Exploratory Data Analysis

Chapter 5

Instructor: Seokho Lee

Hankuk University of Foreign Studies

5. Statistical Graphics

5.1. Distribution

- Density, cumulative distribution function, quantile function and random variate generation for many standard probability distributions are available in the stats package.
- The functions for the density/mass function, cumulative distribution function, quantile function and random variate generation are named in the form `dxxx`, `pxxx`, `qxxx` and `rxxx` respectively.

distribution	name	parameters
beta	`beta`	`shape1, shape2, ncp`
binomial	`binom`	`size, prob`
Cauchy	`cauchy`	`location, scale`
chi-square	`chisq`	`df, ncp`
exponential	`exp`	`rate`
F	`f`	`df1, df2, ncp`
gamma	`gamma`	`shape, scale`
geometry	`geom`	`prob`
hypergeometry	`hyper`	`m, n, k`
log-normal	`lnorm`	`meanlog, sdlog`
logistic	`logis`	`location, scale`
negative binomial	`nbinom`	`size, prob`
normal	`norm`	`mean, sd`
Poisson	`pois`	`lambda`
t	`t`	`df, ncp`
uniform	`unif`	`min, max`
Weibull	`weibull`	`shape, scale`
Wilcoxon	`wilcox`	`m, n`

Definition (normal)

```
dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
rnorm(n, mean = 0, sd = 1)
```

Example (normal)

```
> dnorm(x=5, mean=3, sd=2)     # density at X=5 from N(3,4)
[1] 0.1209854
> 1 / (sqrt(2*pi)*2) * exp(-(5-3)^2 / (2*2^2))     # density from the definition
[1] 0.1209854
> pnorm(q=5, mean=3, sd=2)    # cumulative distribution of X=5 from N(3,4)
[1] 0.8413447
> qnorm(p=0.8413447, mean=3, sd=2)    # 0.8413447-quantile of N(3,4)
[1] 5
> rnorm(n=5, mean=3, sd=2)    # 5 random numbers from N(3,4)
[1] -0.2804209  5.1013362  3.0247510  1.0761024  6.3510984
```

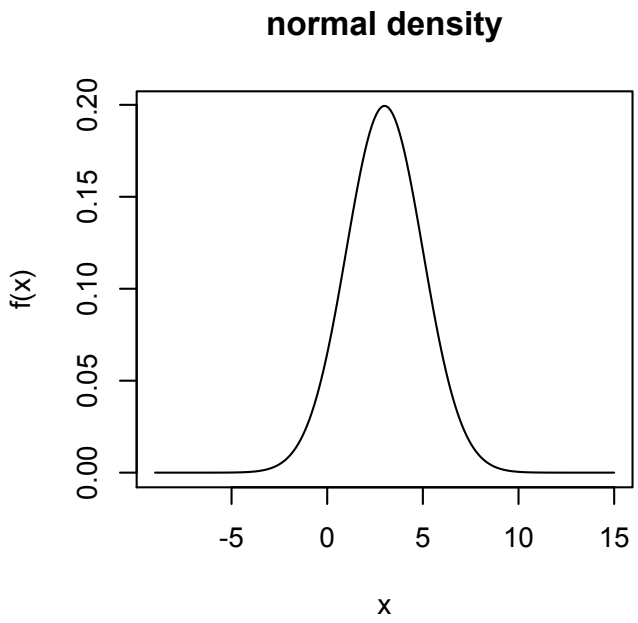
Example (normal)

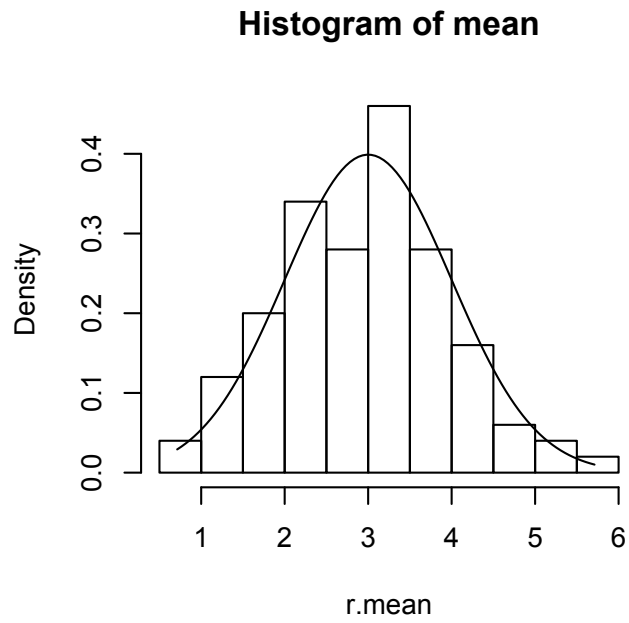
Suppose $X \sim N(3, 2^2)$.

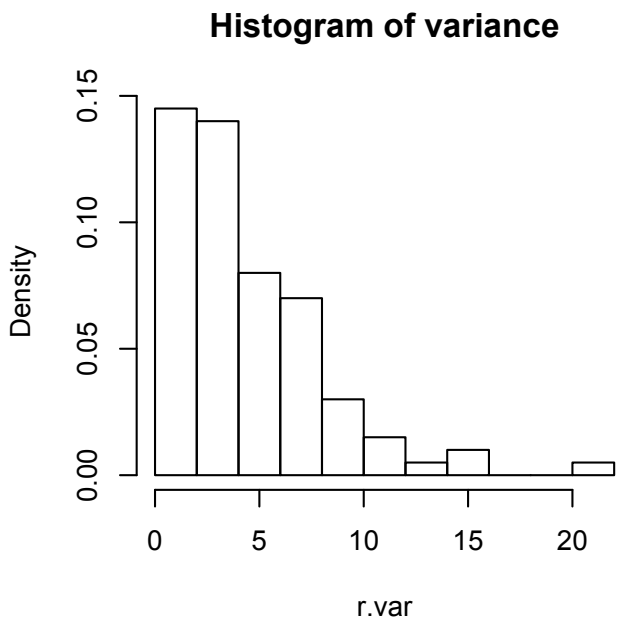
1. Compute $\Pr(X > 5)$.
2. Find c such that $\Pr(X > c)$=0.7.
3. Draw the density plot.
4. Draw the histogram of 100 sample means of 4 random numbers from the above distribution.
5. Draw the histogram of 100 sample variances of 4 random numbers from the above distribution.

Example (normal)

```
> 1-pnorm(q=5, mean=3, sd=2)     # (1)
[1] 0.1586553
> pnorm(q=5, mean=3, sd=2, lower.tail=FALSE)
[1] 0.1586553
> qnorm(p=0.3, mean=3, sd=2)     # (2)
[1] 1.951199
> qnorm(p=0.7, mean=3, sd=2, lower.tail=FALSE)
[1] 1.951199
> norm.mean<-3; norm.sd<-2     # (3)
> x<-seq(norm.mean-6*norm.sd, norm.mean+6*norm.sd, length=200)
> y<-dnorm(x=x, mean=3, sd=2)
> par(mfrow=c(2,2))
> plot(x,y,type="l",ylab="f(x)",main="normal density")
> r.mean<-rep(0,100); r.var<-rep(0,100)     # (4) and (5)
> for(i in seq(100)){
+   r<-rnorm(n=4,mean=3,sd=2)
+   r.mean[i]<-mean(r); r.var[i]<-var(r)
+ }
> hist(r.mean,prob=TRUE,main="Histogram of mean")
> x1<-seq(min(r.mean),max(r.mean),length=100)
> y1<-dnorm(x=x1,mean=3,sd=1)
> lines(x1,y1)
> hist(r.var,prob=TRUE,main="Histogram of variance")
> # alternative way for (3) and (4)
> # without using a loop
> samples<-matrix(rnorm(400,mean=3,sd=2),ncol=4)
> r.mean<-apply(samples,1,mean)
> r.var<-apply(samples,1,var)
```


normal density
f(x)
0.00
0.05
0.10
0.15
0.20
-5
0
5
10
15
x


Histogram of mean
Density
0.0
0.1
0.2
0.3
0.4
1
2
3
4
5
6
r.mean


Histogram of variance
Density
0.00
0.05
0.10
0.15
0
5
10
15
20
r.var

Definition (binomial)

```
dbinom(x, size, prob, log = FALSE)
pbinom(q, size, prob, lower.tail = TRUE, log.p = FALSE)
qbinom(p, size, prob, lower.tail = TRUE, log.p = FALSE)
rbinom(n, size, prob)
```

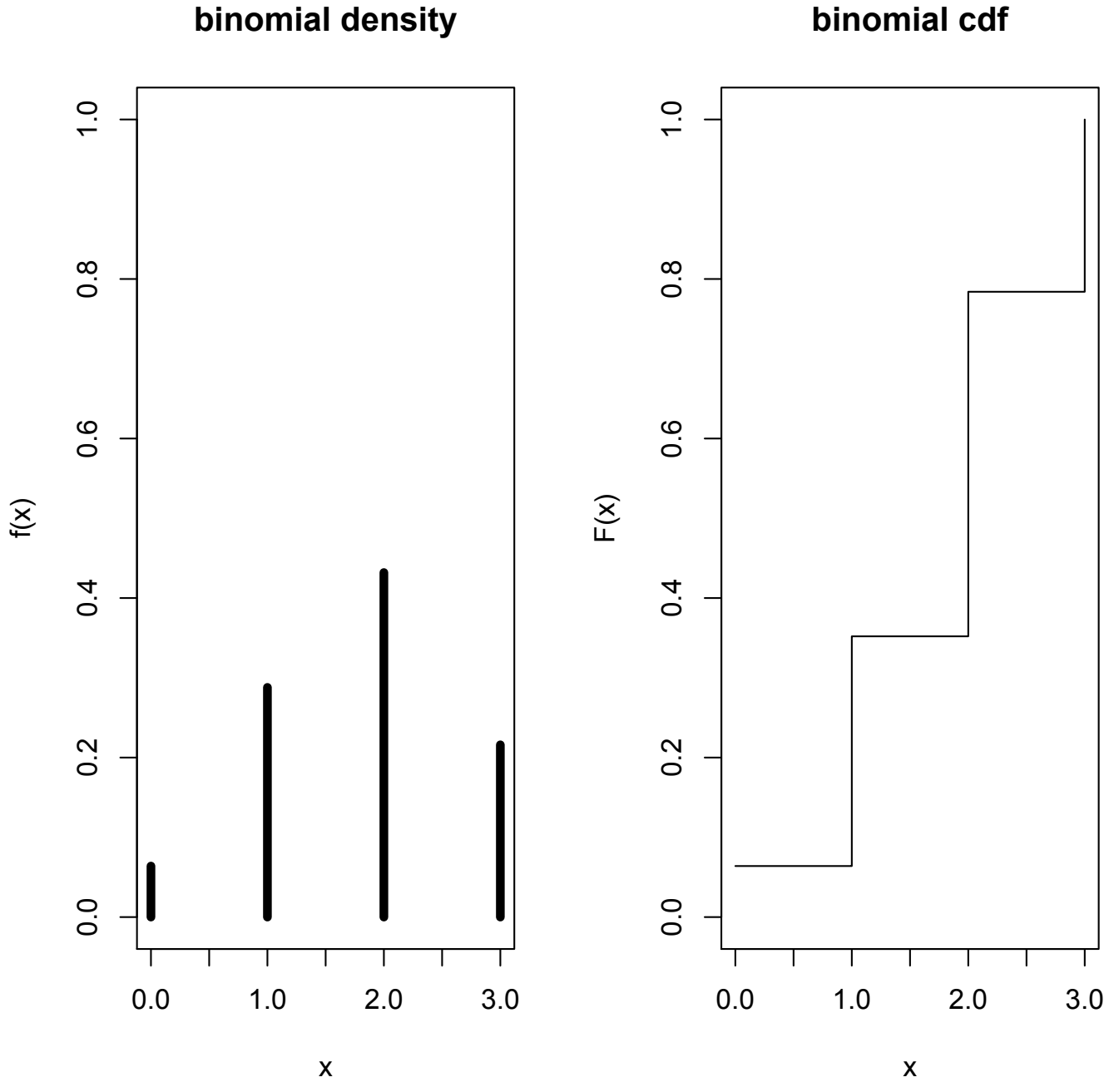
Example (binomial)

Suppose $X \sim B(3, 0.6)$.

1. Obtain the probability mass function of X.
2. Compute $\Pr(X = 2)$.
3. Compute $\Pr(X < 2.5)$.
4. Compute the mean and variance of 5 random numbers.
5. Draw the probability mass function and distribution function of X.

Example (binomial)

```
> dbinom(0:3, size=3, prob=0.6)     # (1) probability mass function
[1] 0.064 0.288 0.432 0.216
> choose(3,1)     # binomial coefficient (3,1)
[1] 3
> choose(3,0:3)     # binomial coefficients (3,0),(3,1),(3,2),(3,3)
[1] 1 3 3 1
> choose(3,0:3)*(0.6)^(0:3)*(1-0.6)^(3:0)
[1] 0.064 0.288 0.432 0.216
> dbinom(2, 3, 0.6)     # (2) probability of X=2
[1] 0.432
> pbinom(2.5, 3, 0.6)     # (3)  cumulative probability of X<2.5
[1] 0.784
> pbinom(2, 3, 0.6)     # cumulative probability of X<=2
[1] 0.784
> sum(dbinom(0:2,3,0.6))
[1] 0.784
> (samples<-rbinom(5,3,0.6))     # (4)
[1] 2 2 2 0 2
> mean(samples)
[1] 1.6
> var(samples)
[1] 0.8
> par(mfrow=c(1,2))     # (5)
> x<-0:3
> y<-dbinom(x,3,0.6)
> plot(x,y,type="h",xlab="x",ylab="f(x)",ylim=c(0,1.0),lwd=5)
> title("binomial density")
> z<-pbinom(x,3,0.6)
> plot(x,z,type="s",xlab="x",ylab="F(x)",ylim=c(0,1))
> title("binomial cdf")
```


binomial density
f(x)
0.0
0.2
0.4
0.6
0.8
1.0
0.0
1.0
2.0
3.0
x
binomial cdf
F(x)
0.0
0.2
0.4
0.6
0.8
1.0
0.0
1.0
2.0
3.0
x

Definition (chi-square, t, F)

```
dchisq(x, df, ncp = 0, log = FALSE)
pchisq(q, df, ncp = 0, lower.tail = TRUE, log.p = FALSE)
qchisq(p, df, nap = 0, lower.tail = TRUE, log.p = FALSE)
rchisq(n, df, ncp = 0)

dt(x, df, ncp, log = FALSE)
pt(q, df, ncp, lower.tail = TRUE, log.p = FALSE)
qt(p, df, ncp, lower.tail = TRUE, log.p = FALSE)
rt(n, df, ncp)

df(x, df1, df2, ncp, log = FALSE)
pf(q, df1, df2, ncp, lower.tail = TRUE, log.p = FALSE)
qf(p, df1, df2, ncp, lower.tail = TRUE, log.p = FALSE)
rf(n, df1, df2, ncp)
```

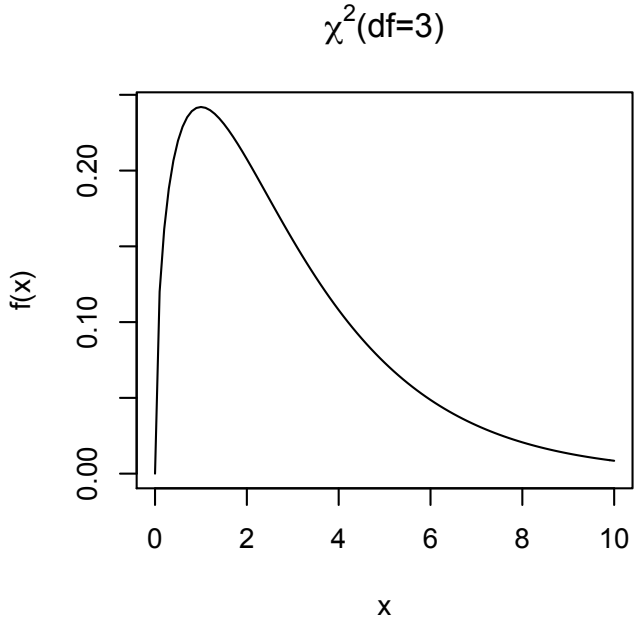
Example (chi-square, t, F)

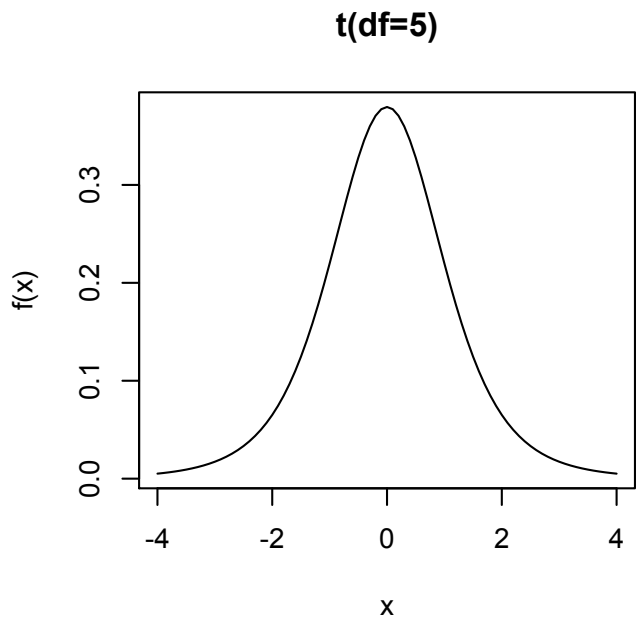
Suppose $X \sim \chi^2(3)$, $Y \sim t(5)$ and $Z \sim F(5,7)$.

