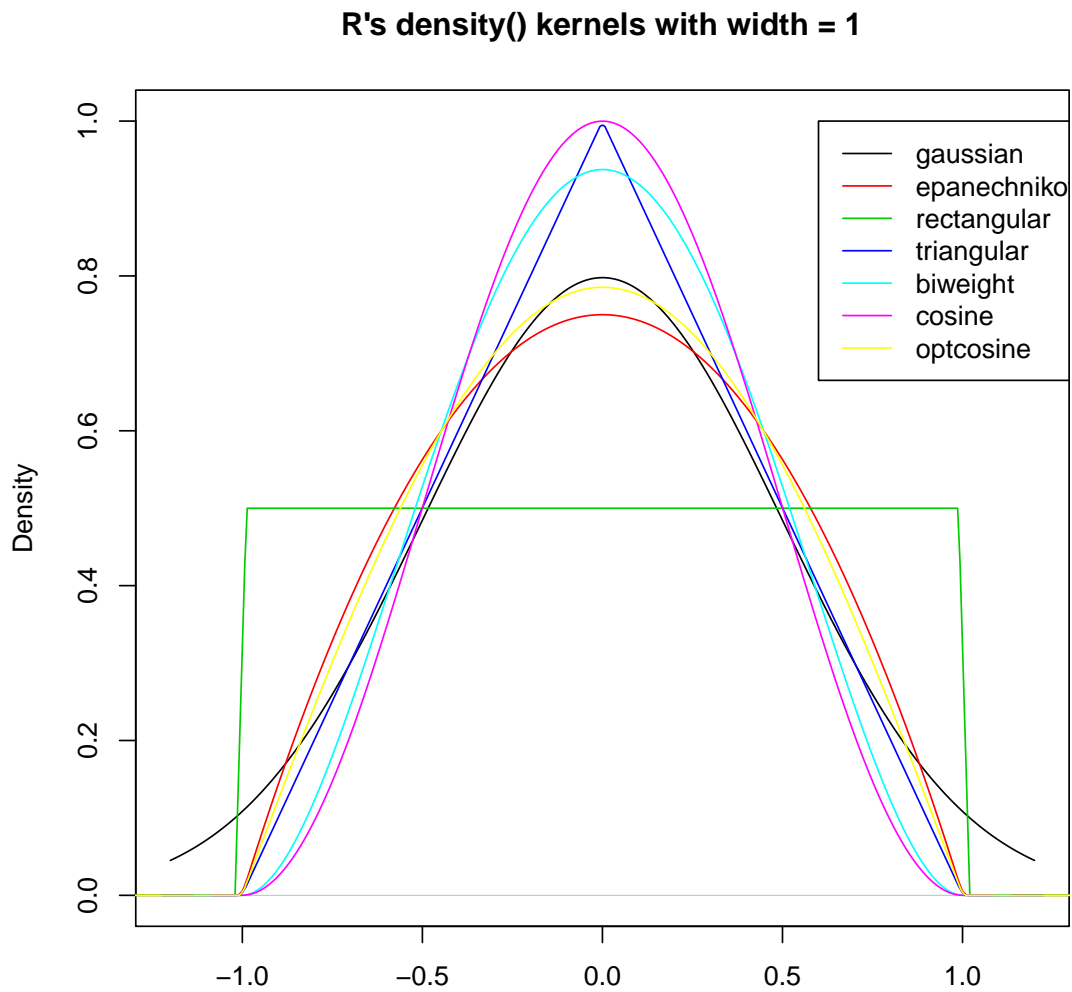
```
(kernels <- eval(formals(density.default)$kernel))
```

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

```
## show the kernels in the R parametrization
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,.4, legend = kernels, col = seq(kernels),
      lty = 1, cex = .8, y.intersp = 1)
```



```
## show the kernels in the S parametrization
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, 1.0, legend = kernels, col = seq(kernels), lty = 1)
```



```
##----- Semi-advanced theoretic from here on -----
(RKs <- cbind(sapply(kernels,
                    function(k) density(kernel = k, give.Rkern = TRUE))))

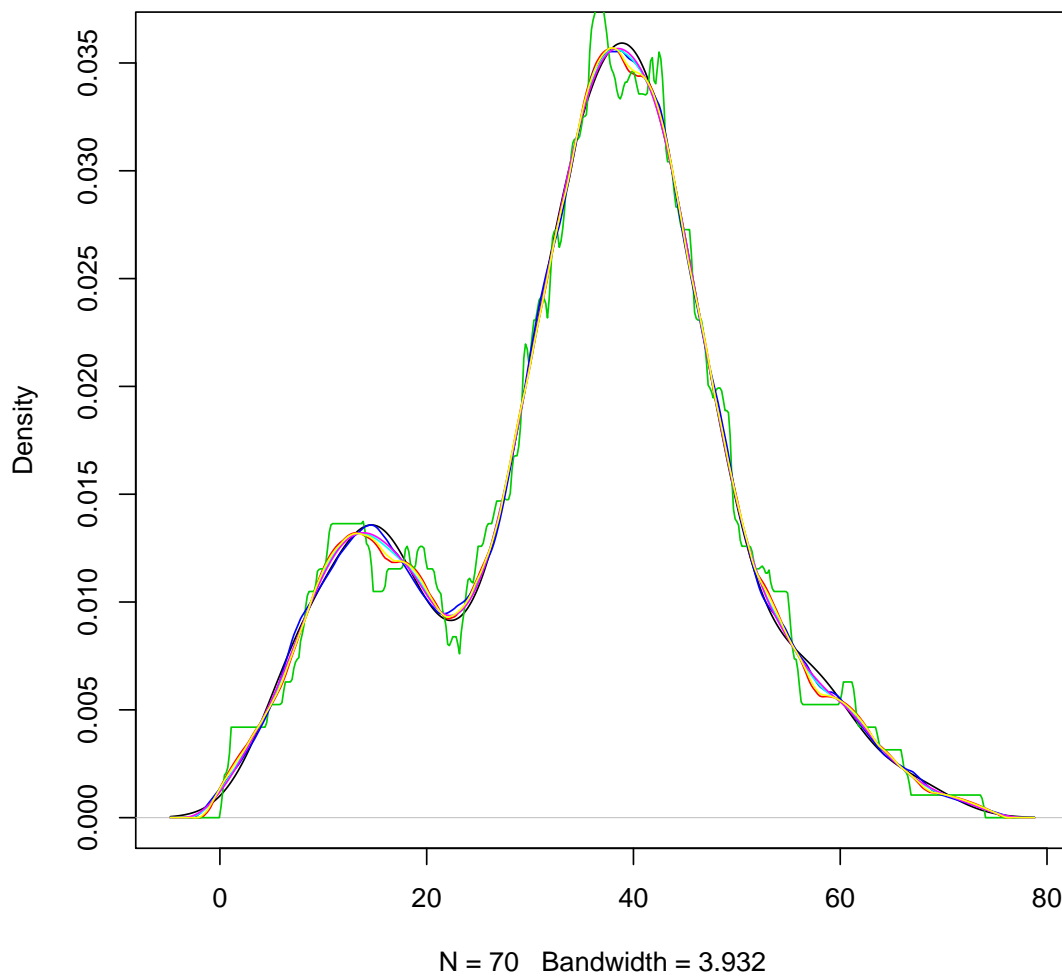
##           [,1]
## gaussian   0.2820948
## epanechnikov 0.2683282
## rectangular 0.2886751
## triangular  0.2721655
## biweight    0.2699746
## cosine      0.2711340
## optcosine   0.2684756

100*round(RKs["epanechnikov",]/RKs, 4) ## Efficiencies
```

```
##           [,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) ## sensible automatic choice
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)
```

same sd bandwidths, 7 different kernels

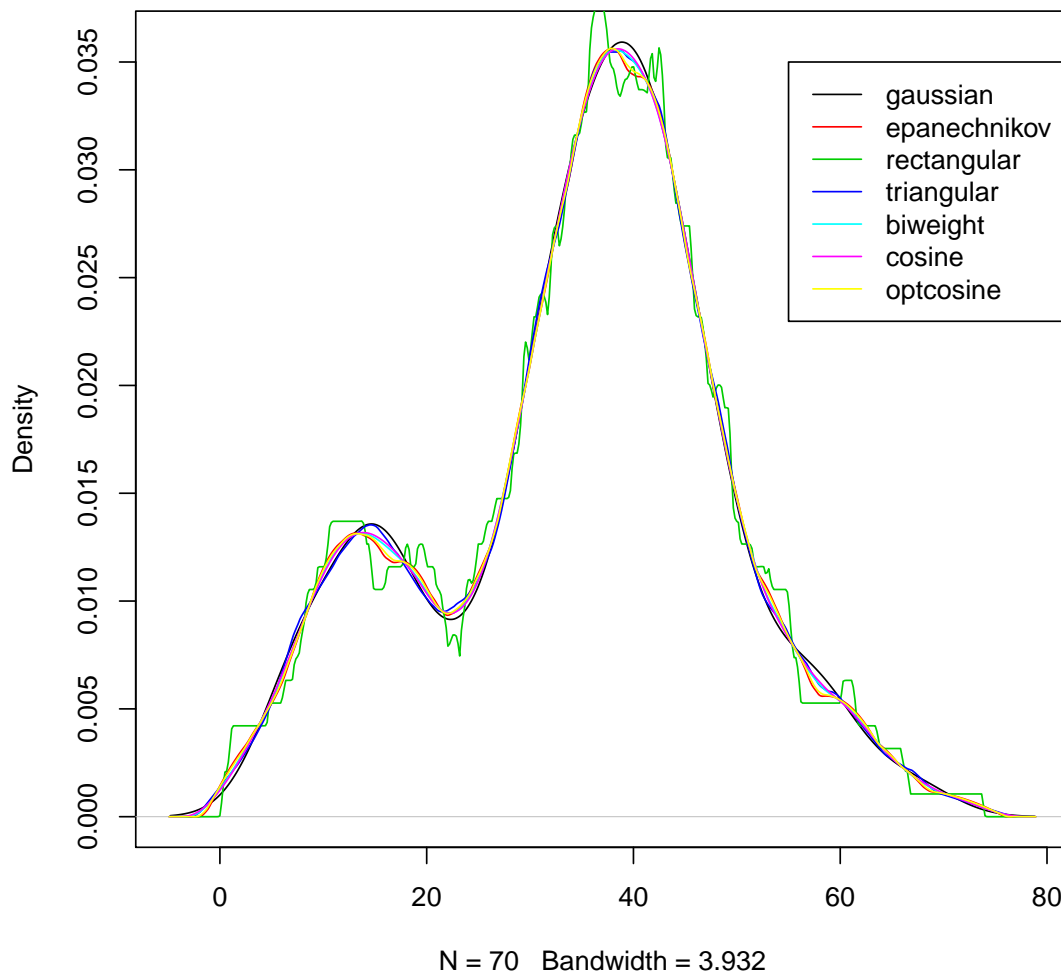


```
## Bandwidth Adjustment for "Exactly Equivalent Kernels"
h.f <- sapply(kernels, function(k) density(kernel = k, give.Rkern = TRUE))
(h.f <- (h.f["gaussian"] / h.f)^ .2)

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

## -> 1, 1.01, .995, 1.007, ... close to 1 => adjustment barely visible..
```