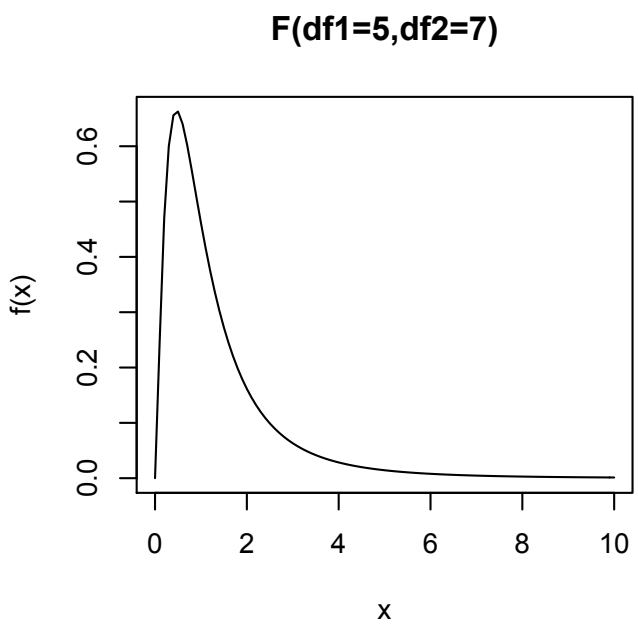
1. Compute $\Pr(X > 5)$
2. Find c such that $\Pr(X \leq c) = 0.7$.
3. Draw the probability density function of X.
4. Compute $\Pr(Y > 7)$
5. Find c such that $\Pr(Y \leq c) = 0.7$.
6. Draw the probability density function of Y.
7. Compute $\Pr(Z > 2)$
8. Find c such that $\Pr(Z \leq c) = 0.7$.
9. Draw the probability density function of Z.

Example (chi-square, t, F)

```
> 1-pchisq(q=5,df=3)     # (1)
[1] 0.1717971
> pchisq(q=5,df=3,lower.tail=FALSE)
[1] 0.1717971
> qchisq(p=0.7,df=3)     # (2)
[1] 3.664871
> 1-pt(q=7,df=5)     # (4)
[1] 0.0004583738
> pt(q=7,df=5,lower.tail=FALSE)
[1] 0.0004583738
> qt(p=0.7,df=5)     # (5)
[1] 0.5594296
> 1-pf(q=2,df1=5,df2=7)     # (7)
[1] 0.1956732
> pf(q=2,df1=5,df2=7,lower.tail=FALSE)
[1] 0.1956732
> qf(p=0.7,df1=5,df2=7)     # (8)
[1] 1.505473
> par(mfrow=c(2,2))
> x1<-(0:100)/10     # (3)
> y1<-dchisq(x1,df=3)
> plot(x1,y1,type="l",xlab="x",ylab="f(x)",main=expression(paste(chi^2,"(df=3)",sep="")))
> x2<-(-40:40)/10     # (6)
> y2<-dt(x2,df=5)
> plot(x2,y2,type="l",xlab="x",ylab="f(x)",main="t(df=5)")
> x3<-(0:100)/10     # (9)
> y3<-df(x3,df1=5,df2=7)
> plot(x3,y3,type="l",xlab="x",ylab="f(x)",main="F(df1=5,df2=7)")
```


χ^2(df=3)
f(x)
0.00
0.10
0.20
0
2
4
6
8
10
x


t(df=5)
f(x)
0.0
0.1
0.2
0.3
-4
-2
0
2
4
x


F(df1=5,df2=7)
f(x)
0.0
0.2
0.4
0.6
0
2
4
6
8
10
x

Definition (exponential, Poisson, hypergeometric)

```
dexp(x, rate = 1, log = FALSE)
pexp(q, rate = 1, lower.tail = TRUE, log.p = FALSE)
qexp(p, rate = 1, lower.tail = TRUE, log.p = FALSE)
rexp(n, rate = 1)

dpois(x, lambda, log = FALSE)
ppois(q, lambda, lower.tail = TRUE, log.p = FALSE)
qpois(p, lambda, lower.tail = TRUE, log.p = FALSE)
rpois(n, lambda)

dhyper(x, m, n, k, log = FALSE)
phyper(q, m, n, k, lower.tail = TRUE, log.p = FALSE)
qhyper(p, m, n, k, lower.tail = TRUE, log.p = FALSE)
rhyper(nn, m, n, k)
```

5.2. High-level graphic functions

Definition (barplot)

```
# Creates a bar plot with vertical or horizontal bars.

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

height
either a vector or matrix of values describing the bars which make up the plot. If height is a vector, the plot consists of a sequence of rectangular bars with heights given by the values in the vector. If height is a matrix and beside is FALSE then each bar of the plot corresponds to a column of height, with the values in the column giving the heights of stacked sub-bars making up the bar. If height is a matrix and beside is TRUE, then the values in each column are juxtaposed rather than stacked.

width
optional vector of bar widths. Re-cycled to length the number of bars drawn. Specifying a single value will have no visible effect unless xlim is specified.

space
the amount of space (as a fraction of the average bar width) left before each bar. May be given as a single number or one number per bar. If height is a matrix and beside is TRUE, space may be specified by two numbers, where the first is the space between bars in the same group, and the second the space between the groups. If not given explicitly, it defaults to c(0,1) if height is a matrix and beside is TRUE, and to 0.2 otherwise.

names.arg
a vector of names to be plotted below each bar or group of bars. If this argument is omitted, then the names are taken from the names attribute of height if this is a vector, or the column names if it is a matrix.

legend.text
a vector of text used to construct a legend for the plot, or a logical indicating whether a legend should be included. This is only useful when height is a matrix. In that case given legend labels should correspond to the rows of height; if legend.text is true, the row names of height will be used as labels if they are non-null.

beside
a logical value. If FALSE, the columns of height are portrayed as stacked bars, and if TRUE the columns are portrayed as juxtaposed bars.

horiz
a logical value. If FALSE, the bars are drawn vertically with the first bar to the left. If TRUE, the bars are drawn horizontally with the first at the bottom.

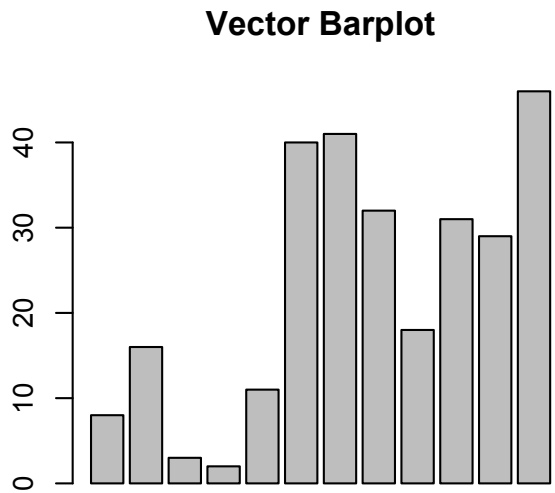
density
a vector giving the density of shading lines, in lines per inch, for the bars or bar components. The default value of NULL means that no shading lines are drawn. Non-positive values of density also inhibit the drawing of shading lines.

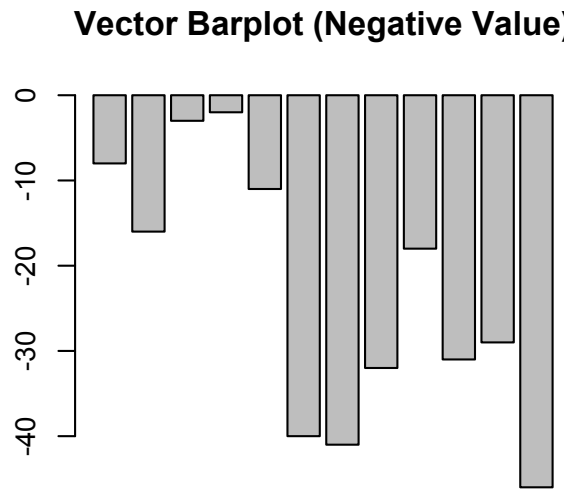
angle
the slope of shading lines, given as an angle in degrees (counter-clockwise), for the bars or bar components.

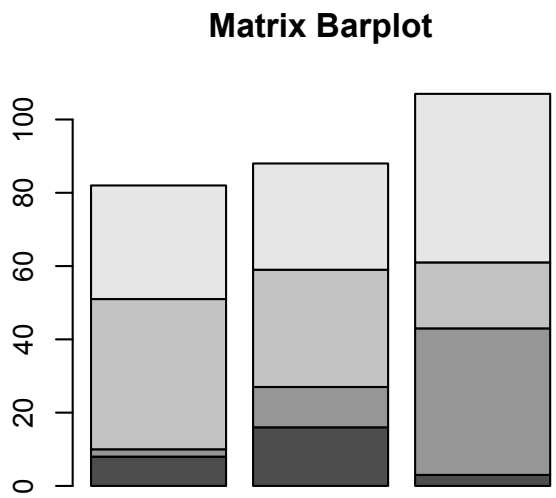
col
a vector of colors for the bars or bar components. By default, grey is used if height is a vector, and a gamma-corrected grey palette if height is a matrix.

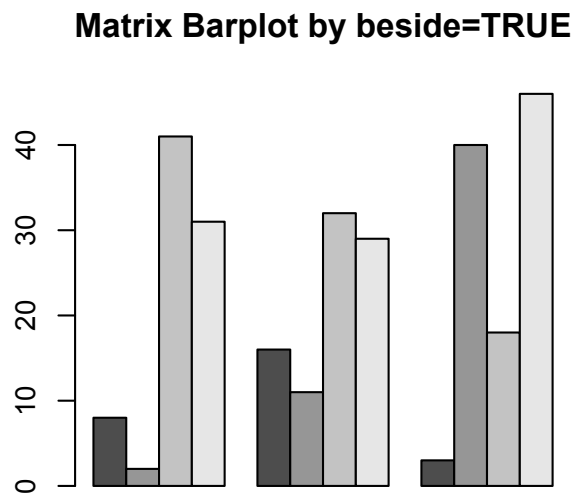
Example (barplot 1)

```
> bar.x <- round(runif(12)*50)
> bar.x
 [1]  8 16  3  2 11 40 41 32 18 31 29 46
> bar.y <- matrix(bar.x, ncol=3, byrow=T)
> bar.y
     [,1] [,2] [,3]
[1,]    8   16    3
[2,]    2   11   40
[3,]   41   32   18
[4,]   31   29   46
> par(mfrow=c(2,2))
> barplot(bar.x)
> title(main="Vector Barplot")
> barplot(-bar.x)
> title(main="Vector Barplot (Negative Value)")
> barplot(bar.y)
> title(main="Matrix Barplot")
> barplot(bar.y, beside=TRUE)
> title(main="Matrix Barplot by beside=TRUE")
```


Vector Barplot
0
10
20
30
40


Vector Barplot (Negative Value)
0
-10
-20
-30
-40

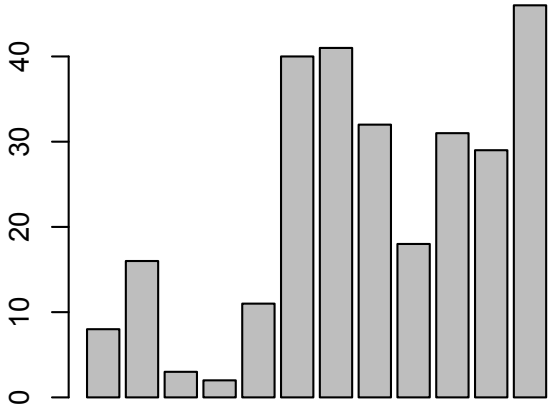

Matrix Barplot
0
20
40
60
80
100


Matrix Barplot by beside=TRUE
0
10
20
30
40

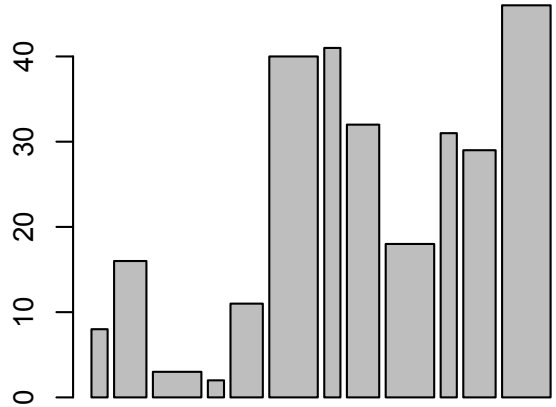
Example (barplot 2)