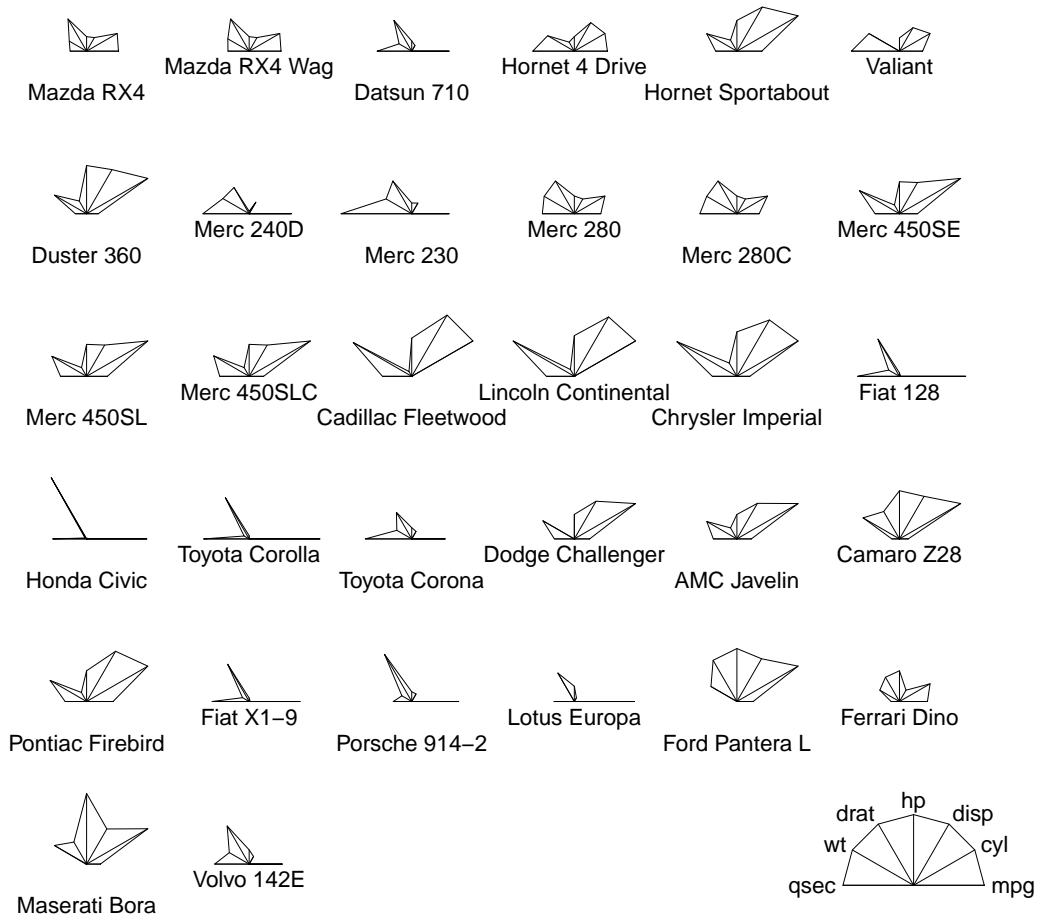
```
plot(density(precip, bw = bw),
     main = "equivalent bandwidths, 7 different kernels")
for(i in 2:length(kernels))
  lines(density(precip, bw = bw, adjust = h.f[i], kernel = kernels[i]),
       col = i)
legend(55, 0.035, legend = kernels, col = seq(kernels), lty = 1)
```

**equivalent bandwidths, 7 different kernels**

## 15 Radar plots

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

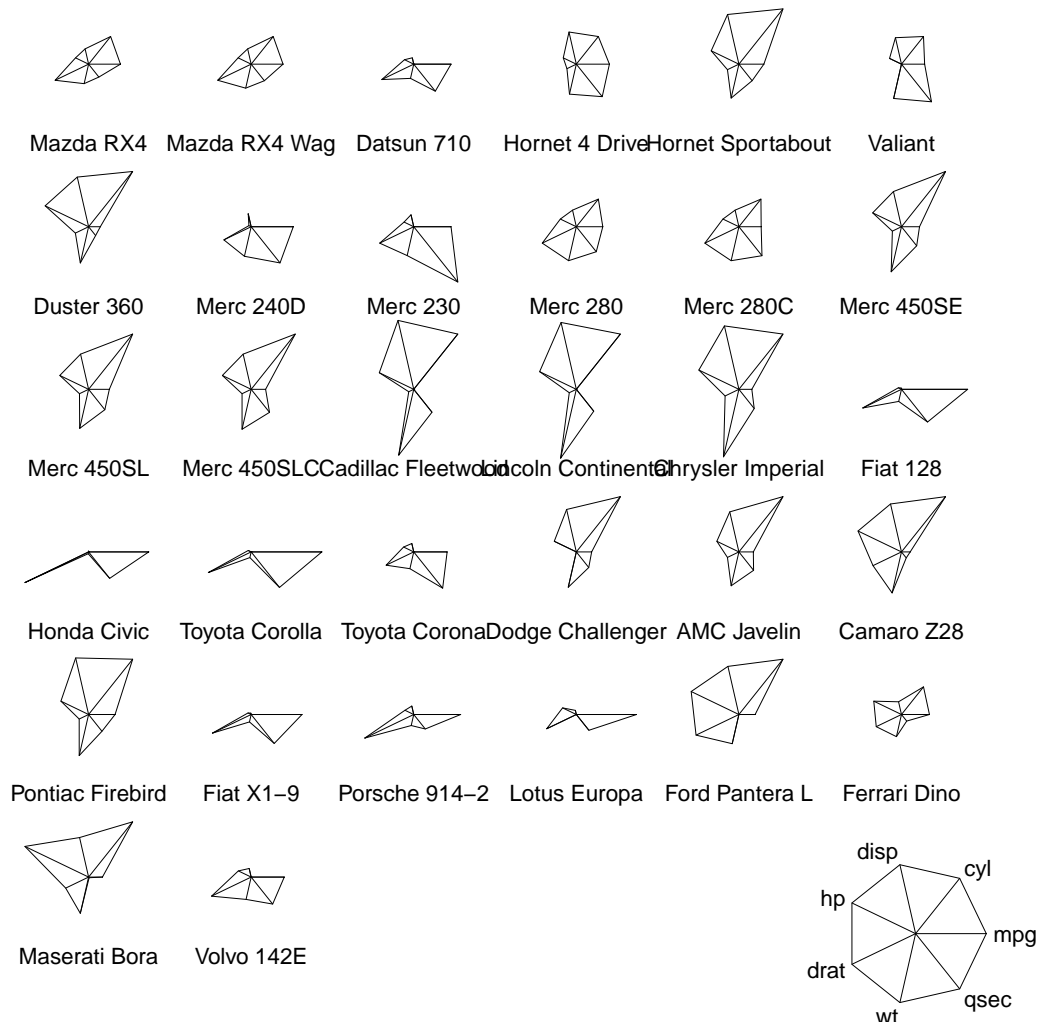
### Motor Trend Cars : stars(\*, full = F)



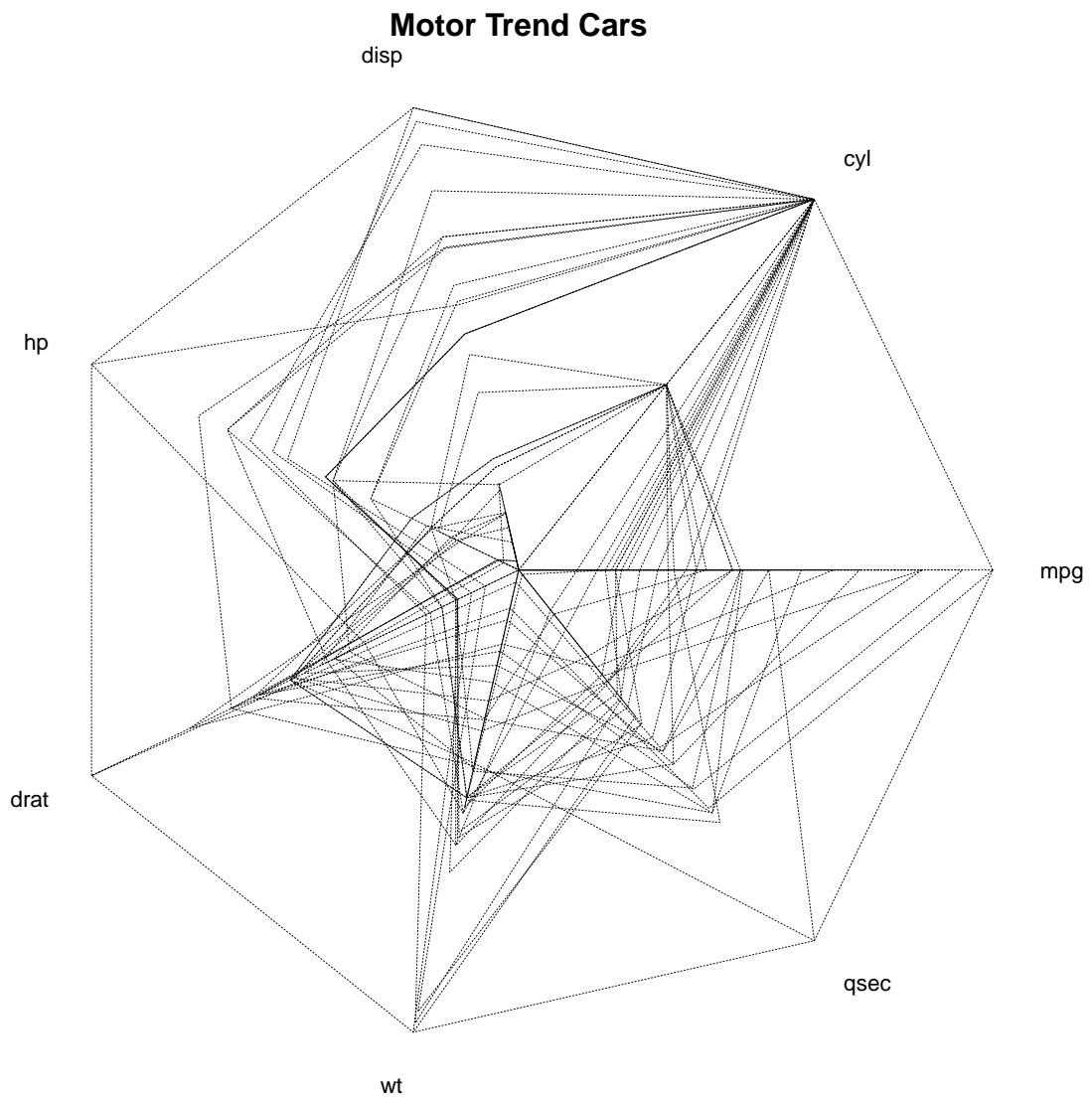
```
stars(mtcars[, 1:7], key.loc = c(14, 1.5),
      main = "Motor Trend Cars : full stars()", flip.labels=FALSE)
```



## Motor Trend Cars : full stars()

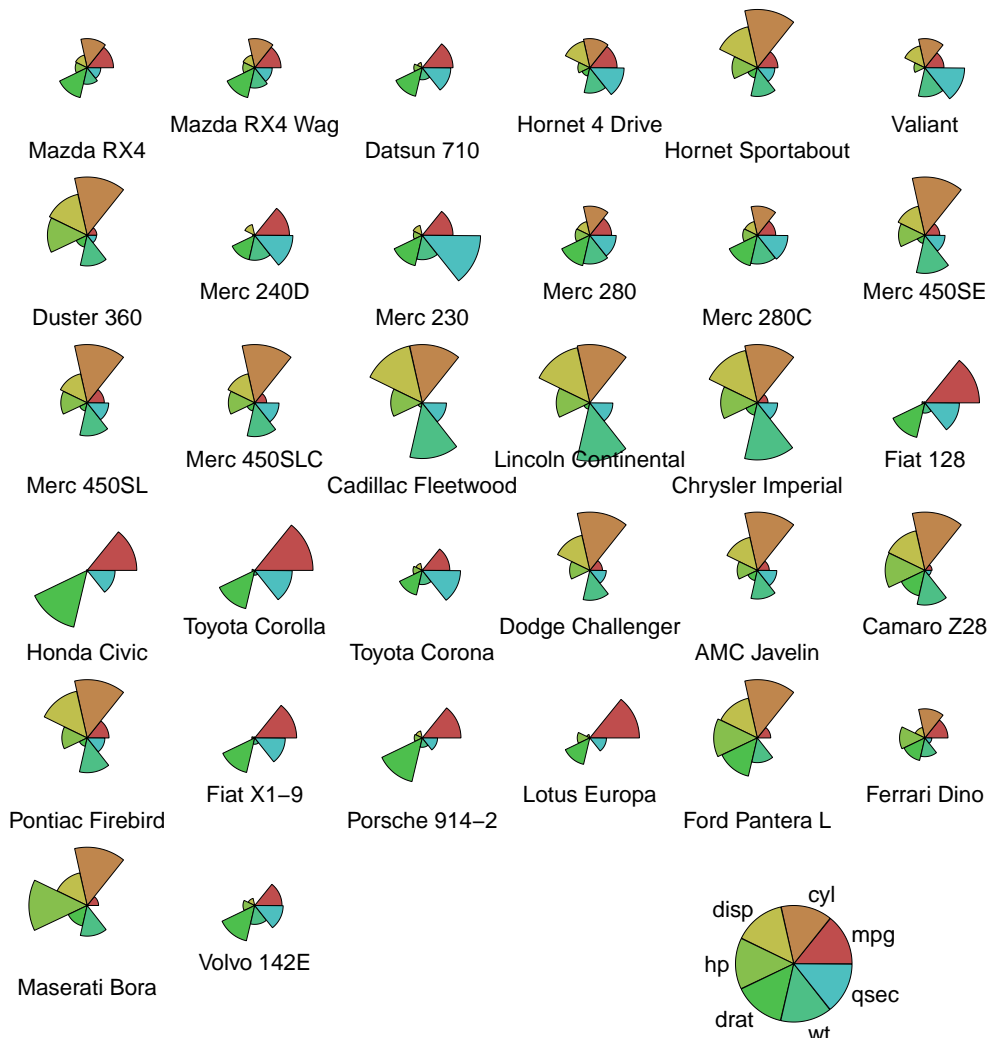


```
## 'Spider' or 'Radar' plot:
stars(mtcars[, 1:7], locations = c(0,0), radius = FALSE,
      key.loc=c(0,0), main="Motor Trend Cars", lty = 2)
```



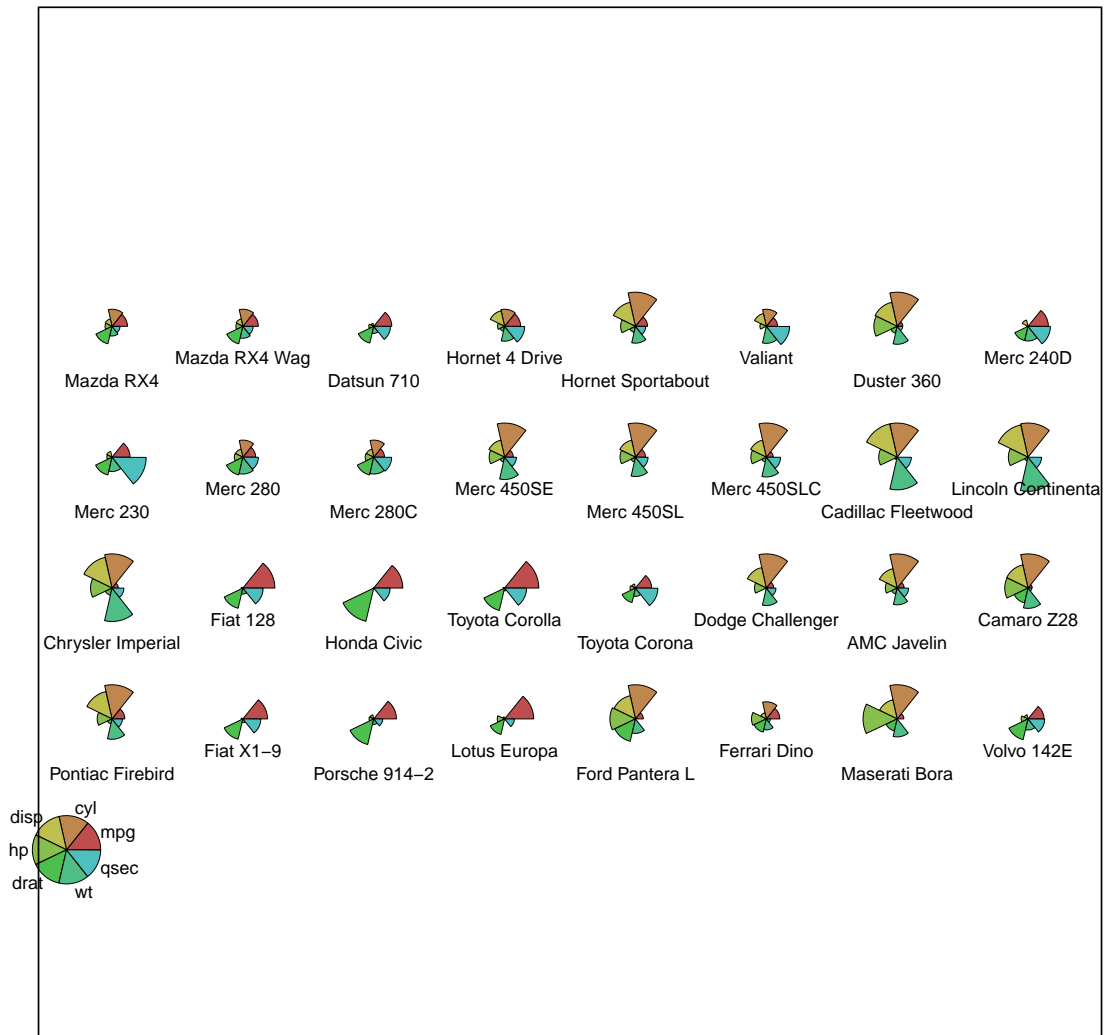
```
## Segment Diagrams:  
palette(rainbow(12, s = 0.6, v = 0.75))  
stars(mtcars[, 1:7], len = 0.8, key.loc = c(12, 1.5),  
      main = "Motor Trend Cars", draw.segments = TRUE)
```