```
> bar.width <- rep(1:3, 4)
> bar.width
 [1] 1 2 3 1 2 3 1 2 3 1 2 3
> par(mfrow=c(2,2))
> barplot(bar.x, width=1)
> title(main="Vector Barplot by default width")
> barplot(bar.x, width=bar.width)
> title(main="Vector Barplot by width 1:3")
> barplot(bar.x, space=2)
> title(main="Vector Barplot by space=2")
> barplot(bar.y, beside=TRUE, space=c(0.5,2))
> title(main="Matrix Barplot by space=c(0.5,2)")
```

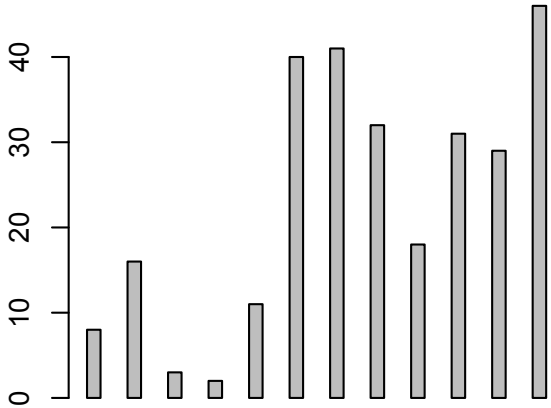
Vector Barplot by default width



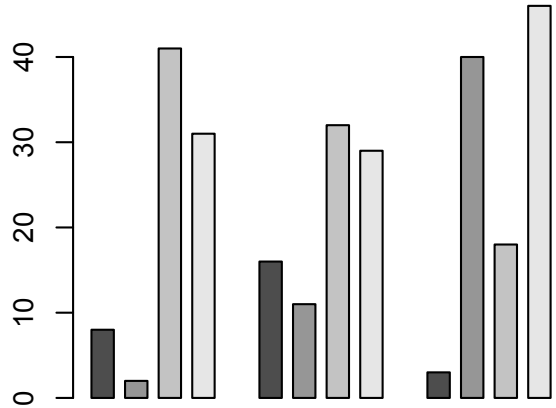
Vector Barplot by width 1:3



Vector Barplot by space=2



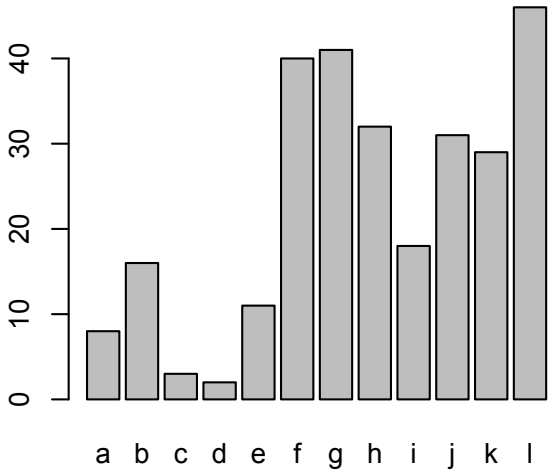
Matrix Barplot by space=c(0.5,2)



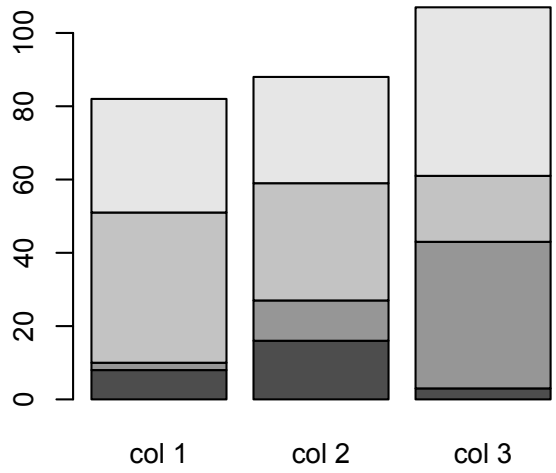
Example (barplot 3)

```
> rownames(bar.y) <- paste("row", 1:4)
> rownames(bar.y)
[1] "row 1" "row 2" "row 3" "row 4"
> colnames(bar.y) <- paste("col", 1:3)
> colnames(bar.y)
[1] "col 1" "col 2" "col 3"
> par(mfrow=c(2,2))
> barplot(bar.x, names.arg=letters[1:length(bar.x)])
> title(main="Vector Barplot using names.arg")
> barplot(bar.y)
> title(main="Matrix Barplot using default names.arg")
> barplot(bar.x, legend.text=letters[1:length(bar.x)])
> title(main="Vector Barplot using legend.text")
> barplot(bar.y, legend.text=T)
> title(main="Matrix Barplot using legend.text=T")
```

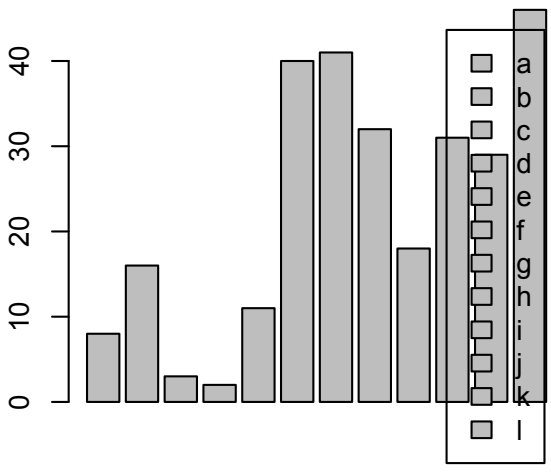
Vector Barplot using names.arg



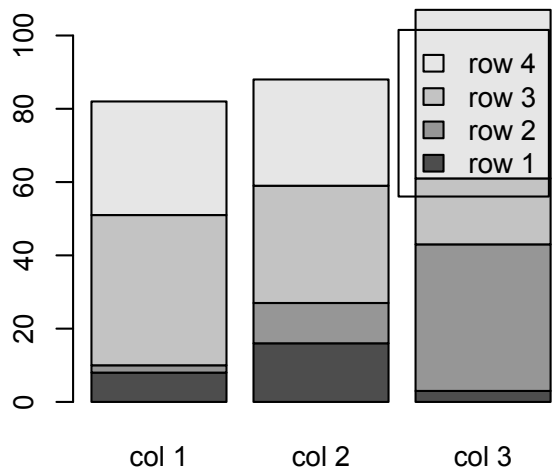
Matrix Barplot using default names.arg



Vector Barplot using legend.text



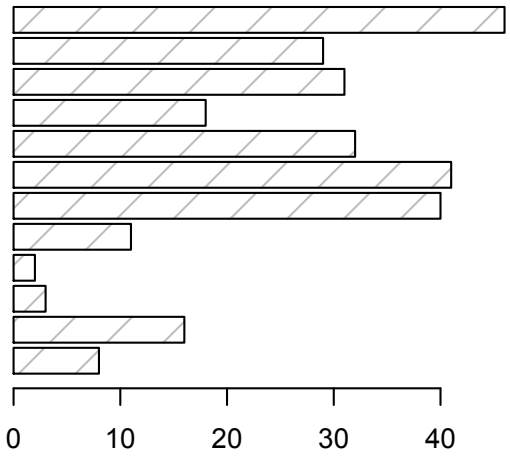
Matrix Barplot using legend.text=T



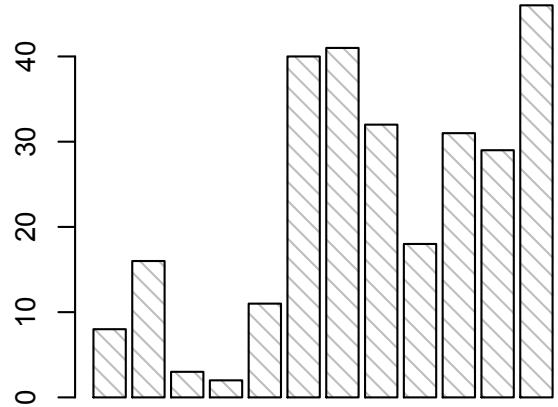
Example (barplot 4)

```
> par(mfrow=c(2,2))
> barplot(bar.x, horiz=T, density=5)
> title(main="Vector Barplot by horiz=T, density=5")
> barplot(bar.x, density=15, angle=135)
> title(main="Vector Barplot by density=15, angle=135")
> barplot(bar.x, col=rainbow(length(bar.x)))
> title(main="Vector Barplot by rainbow color")
> barplot(bar.y, border="red", col=c("lightblue", "mistyrose", "lightcyan", "lavender"))
> title(main="Matrix Barplot by col, border")
```

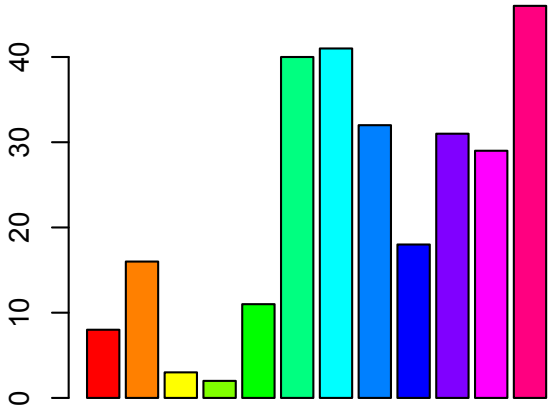
Vector Barplot by horiz=T, density=5



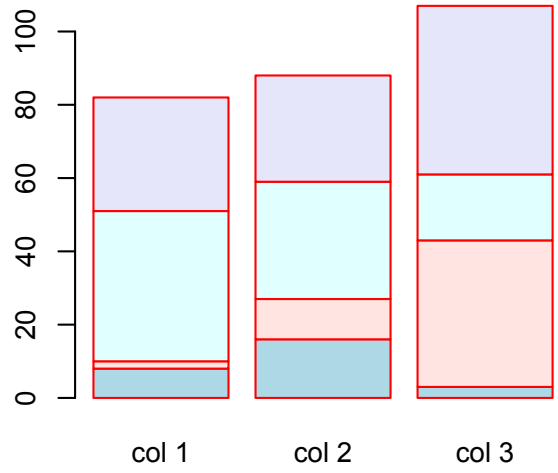
Vector Barplot by density=15, angle=135



Vector Barplot by rainbow color

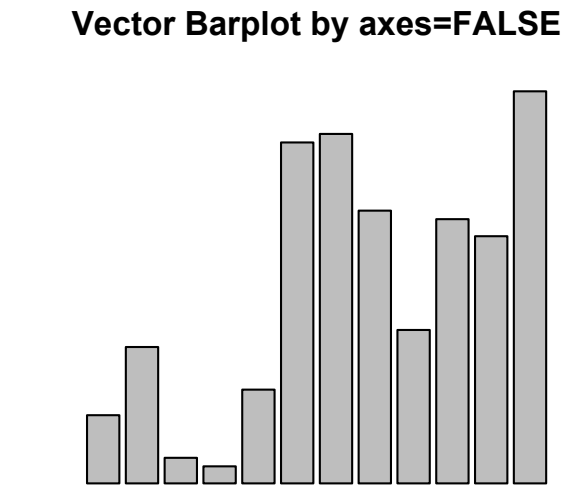


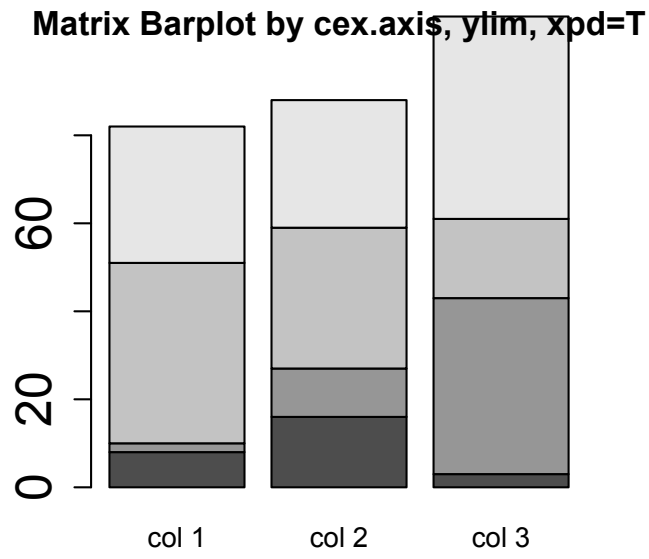
Matrix Barplot by col, border

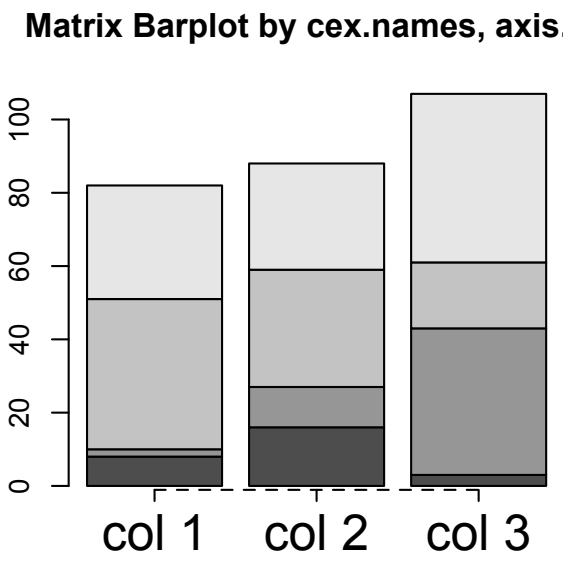


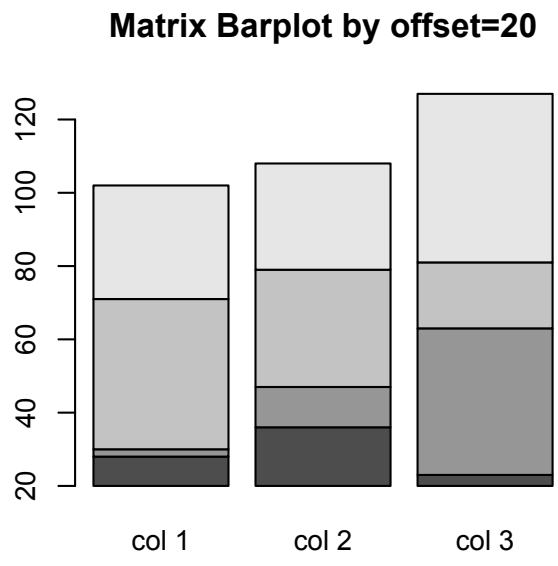
Example (barplot 5)

```
> par(mfrow=c(2,2))
> barplot(bar.x, axes=FALSE)
> title(main="Vector Barplot by axes=FALSE")
> barplot(bar.y, cex.axis=1.8, ylim=c(0,90), xpd=T)
> title(main="Matrix Barplot by cex.axis, ylim, xpd=T")
> barplot(bar.y, axisnames=T, cex.names=1.8, axis.lty=2)
> title(main="Matrix Barplot by cex.names, axis.lty")
> t(barplot(bar.x, plot=F))
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
[1,]  0.7  1.9  3.1  4.3  5.5  6.7  7.9  9.1 10.3  11.5  12.7  13.9
> barplot(bar.y, plot=F)
[1] 0.7 1.9 3.1
> barplot(bar.y, offset=20, main="Matrix Barplot by offset=20")
```


Vector Barplot by axes=FALSE


Matrix Barplot by cex.axis, ylim, xpd=T
0
20
60
col 1
col 2
col 3


Matrix Barplot by cex.names, axis.lty
0
20
40
60
80
100
col 1
col 2
col 3

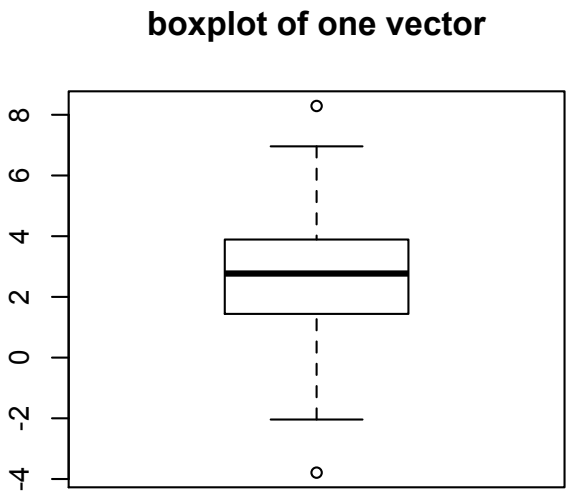

Matrix Barplot by offset=20
20
40
60
80
100
120
col 1
col 2
col 3

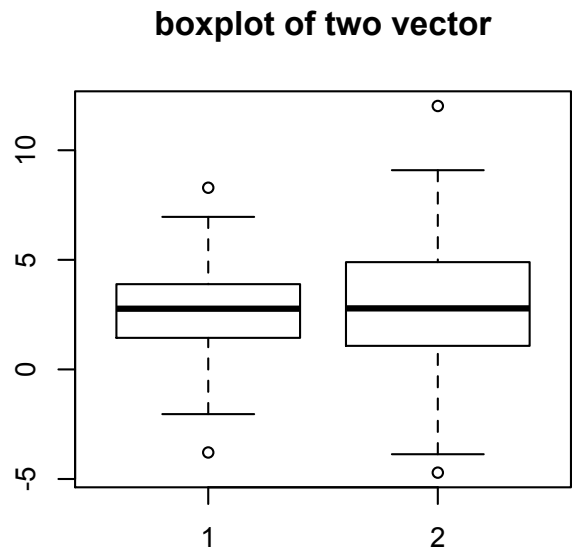
Definition (boxplot)

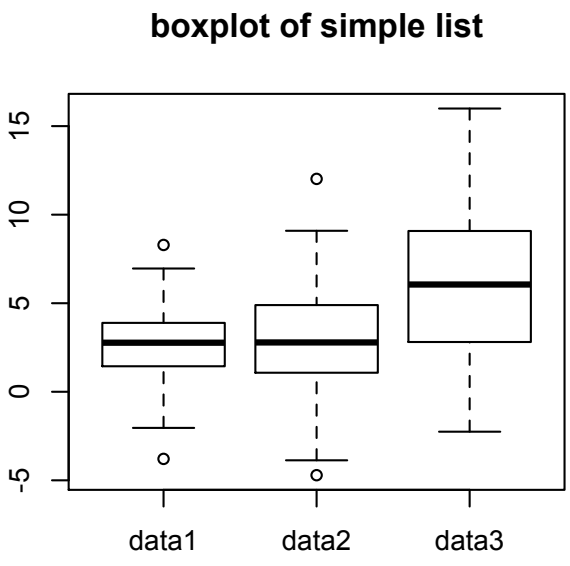
```
# Produce box-and-whisker plot(s) of the given (grouped) values.

boxplot(x, ...)