### Motor Trend Cars



```
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 = .7)
```

## Motor Trend Cars

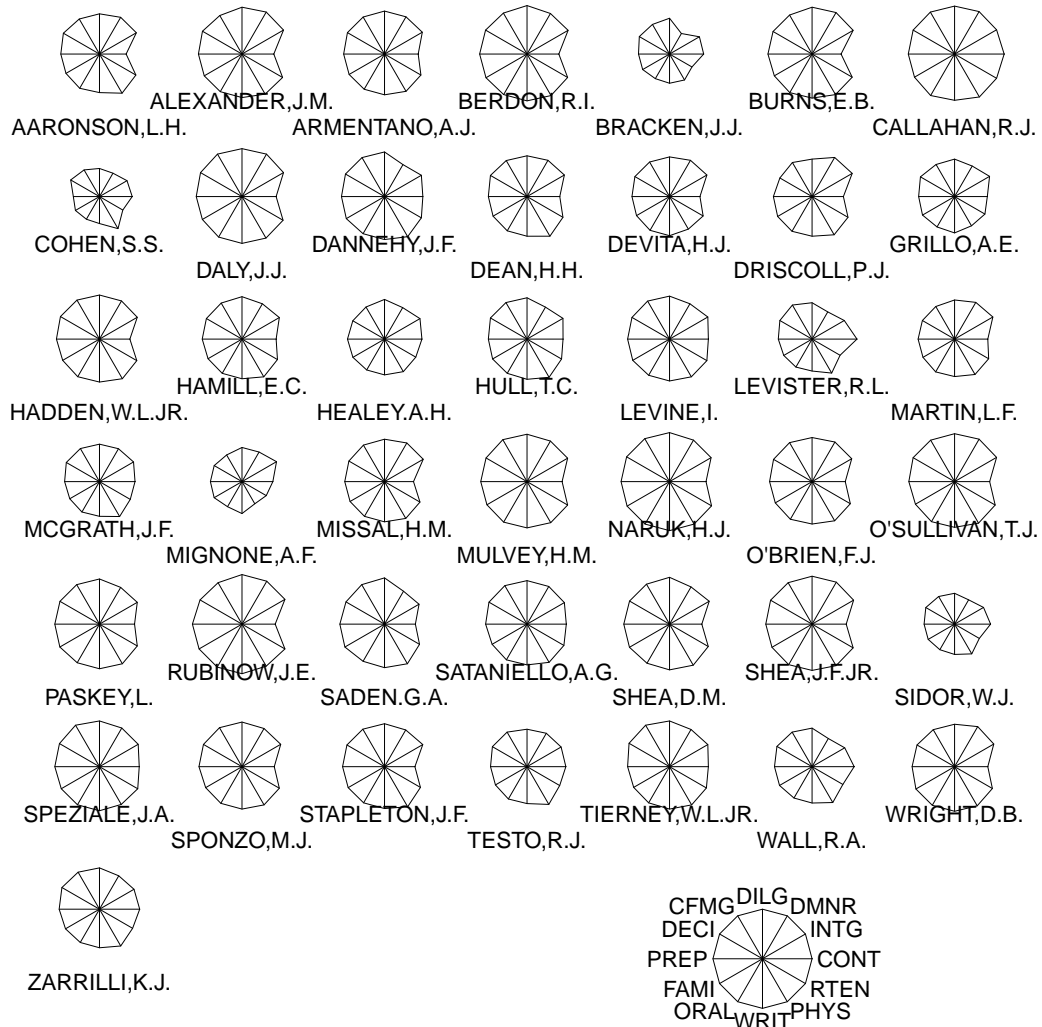


```

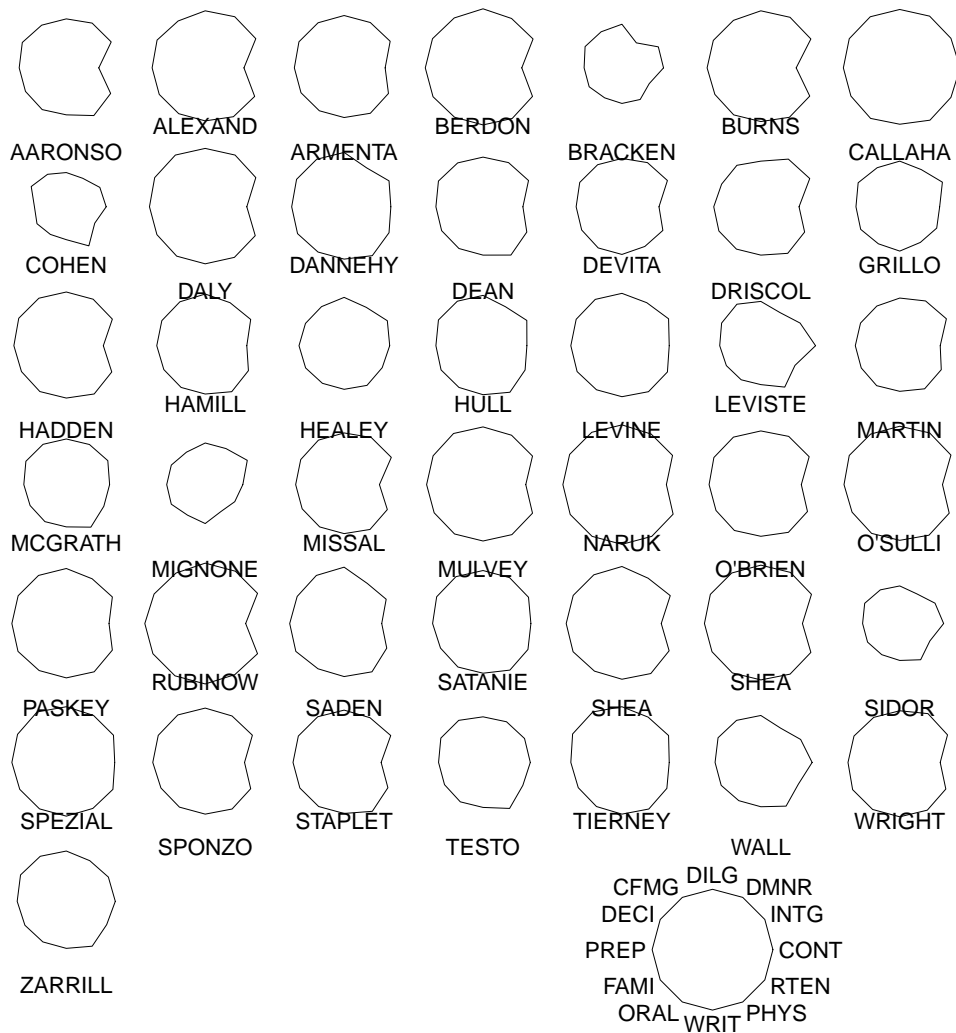
## scale linearly (not affinely) to [0, 1]
USJudge <- apply(USJudgeRatings, 2, function(x) x/max(x))
Jnam <- row.names(USJudgeRatings)
Snam <- abbreviate(substring(Jnam,1,regexr("[,\\.]",Jnam) - 1), 7)
stars(USJudge, labels = Jnam, scale = FALSE,
      key.loc = c(13, 1.5), main = "Judge not ...", len = 0.8)

```

## Judge not ...

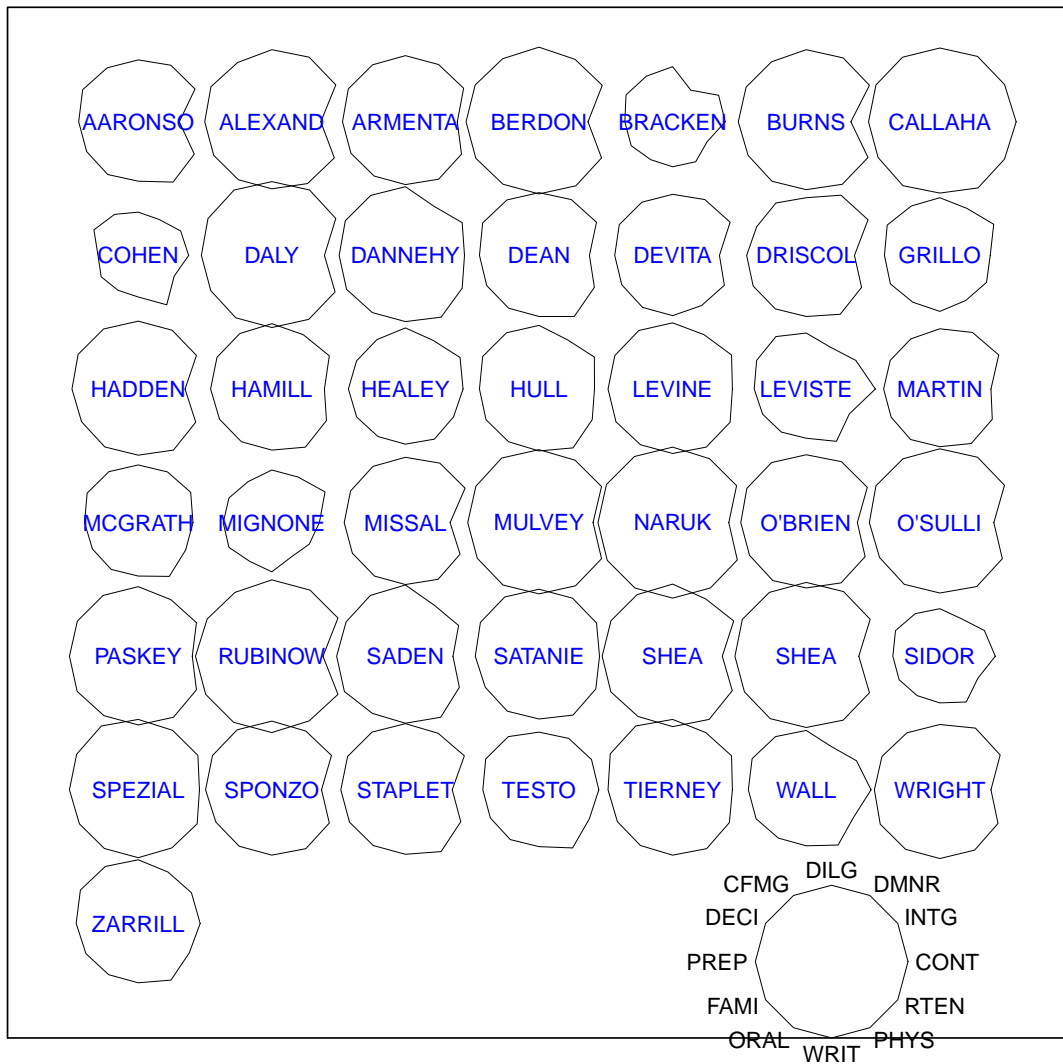


```
stars(USJudge, labels = Snam, scale = FALSE,  
      key.loc = c(13, 1.5), radius = FALSE)
```



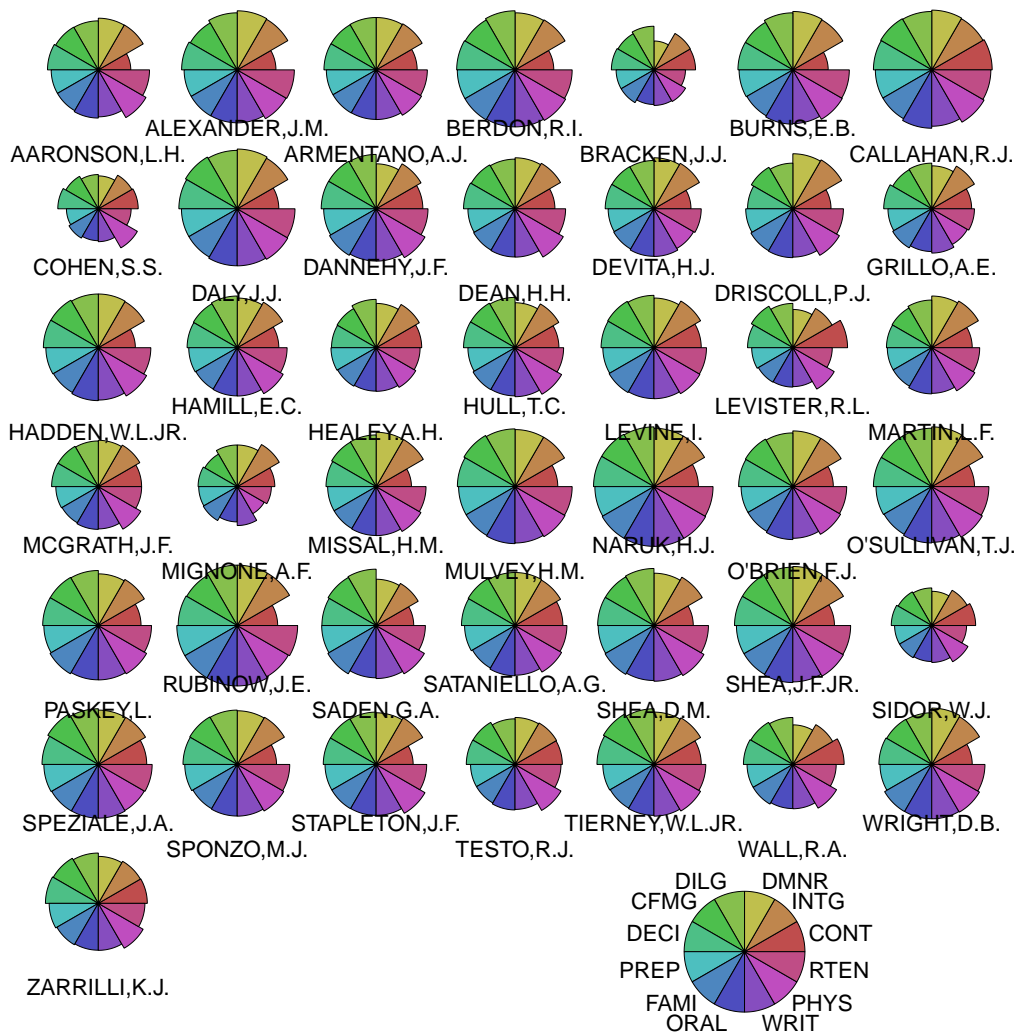
```
loc <- stars(USJudge, labels = NULL, scale = FALSE,
            radius = FALSE, frame.plot = TRUE,
            key.loc = c(13, 1.5), main = "Judge not ...", len = 1.2)
text(loc, Snam, col = "blue", cex = 0.8, xpd = TRUE)
```

## Judge not ...



```
## 'Segments':
```

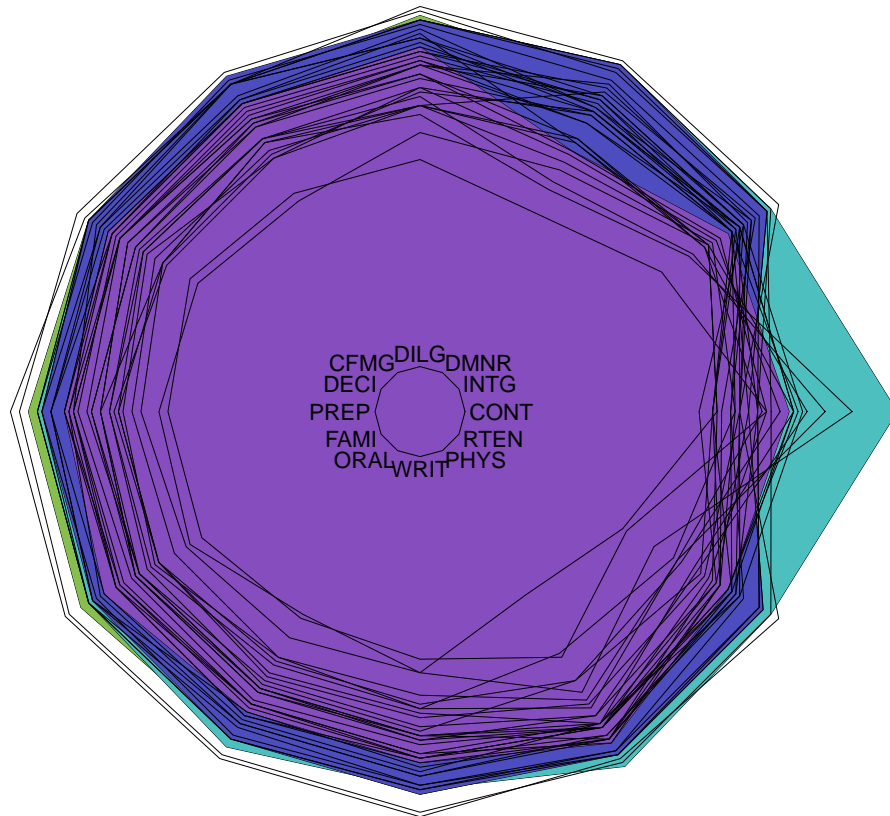
```
stars(USJudge, draw.segments = TRUE, scale = FALSE, key.loc = c(13,1.5))
```



```
## 'Spider':
```

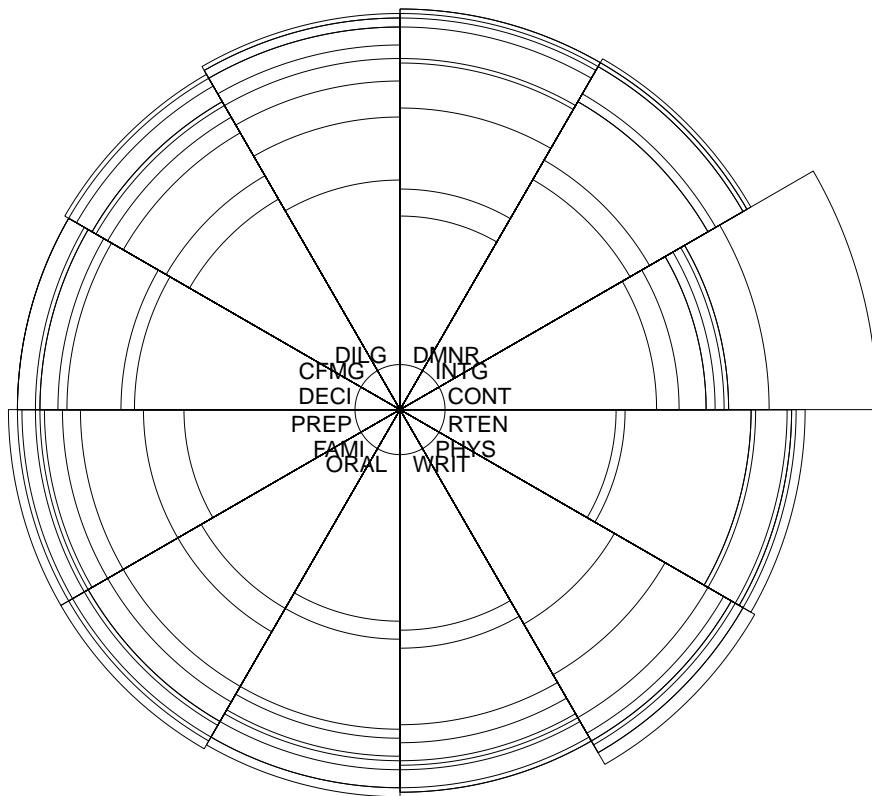
```
stars(USJudgeRatings, locations=c(0,0), scale=FALSE, radius = FALSE,
      col.stars=1:10, key.loc = c(0,0), main="US Judges rated")
```



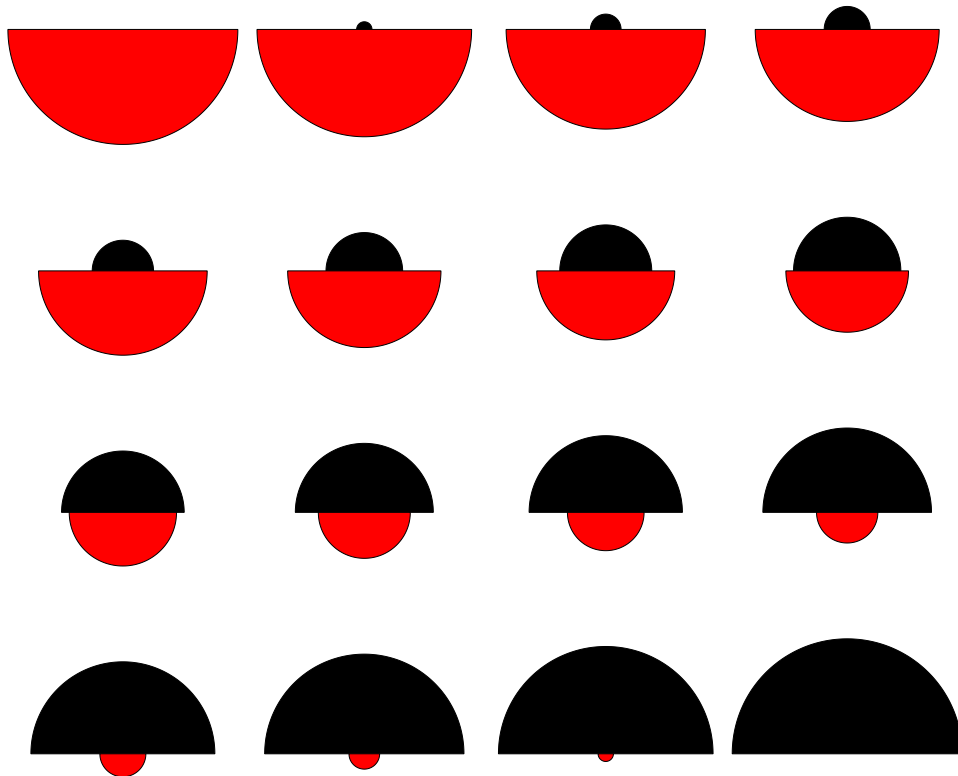
**US Judges rated**

```
## 'Radar-Segments'  
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 ")
```

## 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!")
```

**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 | 000000000000000000000000000000000111111222338  
## 2 | 07  
## 4 | 5  
## 6 | 8  
## 8 | 4  
## 10 | 5
```