## S3 method for class 'formula'
boxplot(formula, data = NULL, ..., subset, na.action = NULL)

## Default S3 method:
boxplot(x, ..., range = 1.5, width = NULL, varwidth = FALSE,
        notch = FALSE, outline = TRUE, names, plot = TRUE,
        border = par("fg"), col = NULL, log = "",
        pars = list(boxwex = 0.8, staplewex = 0.5, outwex = 0.5),
        horizontal = FALSE, add = FALSE, at = NULL)
```

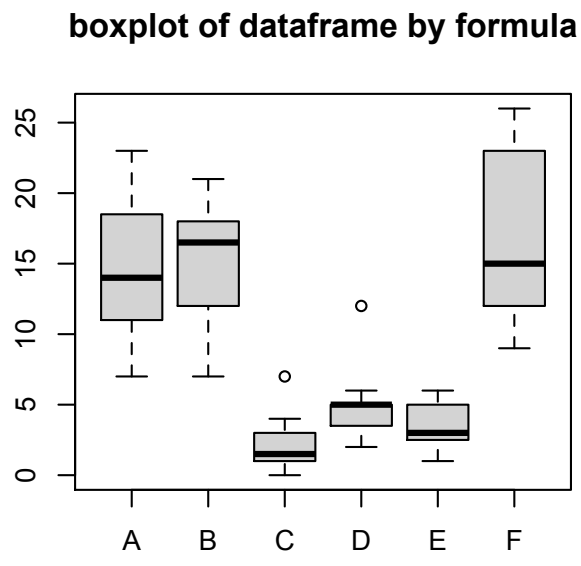
Example (boxplot 1)

```
> par(mfrow=c(2,2))
> norm1 <- round(rnorm(100,3,2), digits=2)
> norm2 <- round(rnorm(100,3,3), digits=2)
> boxplot(norm1)
> title("boxplot of one vector")
> boxplot(norm1, norm2)
> title("boxplot of two vector")
> list1 <- list(data1=norm1, data2=norm2, data3=rnorm(100,7,4))
> boxplot(list1)
> title("boxplot of simple list")
> dimnames(InsectSprays)
[[1]]
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14"
[15] "15" "16" "17" "18" "19" "20" "21" "22" "23" "24" "25" "26" "27" "28"
[29] "29" "30" "31" "32" "33" "34" "35" "36" "37" "38" "39" "40" "41" "42"
[43] "43" "44" "45" "46" "47" "48" "49" "50" "51" "52" "53" "54" "55" "56"
[57] "57" "58" "59" "60" "61" "62" "63" "64" "65" "66" "67" "68" "69" "70"
[71] "71" "72"

[[2]]
[1] "count" "spray"

> dim(InsectSprays)
[1] 72  2
> boxplot(count~spray, data=InsectSprays, col="lightgray")
> title("boxplot of dataframe by formula")
```


boxplot of one vector
8
6
4
2
0
-2
-4

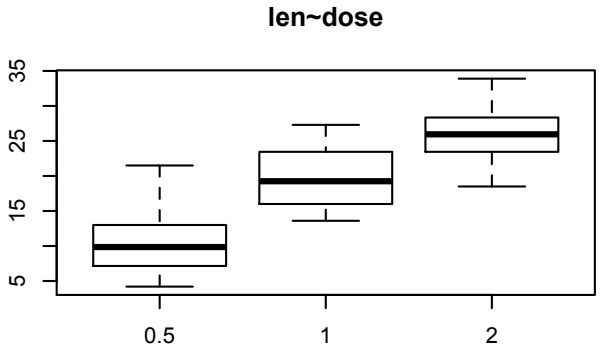

boxplot of two vector
10
5
0
-5
1
2

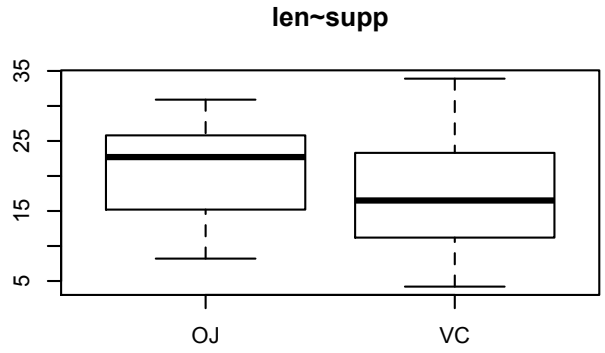

boxplot of simple list
15
10
5
0
-5
data1
data2
data3

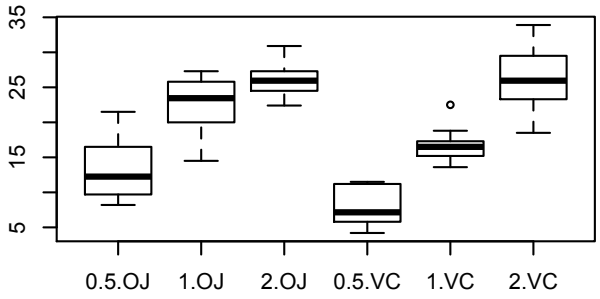

boxplot of dataframe by formula
25
20
15
10
5
0
A
B
C
D
E
F

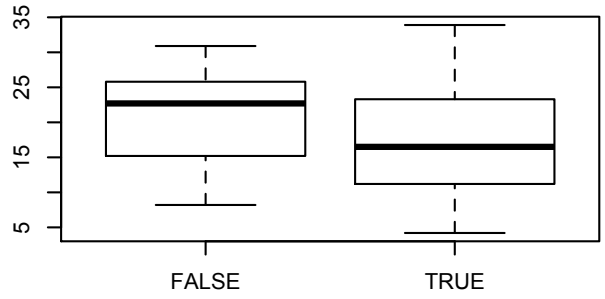
Example (boxplot 2)

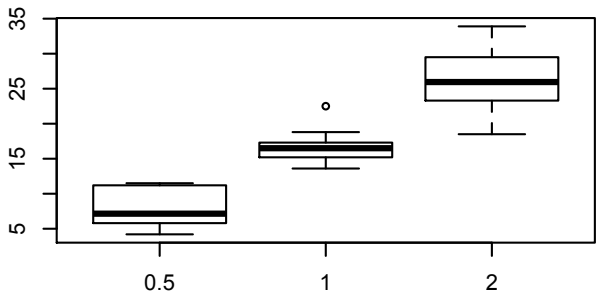
```
> par(mfrow=c(3,2))
> boxplot(len~dose, data=ToothGrowth)
> title("len~dose")
> boxplot(len~supp, data=ToothGrowth)
> title("len~supp")
> boxplot(len~dose+supp, data=ToothGrowth)
> title("len~dose+supp")
> boxplot(len~supp=="VC", data=ToothGrowth)
> title("len~supp==\"VC\"")
> boxplot(len~dose, data=ToothGrowth, subset=supp=="VC")
> title("len~dose, subset=supp==\"VC\"")
> boxplot(len[supp=="VC"]~dose[supp=="VC"], data=ToothGrowth)
> title("len[supp==\"VC\"]~dose[supp==\"VC\"]")
>
# library(help="datasets")
```

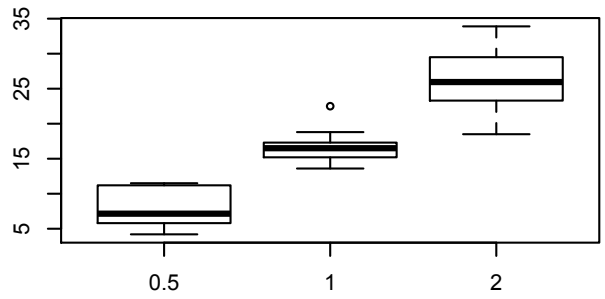

len~dose
5
15
25
35
0.5
1
2


len~supp
5
15
25
35
OJ
VC


len~dose+supp
5
15
25
35
0.5.OJ
1.OJ
2.OJ
0.5.VC
1.VC
2.VC

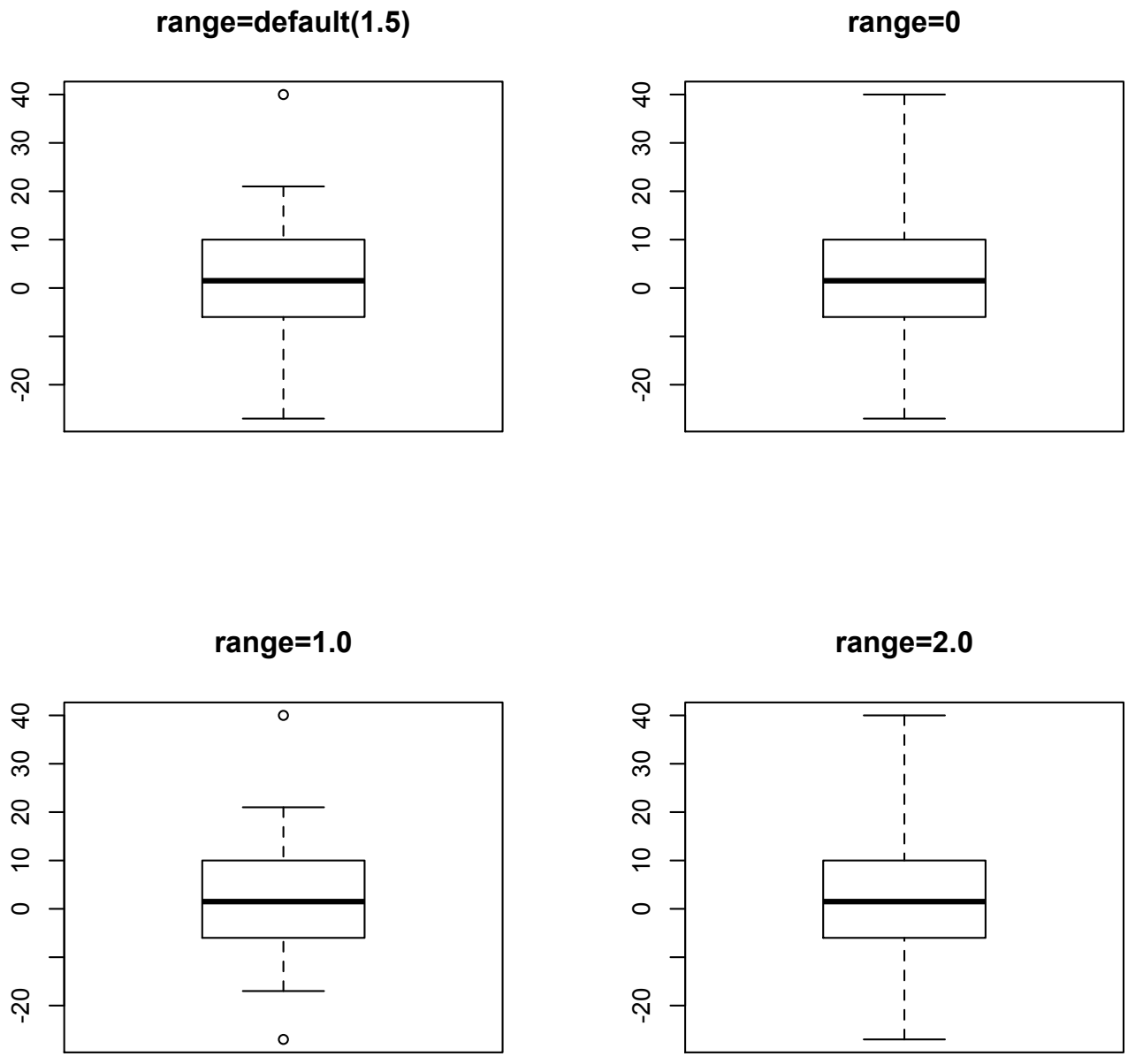

len~supp=="VC"
5
15
25
35
FALSE
TRUE


len~dose, subset=supp=="VC"
5
15
25
35
0.5
1
2


len[supp=="VC"]~dose[supp=="VC"]
5
15
25
35
0.5
1
2

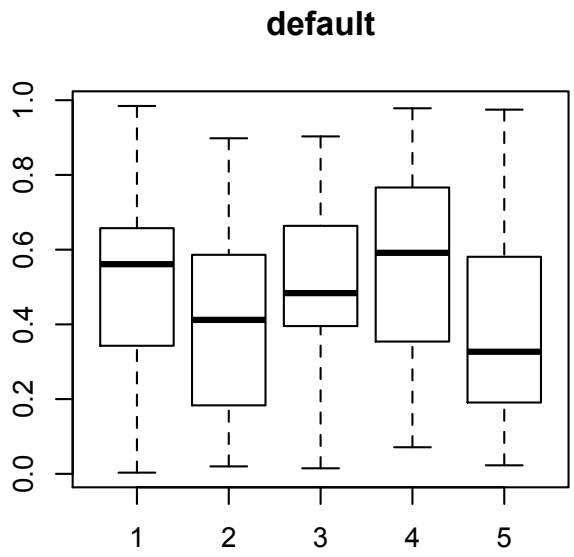
Example (boxplot 3)

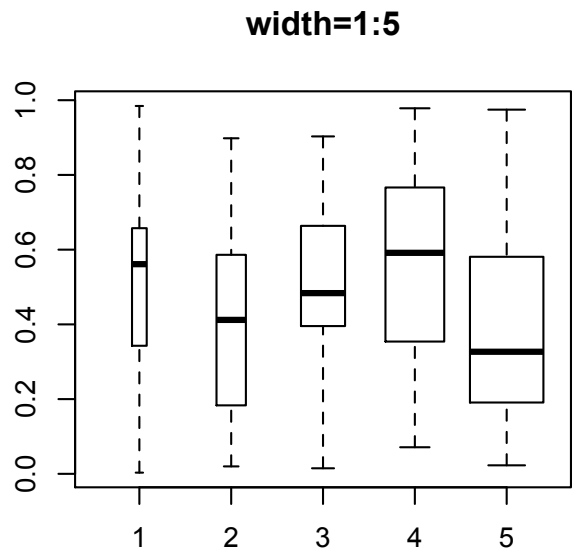
```
> z <- round(rnorm(50)*10)
> summary(z)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -27.00   -6.00    1.00    1.16    8.75   21.00
> z[50] <- 40
> summary(z)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -27.00   -5.50    1.50    2.08    9.75   40.00
> par(mfrow=c(2,2))
> boxplot(z)
> title(main="range=default(1.5)")
> boxplot(z, range=0)
> title(main="range=0")
> boxplot(z, range=1.0)
> title(main="range=1.0")
> boxplot(z, range=2.0)
> title(main="range=2.0")
```

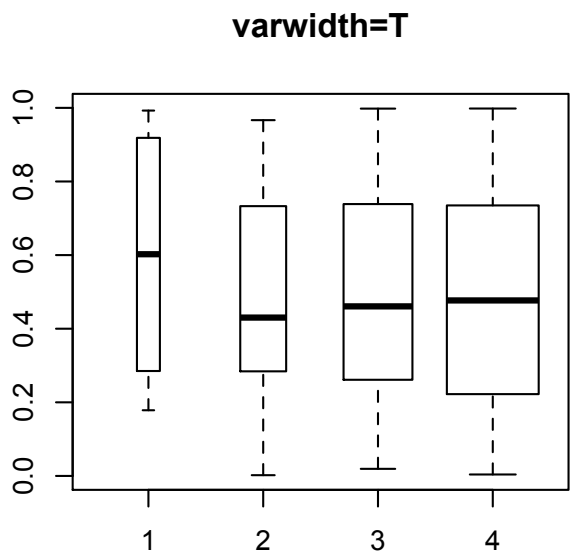

range=default(1.5)
range=0
range=1.0
range=2.0
-20
0
10
20
30
40

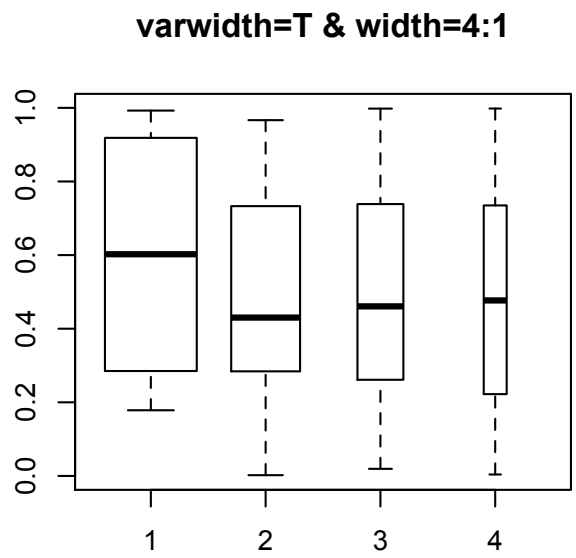
Example (boxplot 4)