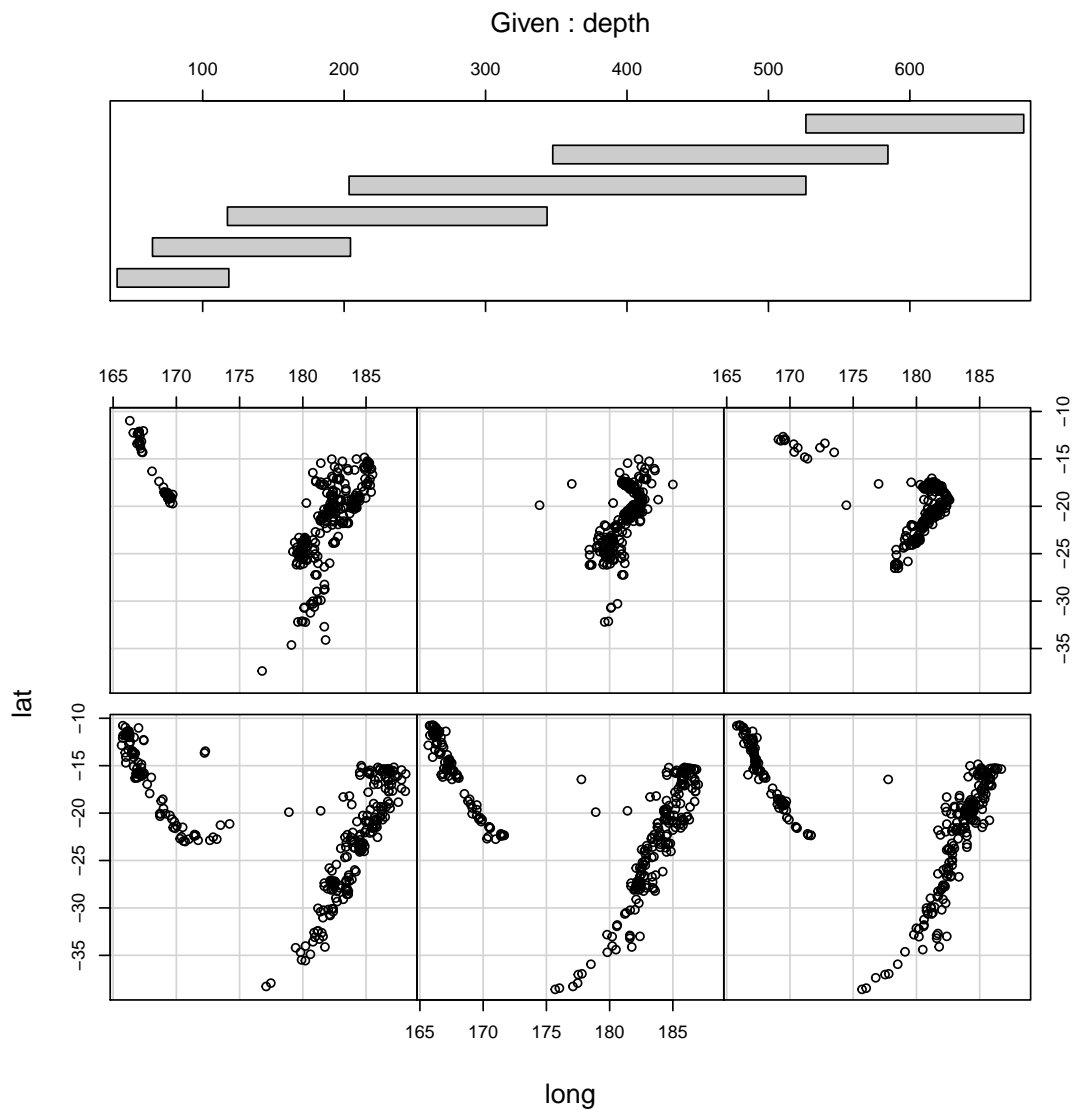
```
## 12 |
## 14 |
## 16 | 0

stem(log10(islands))

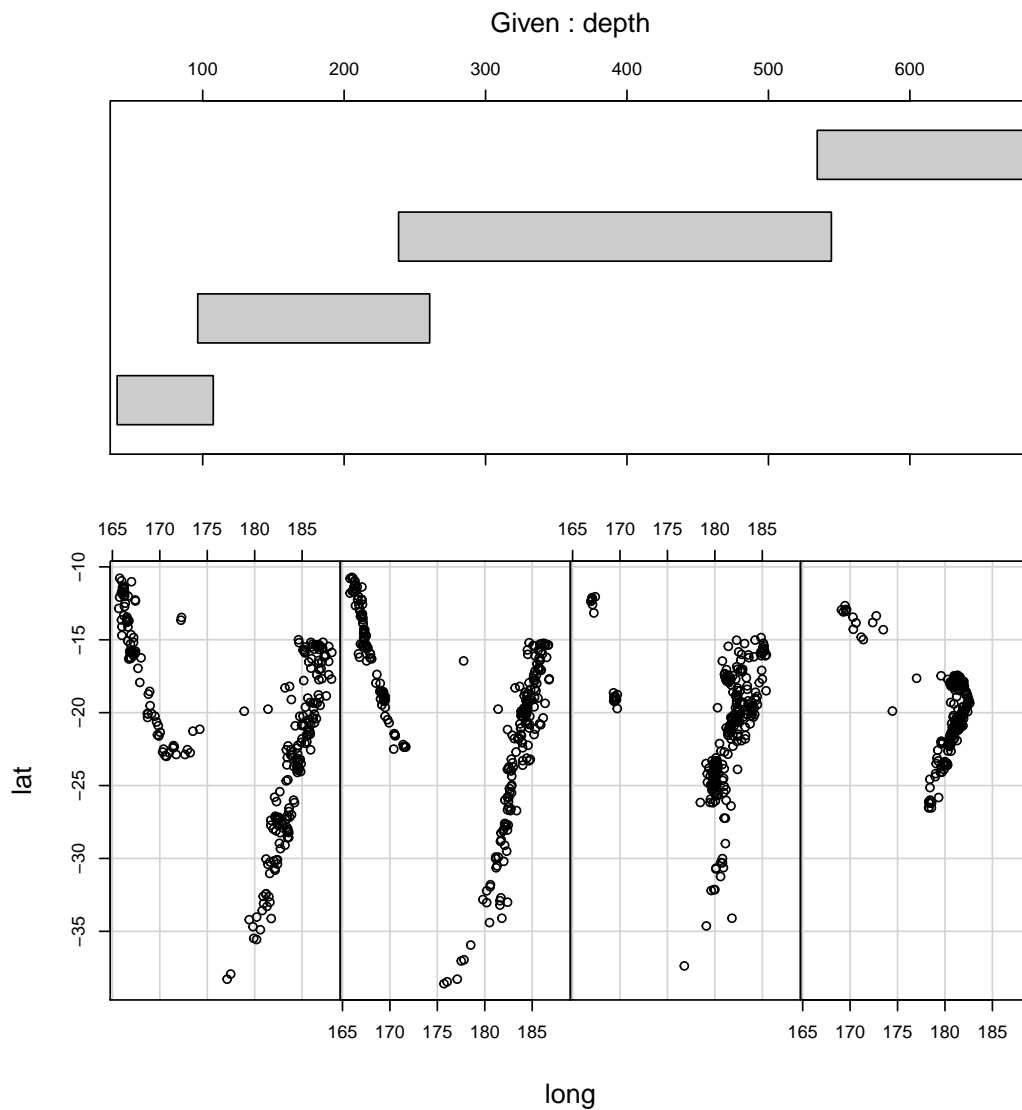
##
## The decimal point is at the |
##
## 1 | 1111112222233444
## 1 | 5555556666667899999
## 2 | 3344
## 2 | 59
## 3 |
## 3 | 5678
## 4 | 012
```

## 17 Conditioning plots

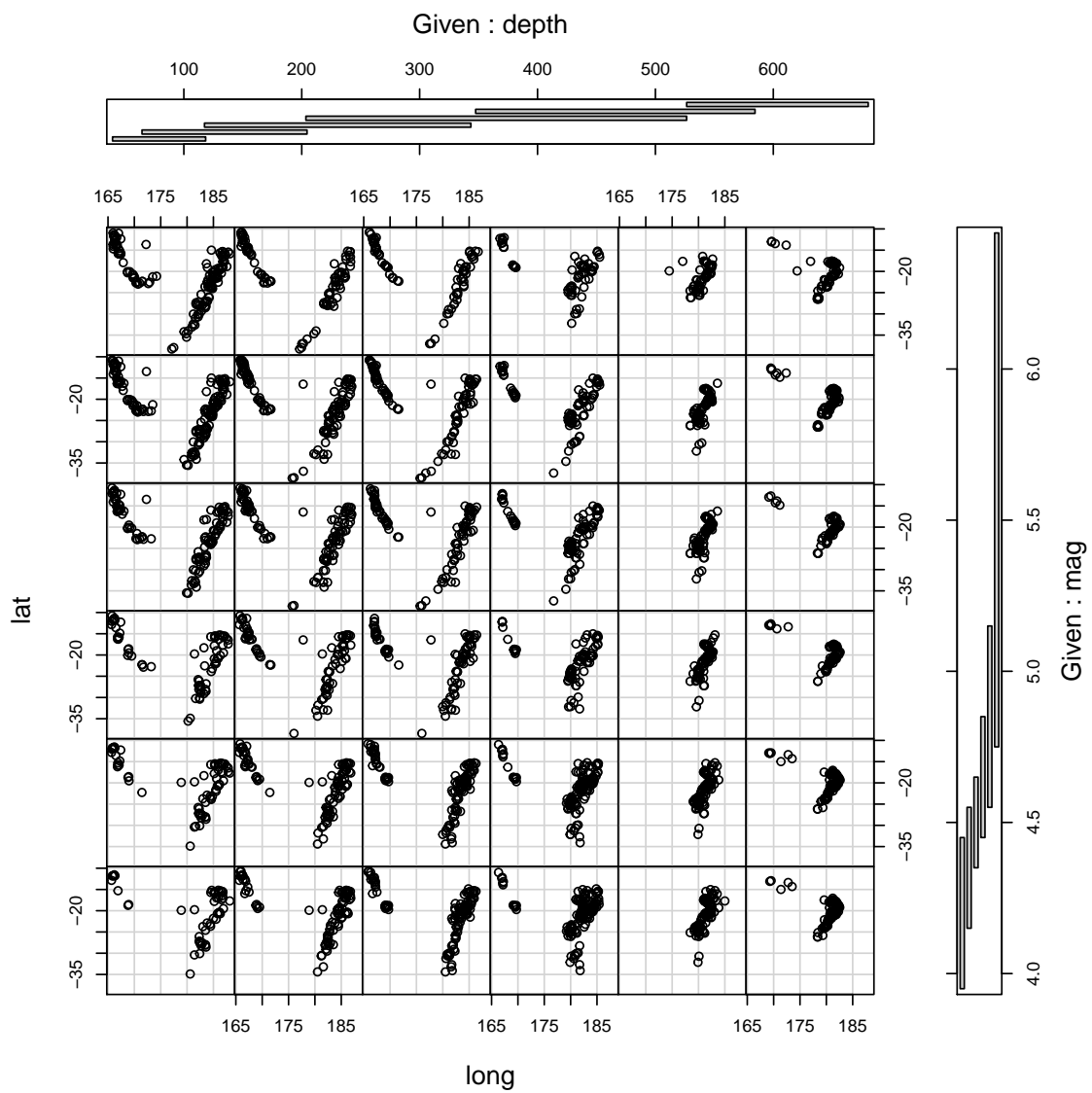
```
## Tonga Trench Earthquakes
coplot(lat ~ long | depth, data = quakes)
```