```
> x1 <- runif(20)
> x2 <- runif(20)
> x3 <- runif(20)
> x4 <- runif(20)
> x5 <- runif(20)
> x <- list(x1,x2,x3,x4,x5)
> y1 <- runif(10)
> y2 <- runif(40)
> y3 <- runif(90)
> y4 <- runif(160)
> y <- list(y1,y2,y3,y4)
> par(mfrow=c(2,2))
> boxplot(x)
> title(main="default")
> boxplot(x, width=1:5)
> title(main="width=1:5")
> boxplot(y, varwidth=T)
> title(main="varwidth=T")
> boxplot(y, varwidth=T, width=4:1)
> title(main="varwidth=T & width=4:1")
```


default
0.0
0.2
0.4
0.6
0.8
1.0
1
2
3
4
5


width=1:5
0.0
0.2
0.4
0.6
0.8
1.0
1
2
3
4
5

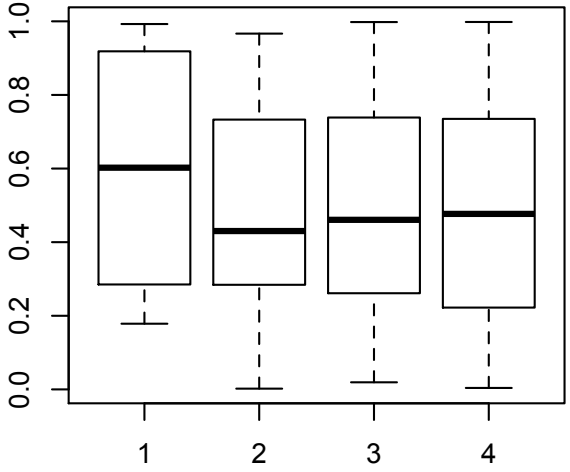

varwidth=T
0.0
0.2
0.4
0.6
0.8
1.0
1
2
3
4


varwidth=T & width=4:1
0.0
0.2
0.4
0.6
0.8
1.0
1
2
3
4

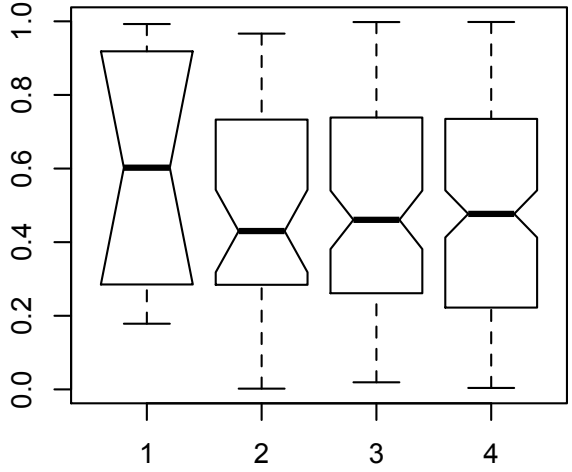
Example (boxplot 5)

```
> par(mfrow=c(2,2))
> boxplot(y)
> title(main="notch=default(FALSE)")
> boxplot(y, notch=T, main="notch=TRUE")
경고 메시지가 손실되었습니다
In bxp(list(stats = c(0.178457653382793, 0.285076956730336, 0.60234840225894,  :
  some notches went outside hinges ('box'): maybe set notch=FALSE
> boxplot(z, main="outline=default(TRUE)")
> boxplot(z, outline=F, main="outline=FALSE")
```

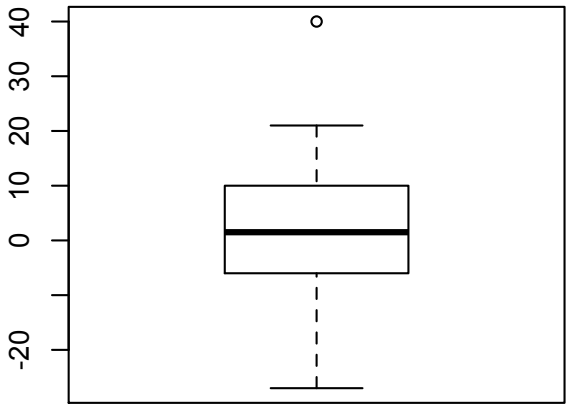
notch=default(FALSE)



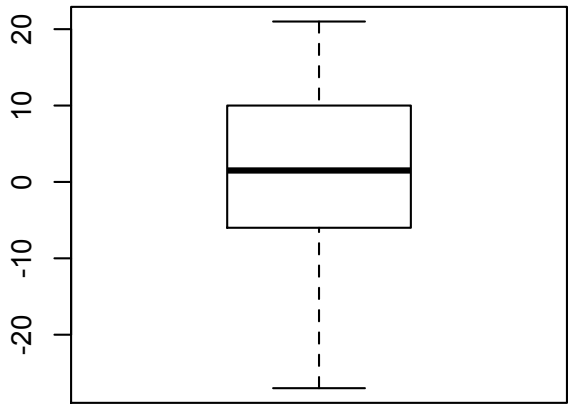
notch=TRUE



outline=default(TRUE)

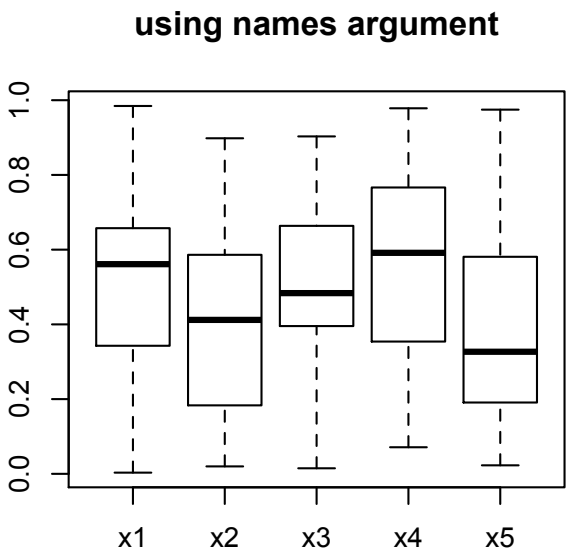


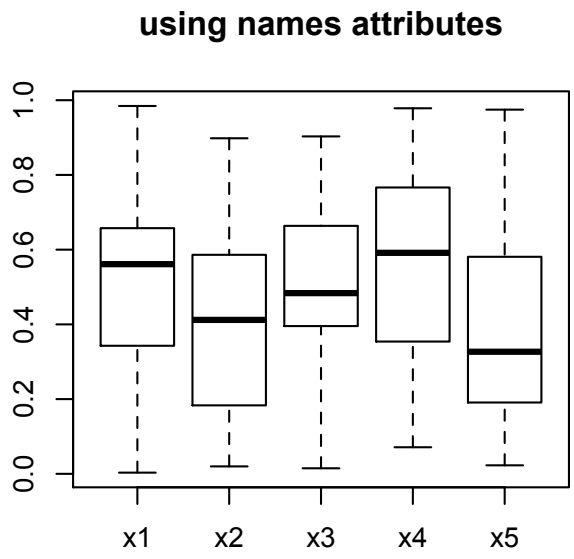
outline=FALSE

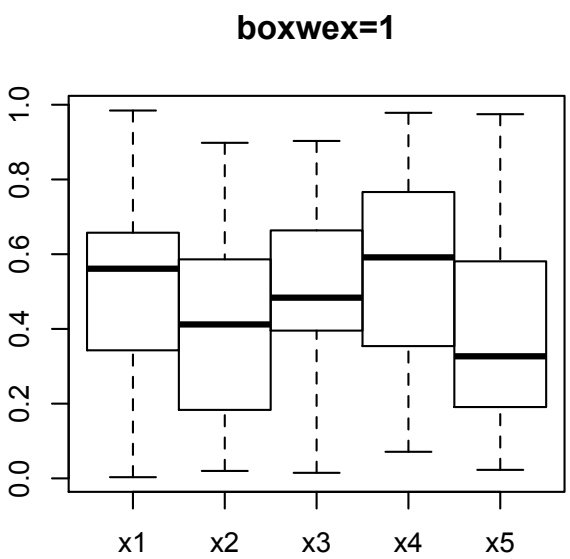


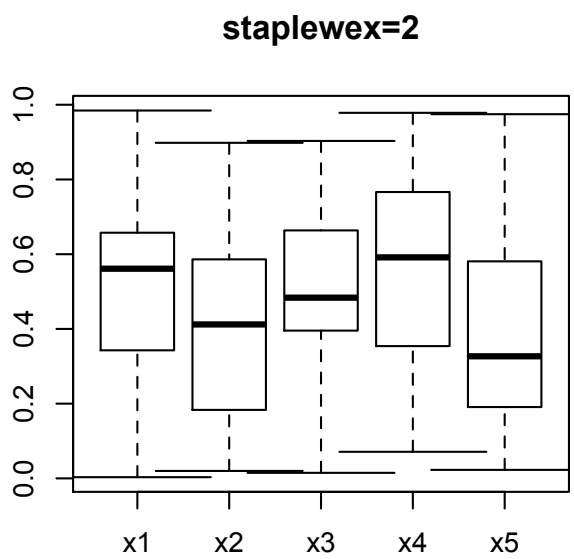
Example (boxplot 6)

```
> par(mfrow=c(2,2))
> xname <- c("x1","x2","x3","x4","x5")
> boxplot(x, names=xname)
> title(main="using names argument")
> names(x) <- c("x1","x2","x3","x4","x5")
> boxplot(x)
> title(main="using names attributes")
> boxplot(x, boxwex=1)
> title(main="boxwex=1")
> boxplot(x, staplewex=2)
> title(main="staplewex=2")
```


using names argument
0.0
0.2
0.4
0.6
0.8
1.0
x1
x2
x3
x4
x5


using names attributes
0.0
0.2
0.4
0.6
0.8
1.0
x1
x2
x3
x4
x5


boxwex=1
0.0
0.2
0.4
0.6
0.8
1.0
x1
x2
x3
x4
x5


staplewex=2
0.0
0.2
0.4
0.6
0.8
1.0
x1
x2
x3
x4
x5

Example (boxplot 7)

```
> boxplot(x, plot=FALSE)
$stats
            [,1]       [,2]      [,3]       [,4]       [,5]
[1,] 0.003196024 0.01990981 0.0148445 0.07114875 0.02278012
[2,] 0.342756815 0.18327271 0.3954680 0.35404932 0.19089682
[3,] 0.561147936 0.41200136 0.4836502 0.59151918 0.32675711
[4,] 0.657495861 0.58619521 0.6636964 0.76640035 0.58089197
[5,] 0.984530546 0.89810180 0.9030898 0.97837070 0.97472234

$n
[1] 20 20 20 20 20

$conf
          [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 0.4499510 0.2696493 0.3888854 0.4458361 0.1889723
[2,] 0.6723448 0.5543534 0.5784150 0.7372023 0.4645419

$out
numeric(0)

$group
numeric(0)

$names
[1] "x1" "x2" "x3" "x4" "x5"
```

stats
a matrix, each column contains the extreme of the lower whisker, the lower hinge, the median, the upper hinge and the extreme of the upper whisker for one group/plot. If all the inputs have the same class attribute, so will this component.

n
a vector with the number of observations in each group.

conf
a matrix where each column contains the lower and upper extremes of the notch.

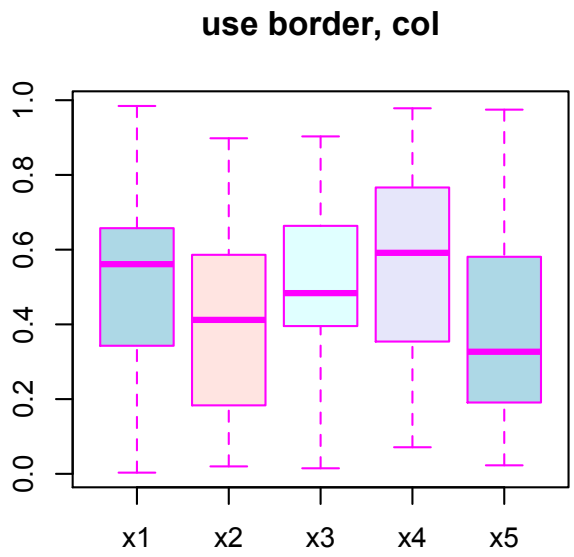
out
the values of any data points which lie beyond the extremes of the whiskers.

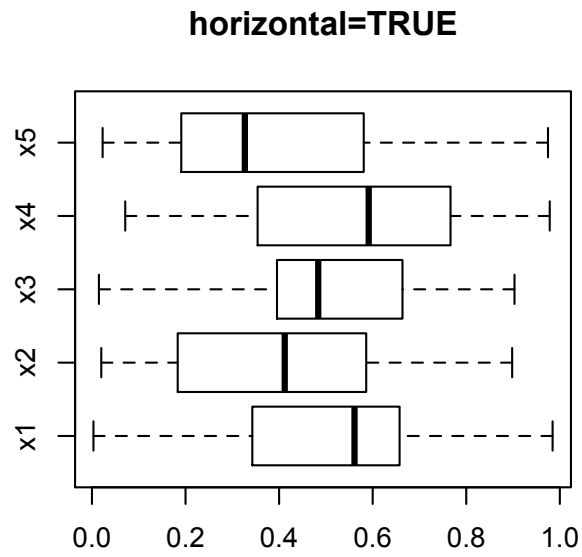
group
a vector of the same length as out whose elements indicate to which group the outlier belongs.

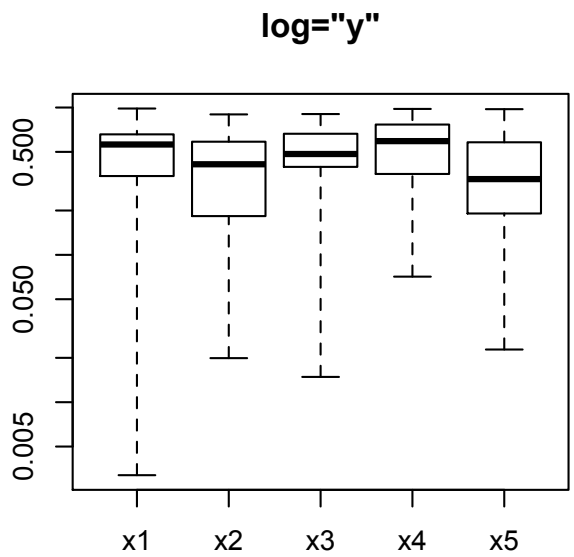
names
a vector of names for the groups.

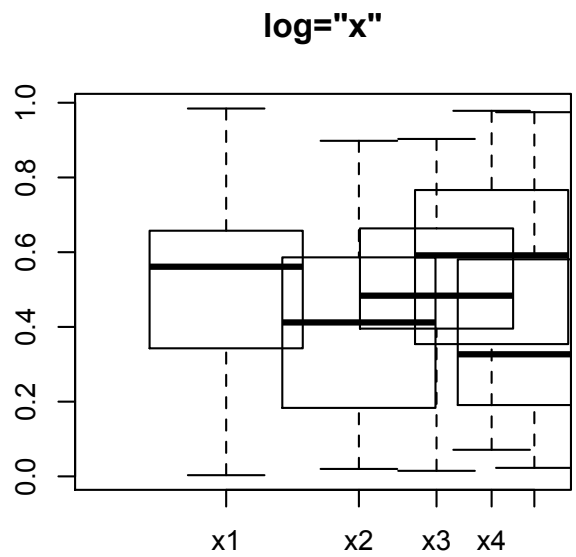
Example (boxplot 8)