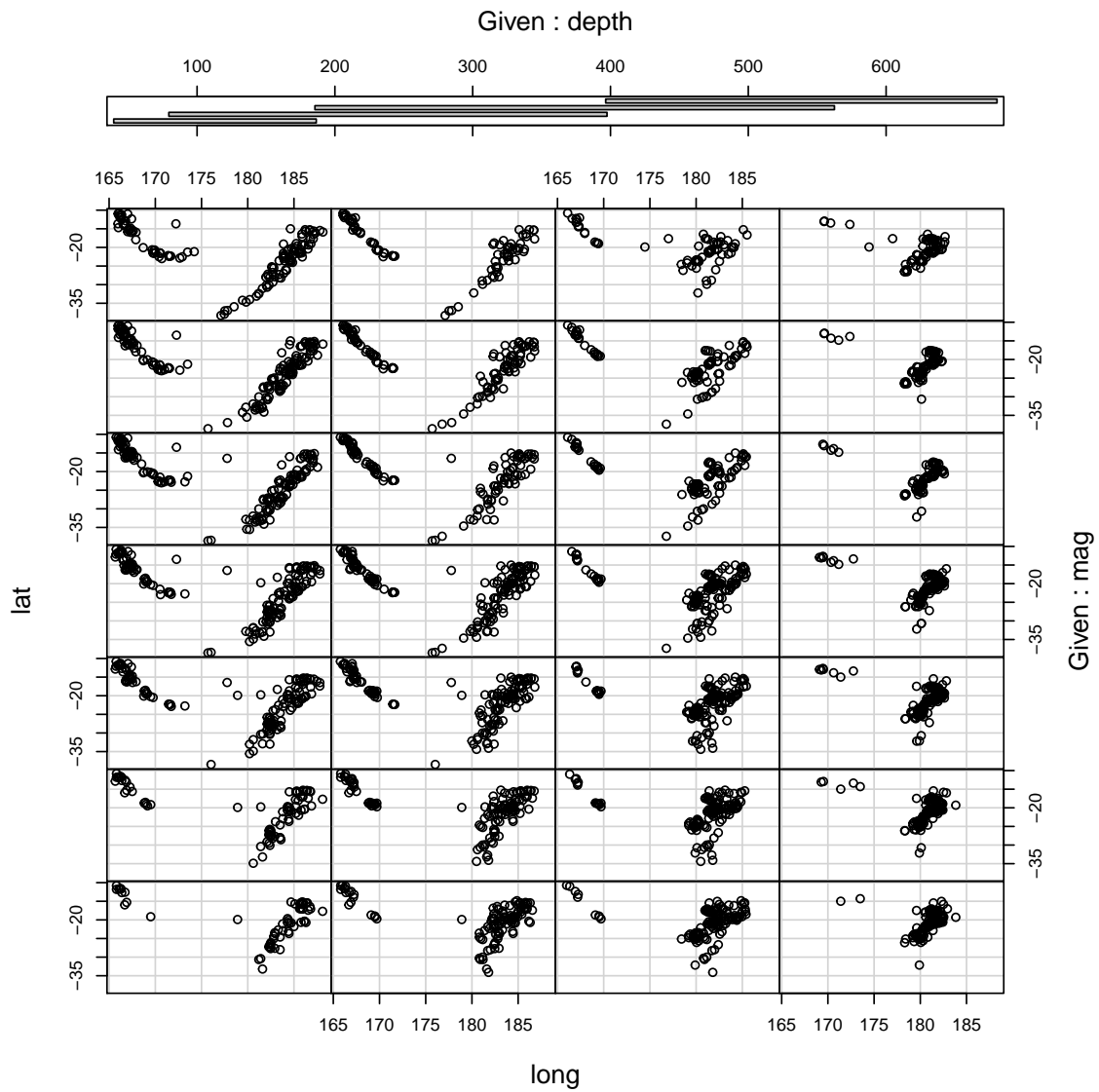
```
given.depth <- co.intervals(quakes$depth, number=4, overlap=.1)
coplot(lat ~ long | depth, data = quakes, given.v=given.depth, rows=1)
```



```
## Conditioning on 2 variables:  
ll.dm <- lat ~ long | depth * mag  
coplot(ll.dm, data = quakes)
```

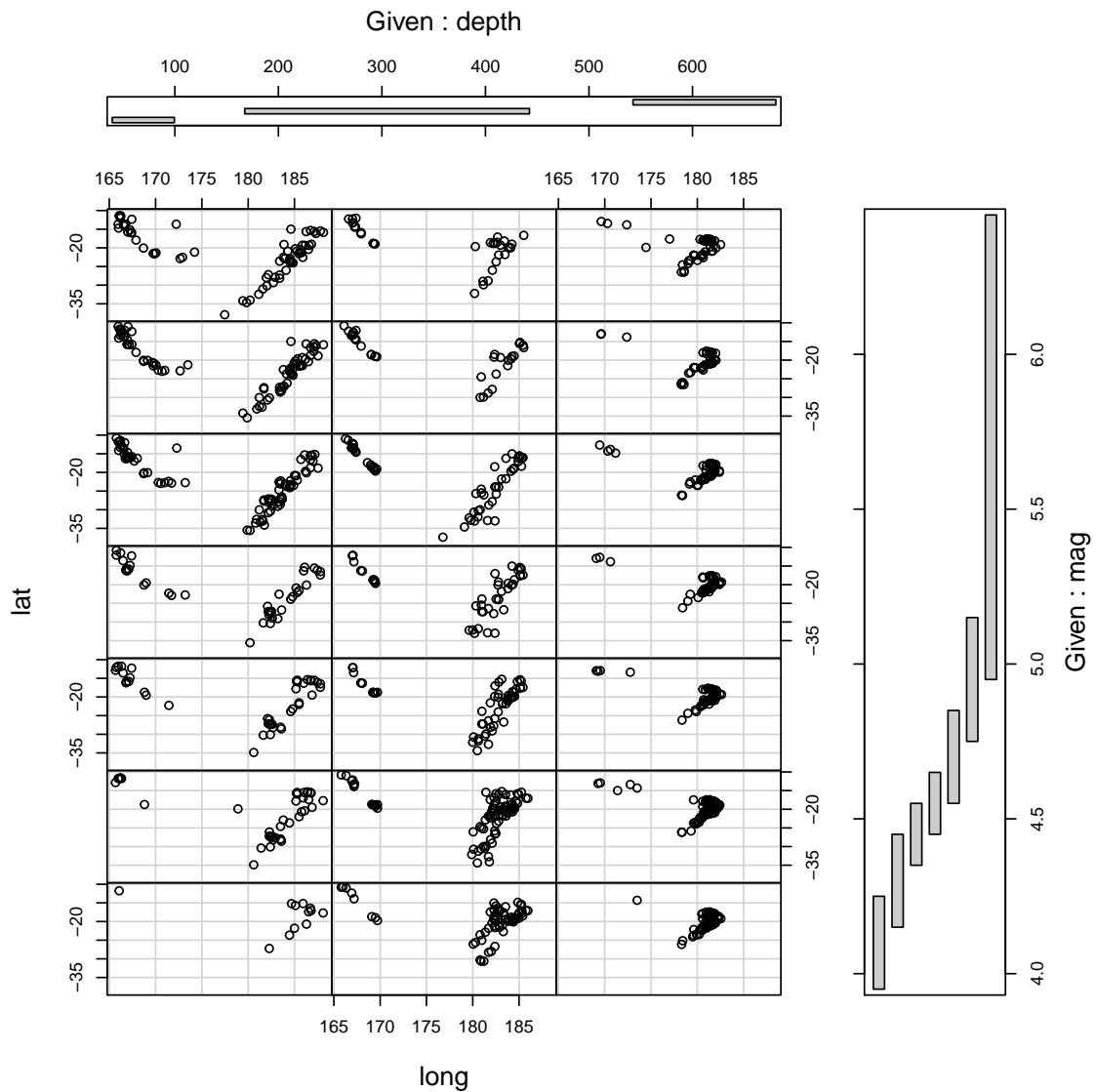


```
coplot(l1.dm, data = quakes, number=c(4,7), show.given=c(TRUE,FALSE))
```

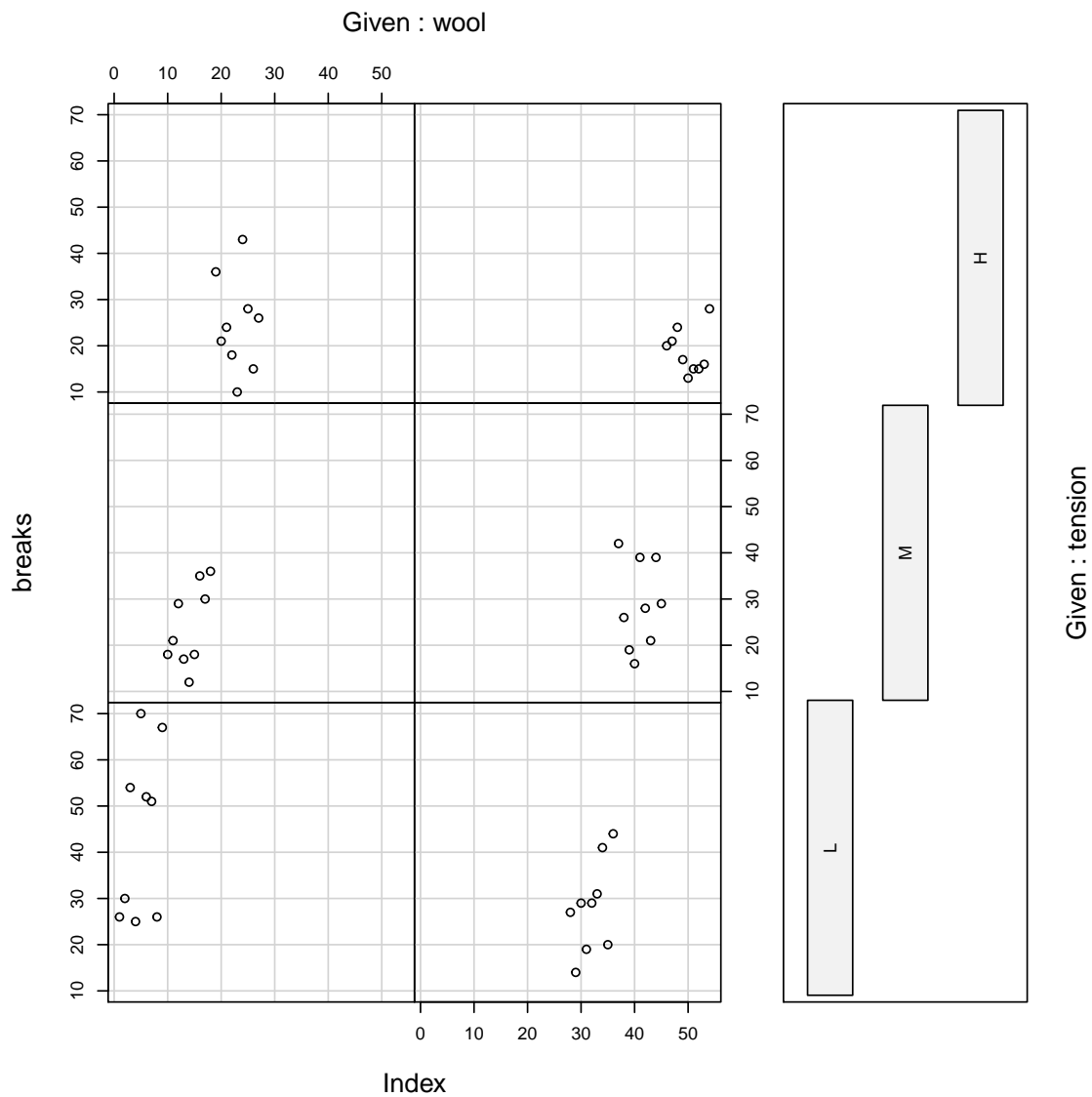


```
coplot(l1.dm, data = quakes, number=c(3,7),
       overlap=c(-.5,.1)) # negative overlap DROPS values
```





```
## given two factors
Index <- seq(length=nrow(warpbreaks)) # to get nicer default labels
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"))
```