```
> par(mfrow=c(2,2))
> boxplot(x, border="magenta", col=c("lightblue","mistyrose","lightcyan","lavender"))
> title(main="use border, col")
> boxplot(x, horizontal=TRUE)
> title(main="horizontal=TRUE")
> boxplot(x, log="y", main="log=\"y\"")
> boxplot(x, log="x", main="log=\"x\"")
```


use border, col
0.0
0.2
0.4
0.6
0.8
1.0
x1
x2
x3
x4
x5

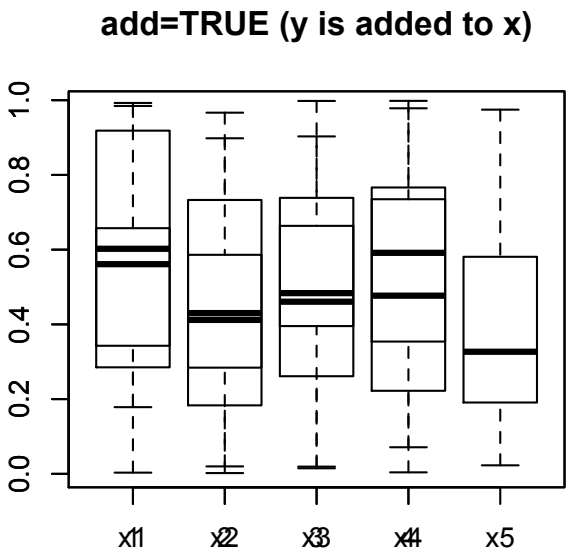

horizontal=TRUE
x1
x2
x3
x4
x5
0.0
0.2
0.4
0.6
0.8
1.0

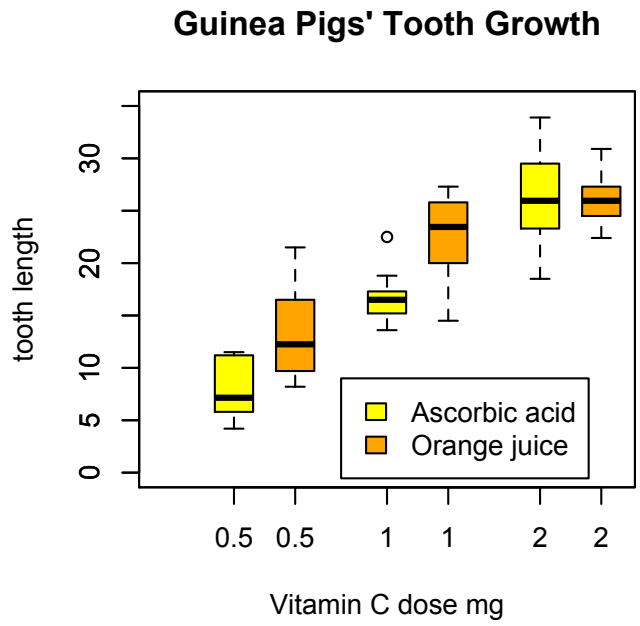

log="y"
0.005
0.050
0.500
x1
x2
x3
x4
x5

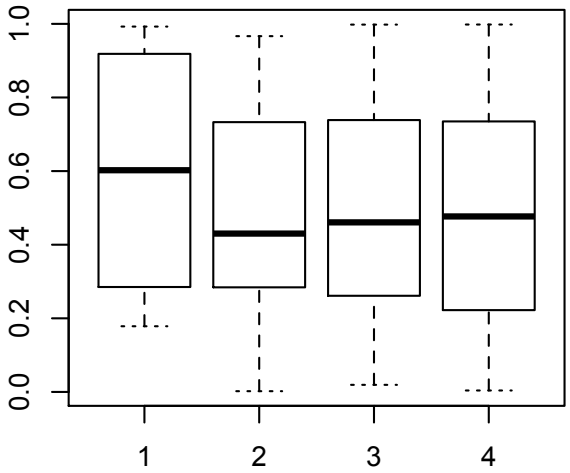

log="x"
0.0
0.2
0.4
0.6
0.8
1.0
x1
x2
x3
x4

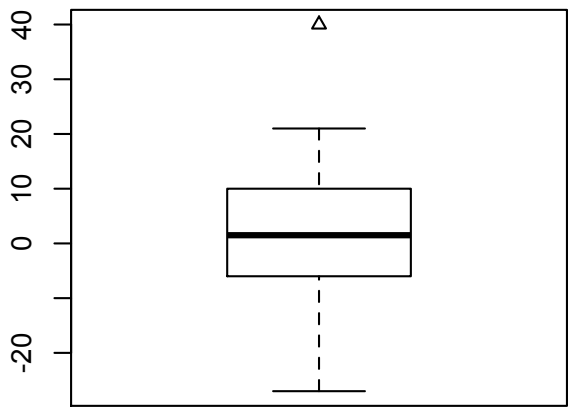
Example (boxplot 9)

```
> par(mfrow=c(2,2))
> boxplot(x)
> boxplot(y, add=TRUE)
> title(main="add=TRUE (y is added to x)")
> 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", ylim=c(0,35))
> boxplot(len~dose, data=ToothGrowth, add=TRUE, boxwex=0.25, at=1:3 +0.2,
+         subset=supp=="OJ", col="orange")
> legend(1.5, 9, c("Ascorbic acid", "Orange juice"), fill=c("yellow","orange"))
> boxplot(y, staplelty=3)
> title(main="staplelty=3")
> boxplot(z, outpch=2)
> title(main="outpch=2")
```


add=TRUE (y is added to x)
0.0
0.2
0.4
0.6
0.8
1.0
x1
x2
x3
x4
x5


Guinea Pigs' Tooth Growth
tooth length
0
5
10
20
30
0.5
0.5
1
1
2
2
Ascorbic acid
Orange juice
Vitamin C dose mg


staplelty=3
0.0
0.2
0.4
0.6
0.8
1.0
1
2
3
4


outpch=2
-20
0
10
20
30
40

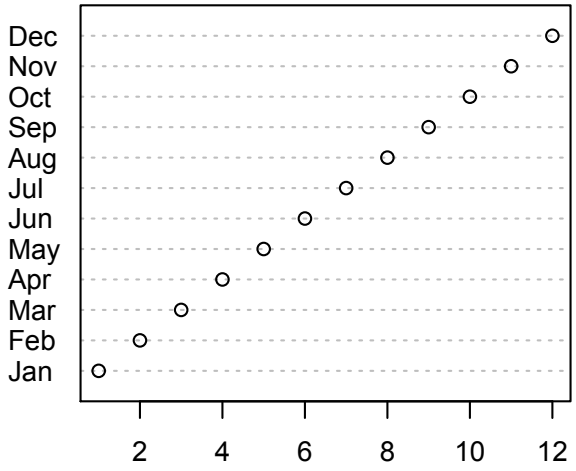
Definition (dotchart)

```
# Draw a Cleveland dot plot.

dotchart(x, labels = NULL, groups = NULL, gdata = NULL,
         cex = par("cex"), pch = 21, gpch = 21, bg = par("bg"),
         color = par("fg"), gcolor = par("fg"), lcolor = "gray",
         xlim = range(x[is.finite(x)]),
         main = NULL, xlab = NULL, ylab = NULL, ...)
```

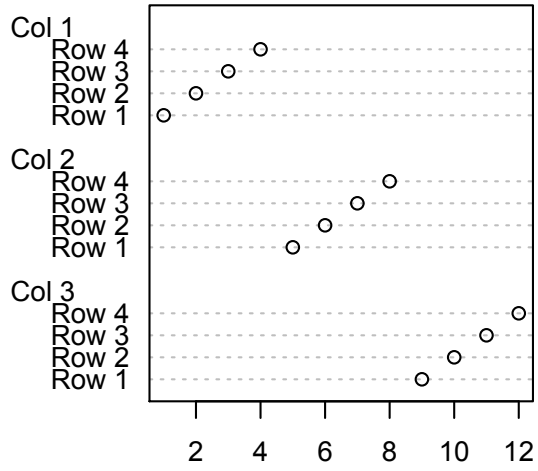
Example (dotchart 1)

```
> par(mfrow=c(2,2))
> month <- matrix(1:12, ncol=3)
> month
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
> dotchart(as.vector(month), label=month.abb)
> title(main="x is a vector")
> rownames(month) <- paste("Row", 1:4)
> colnames(month) <- paste("Col", 1:3)
> dotchart(month)
> title(main="x is a matrix")
> quarter.name <- c("1QT","2QT","3QT","4QT")
> quarter <- factor(row(month), label=quarter.name)
> quarter
 [1] 1QT 2QT 3QT 4QT 1QT 2QT 3QT 4QT 1QT 2QT 3QT 4QT
Levels: 1QT 2QT 3QT 4QT
> dotchart(month, groups=quarter)
> title(main="groups=quarter")
> name <- c("1st", "2nd", "3rd")
> dotchart(month, groups=quarter, labels=name)
> title(main="groups=quarter, labels=name")
```

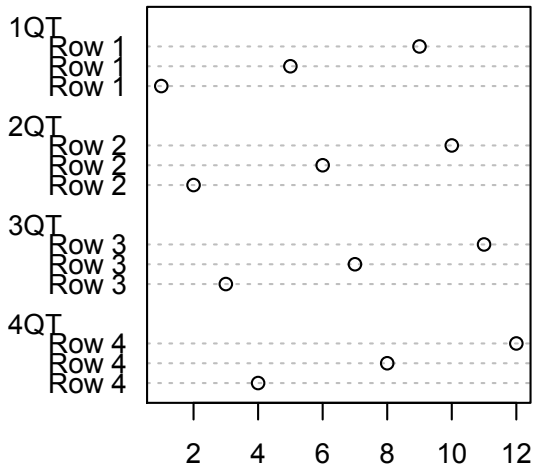
x is a vector



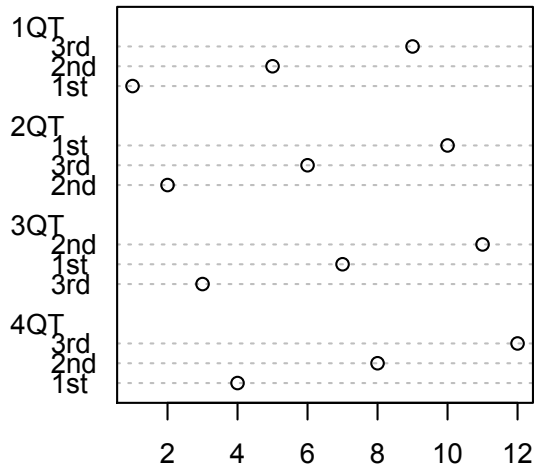
x is a matrix



groups=quarter

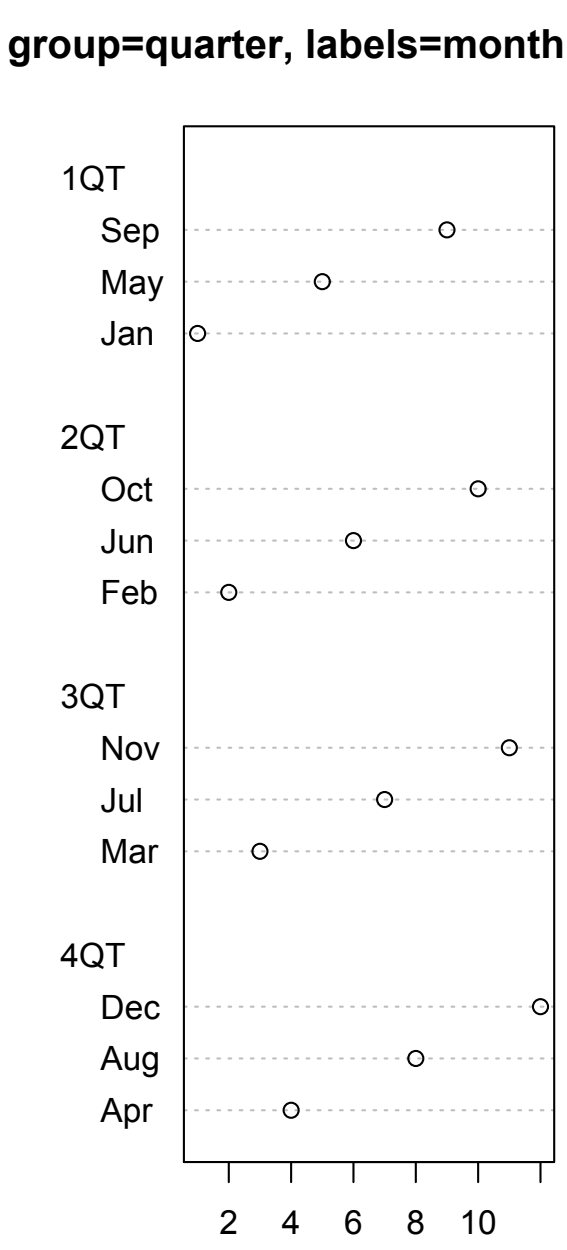


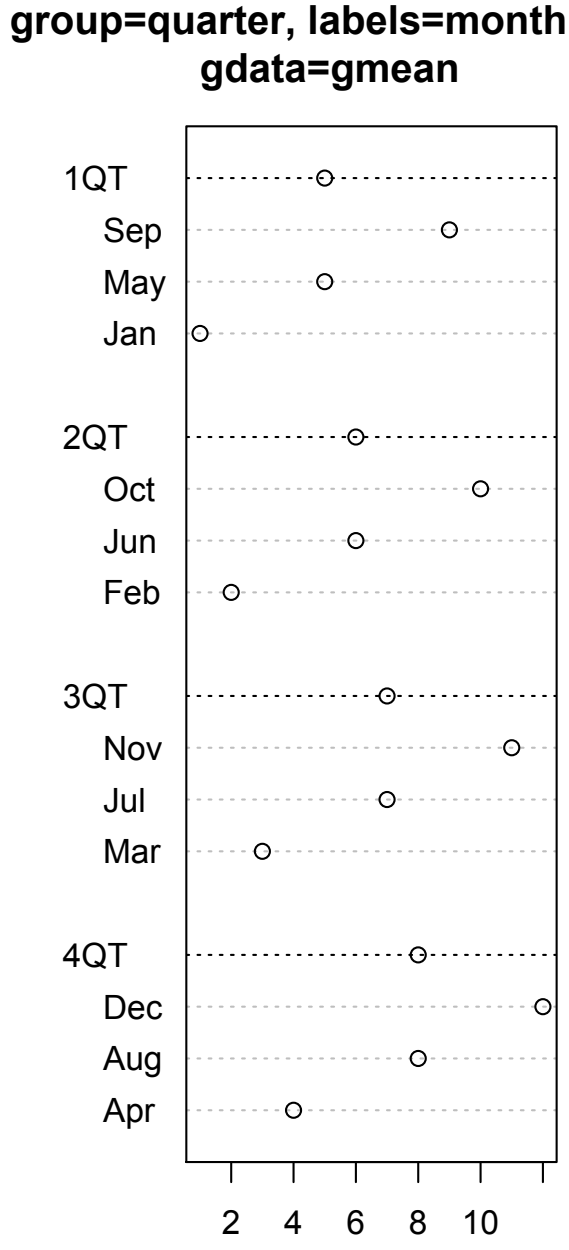
groups=quarter, labels=name



Example (dotchart 2)

```
> par(mfrow=c(1,2))
> dotchart(month, group=quarter, labels=month.abb)
> title(main="group=quarter, labels=month.abb")
> gmean <- tapply(month, quarter, mean)
> gmean
1QT 2QT 3QT 4QT
  5   6   7   8
> dotchart(month, group=quarter, labels=month.abb, gdata=gmean)
> title(main="group=quarter, labels=month.abb\ngdata=gmean")
```

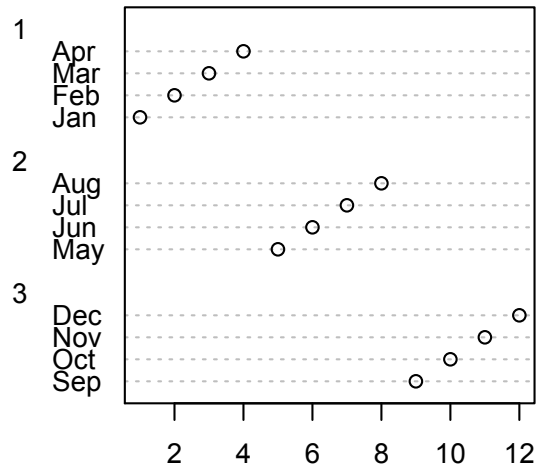

group=quarter, labels=month.abb
1QT
Sep
May
Jan
2QT
Oct
Jun
Feb
3QT
Nov
Jul
Mar
4QT
Dec
Aug
Apr
2
4
6
8
10


group=quarter, labels=month.abb
gdata=gmean
1QT
Sep
May
Jan
2QT
Oct
Jun
Feb
3QT
Nov
Jul
Mar
4QT
Dec
Aug
Apr
2
4
6
8
10

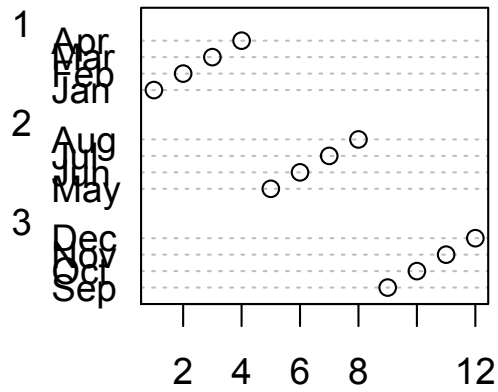
Example (dotchart 3)

```
> par(mfrow=c(2,2))
> dotchart(month, labels=month.abb, main="default cex")
> dotchart(month, labels=month.abb, cex=1.1, main="cex=1.1")
> dotchart(month, labels=month.abb, pch=2, main="pch=2")
> dotchart(month, labels=month.abb, group=quarter, pch=2, gpch=5, gdata=gmean)
> title(main="pch=2, gpch=5")
```

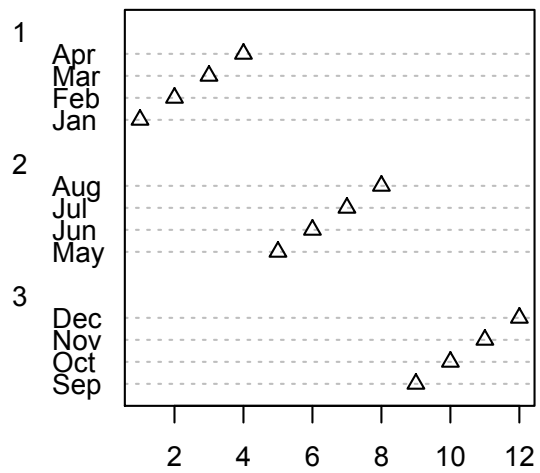
default cex



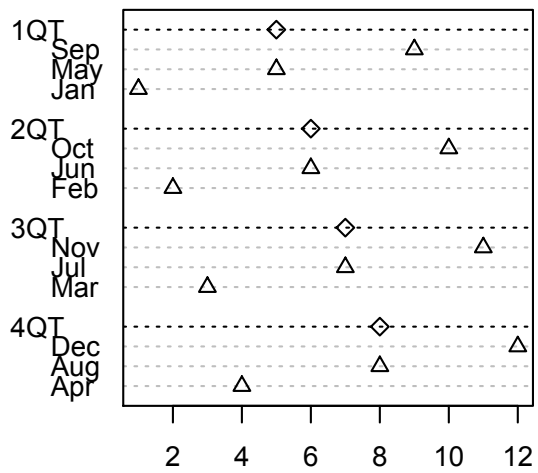
cex=1.1



pch=2



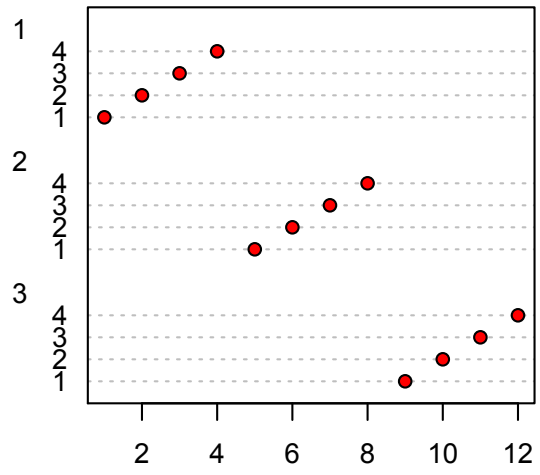
pch=2, gpch=5



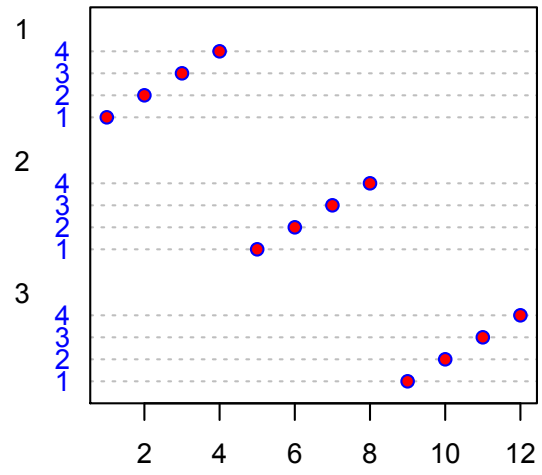
Example (dotchart 3)

```
> par(mfrow=c(2,2))
> dotchart(month, bg="red")
> title(main="bg=\"red\"")
> dotchart(month, bg="red", color="blue")
> title(main="bg=\"red\", color=\"blue\"")
> dotchart(month, color="red", gcolor="blue", groups=quarter, gdata=gmean)
> title(main="color=\"red\", gcolor=\"blue\"")
> dotchart(month, lcolor="red", gcolor="blue", groups=quarter, gdata=gmean)
> title(main="lcolor=\"red\", gcolor=\"blue\"")
```

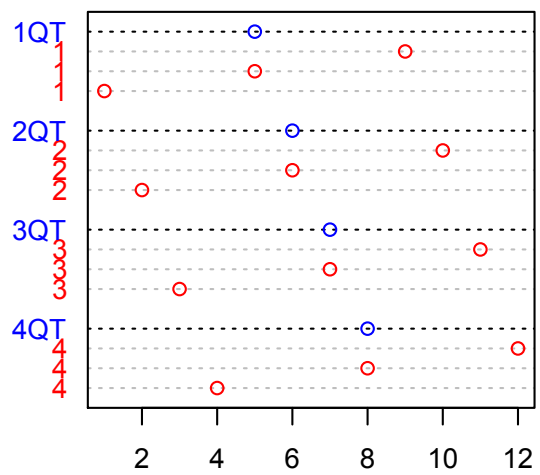
bg="red"



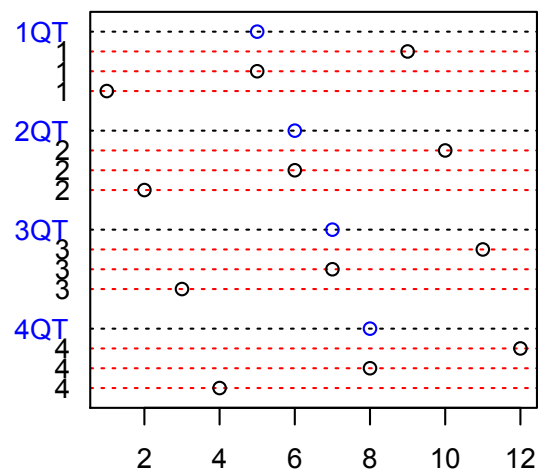
bg="red", color="blue"



color="red", gcolor="blue"



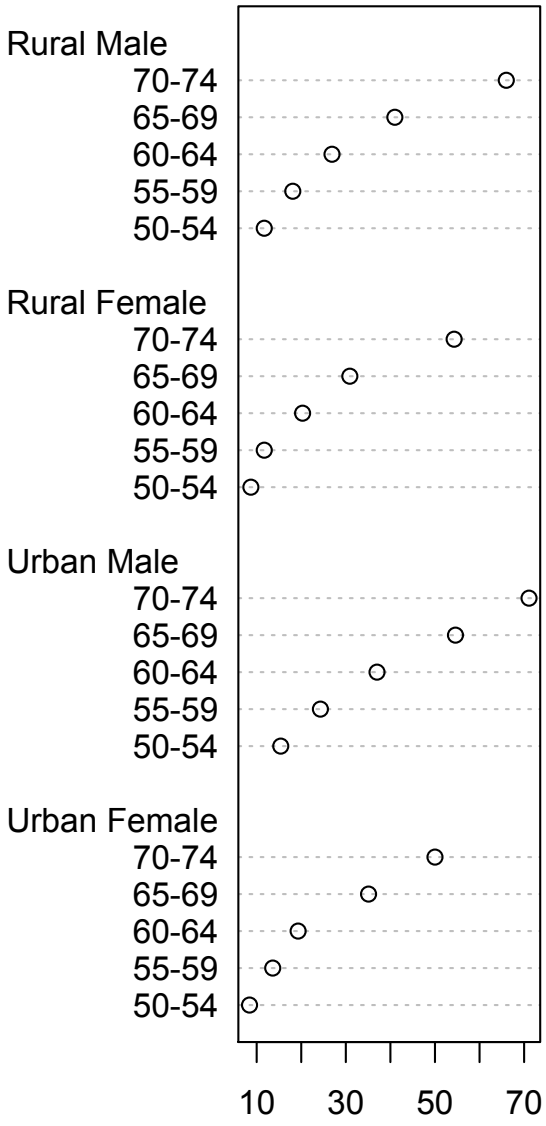
lcolor="red", gcolor="blue"



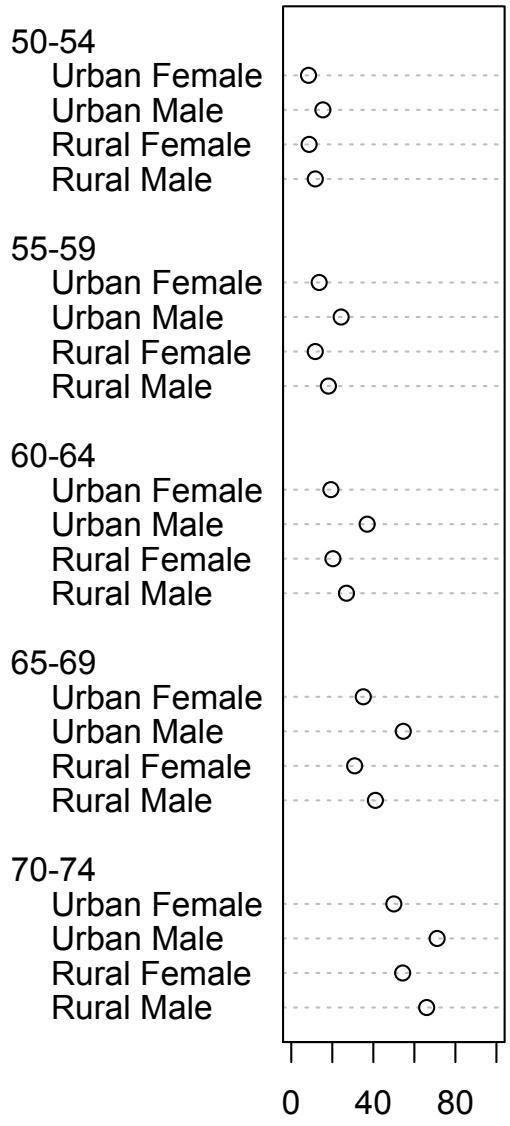
Example (dotchart 3)

```
> VADeaths
      Rural Male Rural Female Urban Male Urban Female
50-54       11.7          8.7       15.4          8.4
55-59       18.1         11.7       24.3         13.6
60-64       26.9         20.3       37.0         19.3
65-69       41.0         30.9       54.6         35.1
70-74       66.0         54.3       71.1         50.0
> par(mfrow=c(1,2))
> dotchart(VADeaths)
> title(main="Death Rates in Virginia\n(Population group)")
> dotchart(t(VADeaths), xlim=c(0,100))
> title(main="Death Rates in Virginia\n(Age group)")
```

Death Rates in Virginia (Population group)



Death Rates in Virginia (Age group)

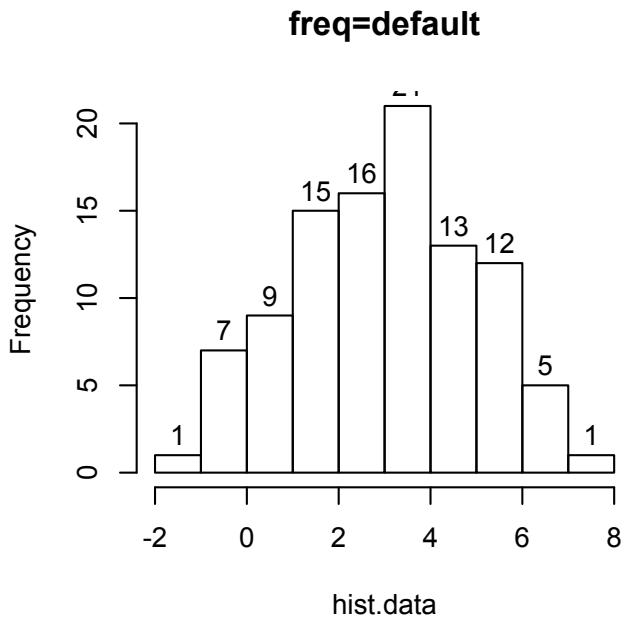


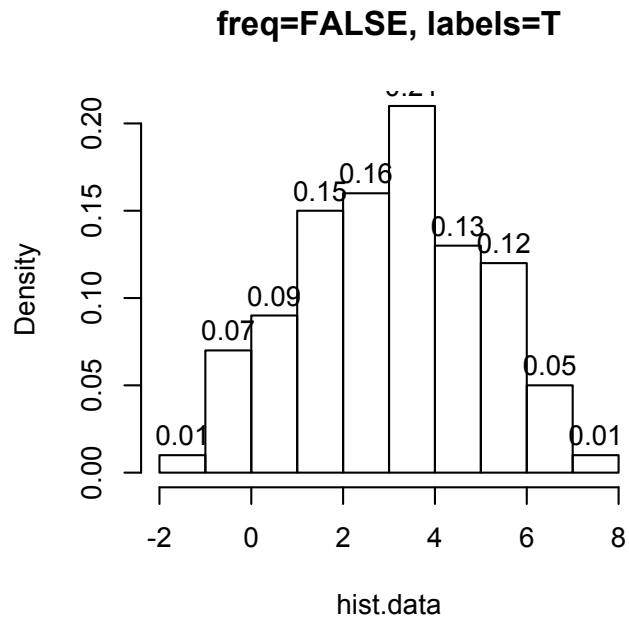
Definition (hist)

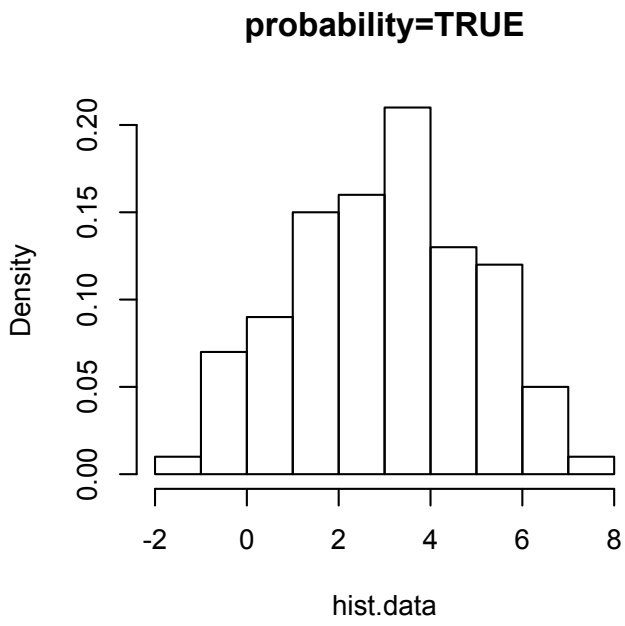
```
# The generic function hist computes a histogram of the given data values.
# If plot=TRUE, the resulting object of class "histogram" is plotted by
# plot.histogram, before it is returned.

hist(x, ...)