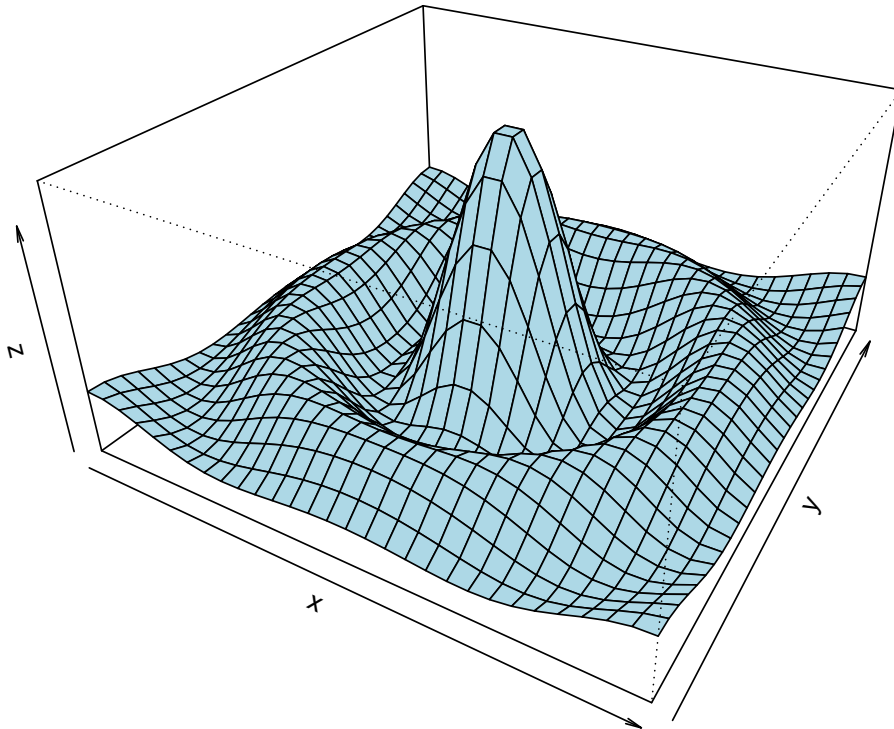
```
## Example with empty panels:
with(data.frame(state.x77), {
coplot(Life.Exp ~ Income | Illiteracy * state.region, number = 3,
       panel = function(x, y, ...) panel.smooth(x, y, span = .8, ...))
})
```

```
## y ~ factor -- not really sensical, but 'show off':  
with(data.frame(state.x77), {  
  coplot(Life.Exp ~ state.region | Income * state.division,  
         panel = panel.smooth)  
})
```

## 18 persp

```
require(grDevices)  
# for trans3d  
## More examples in demo(persp) !!  
## -----  
# (1) The Obligatory Mathematical surface.  
#   Rotated sinc function.  
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")
```



```

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
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

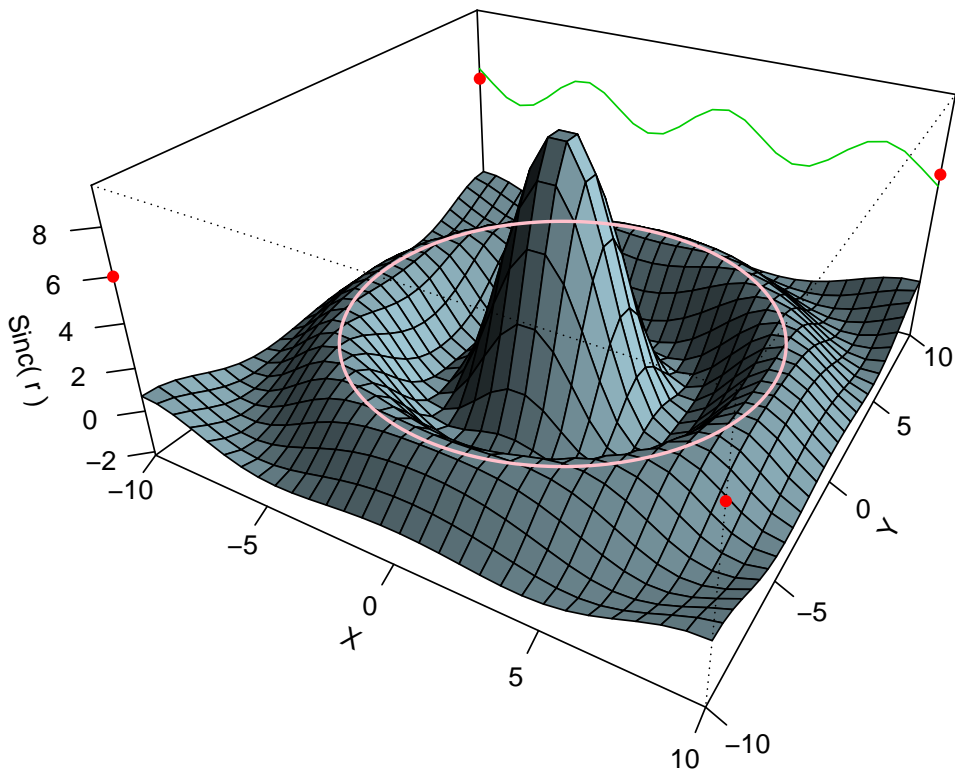
# (2) Add to existing persp plot - using trans3d() :
xE <- c(-10,10); xy <- expand.grid(xE, xE)

```

```

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)

```



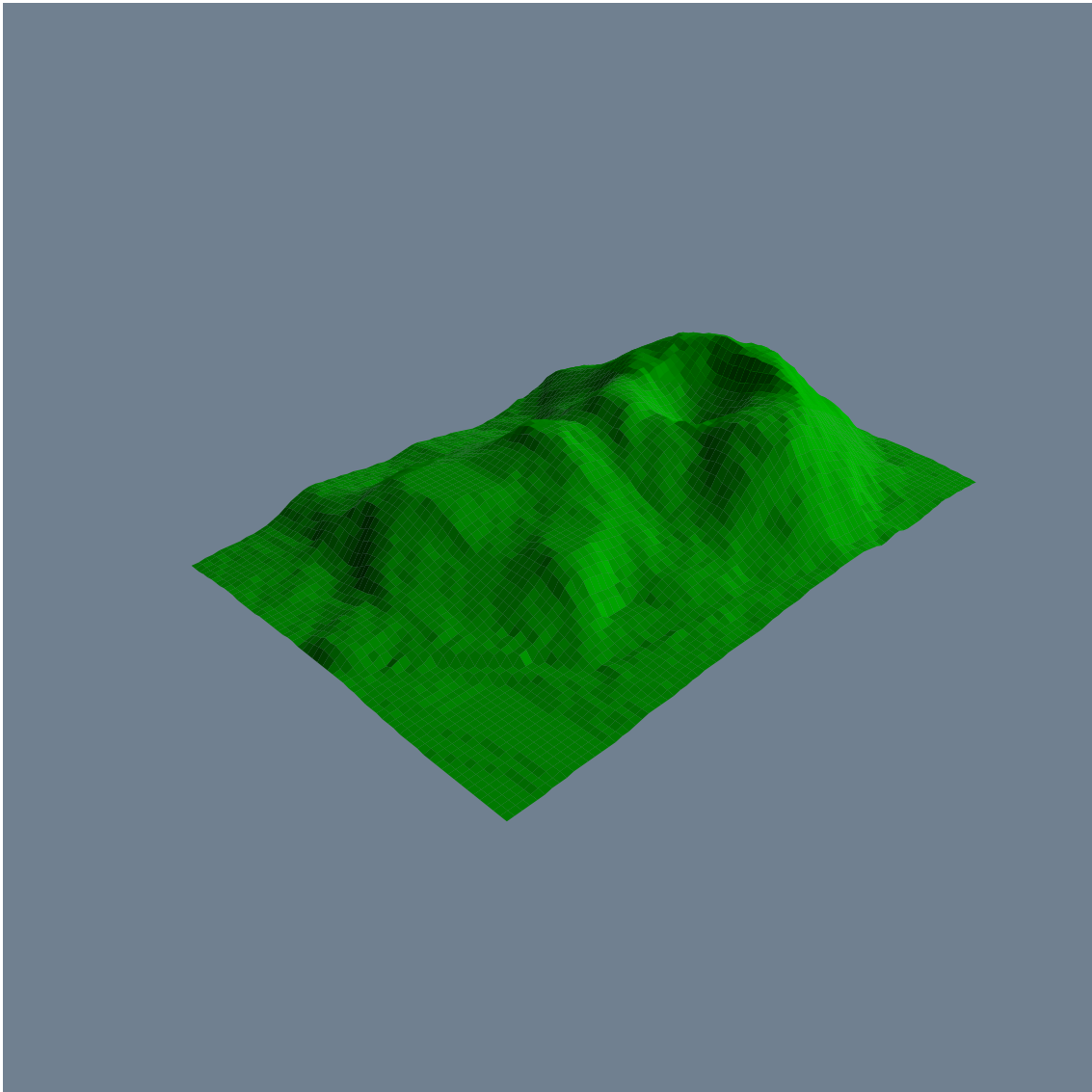
```
## (no hidden lines)
```

```

# (3) Visualizing a simple DEM model
z <- 2 * volcano # Exaggerate the relief

```

```
x <- 10 * (1:nrow(z)) # 10 meter spacing (S to N)
y <- 10 * (1:ncol(z)) # 10 meter spacing (E to W)
## Don't draw the grid lines : border = NA
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)
```

## Table des matières

1 Contenu

1

---

<b>2</b>	<b>dotchart</b>	<b>1</b>
<b>3</b>	<b>barplot</b>	<b>3</b>
<b>4</b>	<b>hist</b>	<b>12</b>
<b>5</b>	<b>Boîtes à moustaches</b>	<b>17</b>
5.1	boxplot d'une formule . . . . .	17
5.2	boxplot d'une matrice . . . . .	20
<b>6</b>	<b>pie</b>	<b>23</b>
<b>7</b>	<b>Tableaux de contingence</b>	<b>30</b>
7.1	balloonplot . . . . .	30
7.2	assocplot . . . . .	33
7.3	mosaicplot . . . . .	34
7.4	splineplot . . . . .	40
7.5	cdplot (Conditional Density Plots) . . . . .	47
<b>8</b>	<b>Plot factor variables</b>	<b>53</b>
<b>9</b>	<b>Matrix plot</b>	<b>57</b>
<b>10</b>	<b>Simplepanel plots</b>	<b>62</b>
<b>11</b>	<b>jitter</b>	<b>64</b>
<b>12</b>	<b>curves</b>	<b>65</b>
<b>13</b>	<b>loess (regression non-paramétrique)</b>	<b>74</b>
<b>14</b>	<b>density estimation</b>	<b>77</b>
<b>15</b>	<b>Radar plots</b>	<b>87</b>
<b>16</b>	<b>Steam and leaf</b>	<b>99</b>
<b>17</b>	<b>Conditioning plots</b>	<b>100</b>
<b>18</b>	<b>persp</b>	<b>107</b>