## Default S3 method:
hist(x, breaks = "Sturges",
     freq = NULL, probability = !freq,
     include.lowest = TRUE, right = TRUE,
     density = NULL, angle = 45, col = NULL, border = NULL,
     main = paste("Histogram of" , xname),
     xlim = range(breaks), ylim = NULL,
     xlab = xname, ylab,
     axes = TRUE, plot = TRUE, labels = FALSE,
     nclass = NULL, warn.unused = TRUE, ...)
```

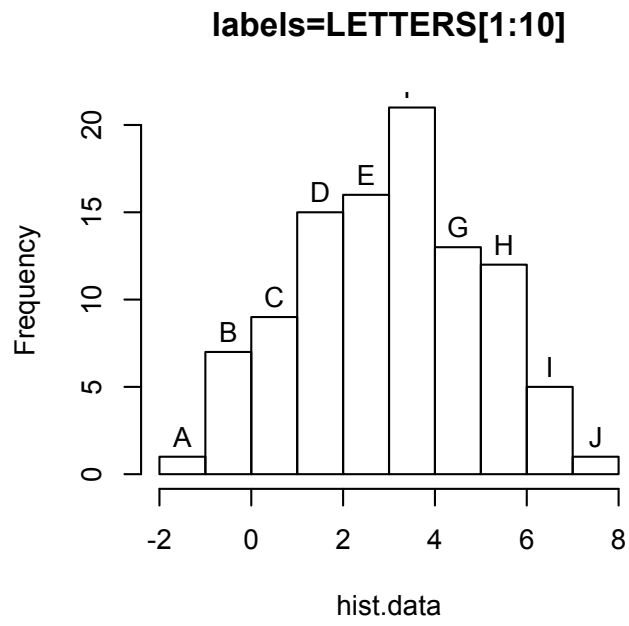
Example (hist 1)

```
> hist.data <- rnorm(100, 3, 2)
> hist.data <- round(hist.data, digits=2)
> par(mfrow=c(2,2))
> hist(hist.data, labels=T, main="freq=default")
> hist(hist.data, freq=F, labels=T, main="freq=FALSE, labels=T")
> hist(hist.data, probability=TRUE, main="probability=TRUE")
> hist(hist.data, labels=LETTERS[1:10], main="labels=LETTERS[1:10]")
```


freq=default
Frequency
hist.data
1
7
9
15
16
13
12
5
1

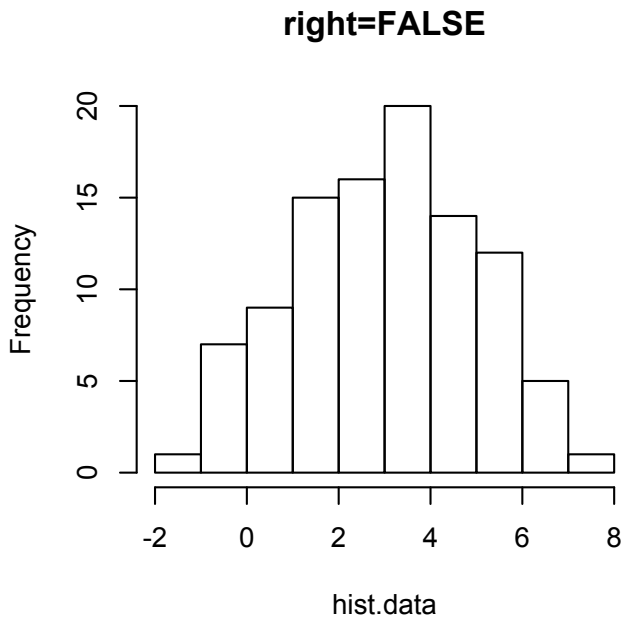

freq=FALSE, labels=T
Density
hist.data
0.01
0.07
0.09
0.15
0.16
0.13
0.12
0.05
0.01

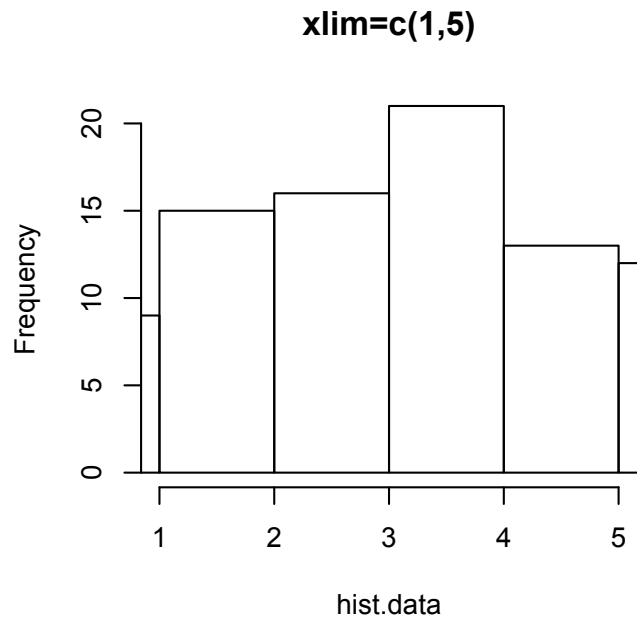

probability=TRUE
Density
hist.data

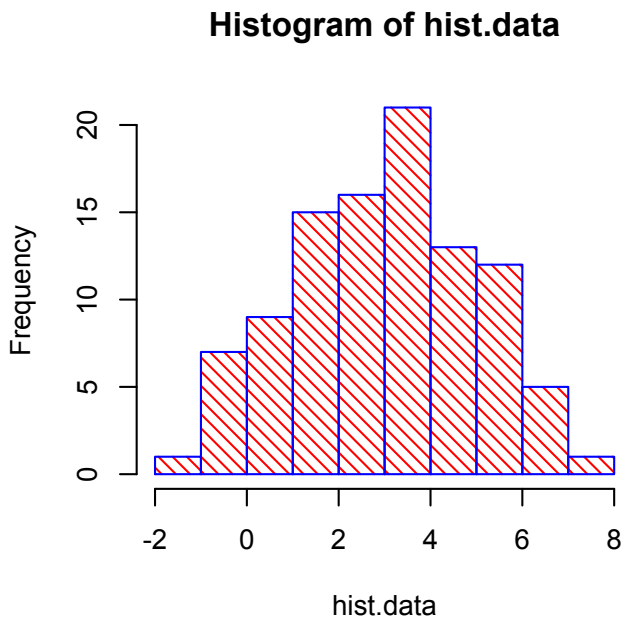

labels=LETTERS[1:10]
Frequency
hist.data
A
B
C
D
E
G
H
I
J

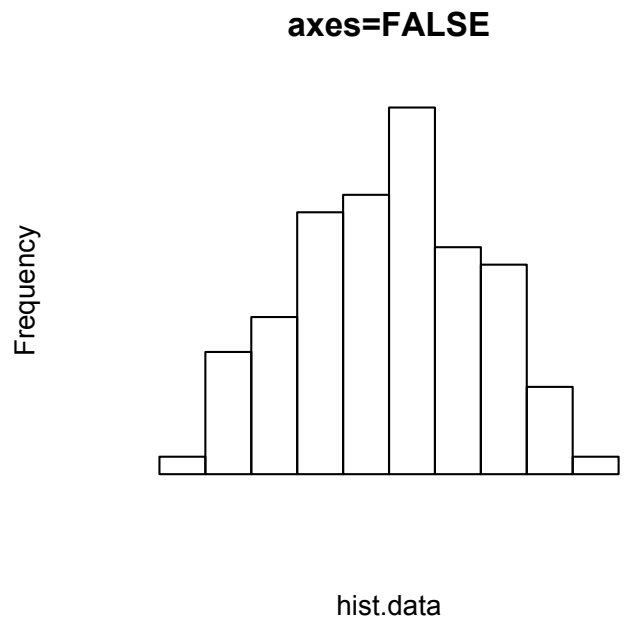
Example (hist 2)

```
> par(mfrow=c(2,2))
> hist(hist.data, right=FALSE, main="right=FALSE")
> hist(hist.data, xlim=c(1,5), main="xlim=c(1,5)")
> hist(hist.data, density=20, col="red", angle=135, border="blue")
> hist(hist.data, axes=FALSE, main="axes=FALSE")
```


right=FALSE
Frequency
0
5
10
15
20
-2
0
2
4
6
8
hist.data


xlim=c(1,5)
Frequency
0
5
10
15
20
1
2
3
4
5
hist.data


Histogram of hist.data
Frequency
0
5
10
15
20
-2
0
2
4
6
8
hist.data


axes=FALSE
Frequency
hist.data

Example (hist 3)

```
> hist(hist.data, plot=FALSE)
$breaks
 [1] -2 -1  0  1  2  3  4  5  6  7  8

$counts
 [1]  1  7  9 15 16 21 13 12  5  1

$intensities
 [1] 0.01 0.07 0.09 0.15 0.16 0.21 0.13 0.12 0.05 0.01

$density
 [1] 0.01 0.07 0.09 0.15 0.16 0.21 0.13 0.12 0.05 0.01

$mids
 [1] -1.5 -0.5  0.5  1.5  2.5  3.5  4.5  5.5  6.5  7.5

$xname
[1] "hist.data"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
```

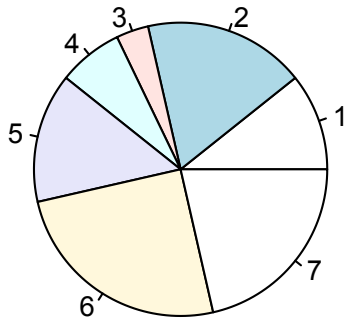
Definition (pie)

```
# Draw a pie chart.

pie(x, labels = names(x), edges = 200, radius = 0.8,
    clockwise = FALSE, init.angle = if(clockwise) 90 else 0,
    density = NULL, angle = 45, col = NULL, border = NULL,
    lty = NULL, main = NULL, ...)
```

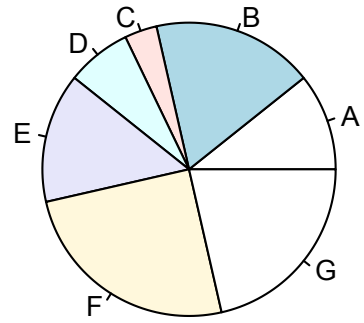
Example (pie 1)

```
> pie.data <- sample(7)
> pie.data
[1] 3 5 1 2 4 7 6
> par(mfrow=c(2,2))
> pie(pie.data, main="default")
> pie(pie.data, labels=LETTERS[1:7], main="labels=LETTERS[1:7]")
> pie(pie.data, edges=10, main="edges=10")
> pie(pie.data, edges=20, main="edges=20")
```

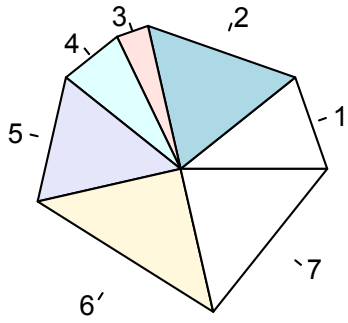
default



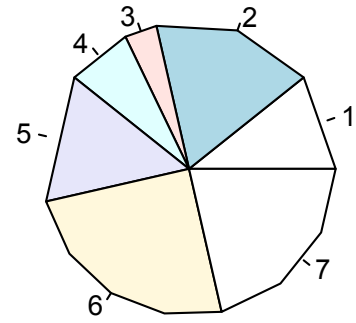
labels=LETTERS[1:7]



edges=10



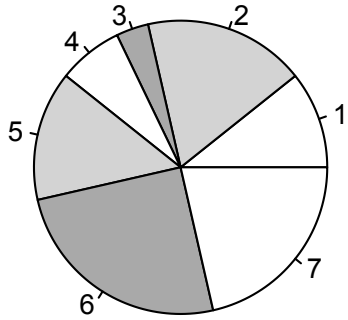
edges=20



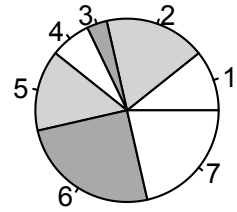
Example (pie 2)

```
> my.color=c("white", "lightgray", "darkgray")
> par(mfrow=c(2,2))
> pie(pie.data, col=my.color, main="default radius")
> pie(pie.data, col=my.color, radius=0.5, main="radius=0.5")
> pie(pie.data, col=my.color, radius=1.5, main="radius=1.5")
> pie(pie.data, col=my.color, radius=0, main="radius=0")
```

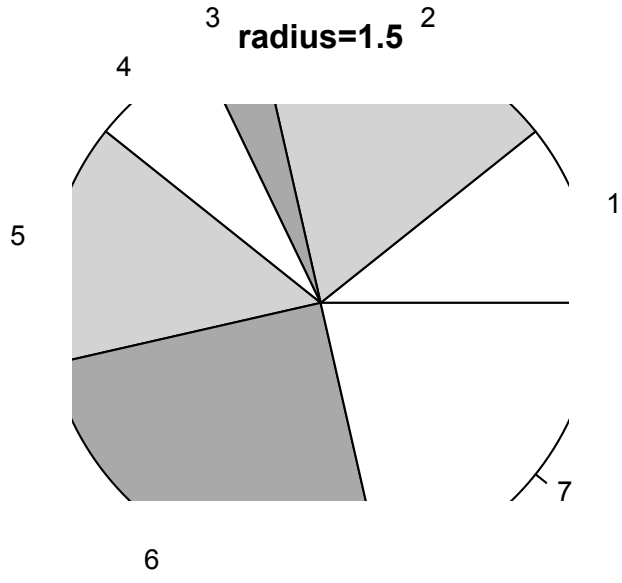
default radius



radius=0.5



radius=1.5



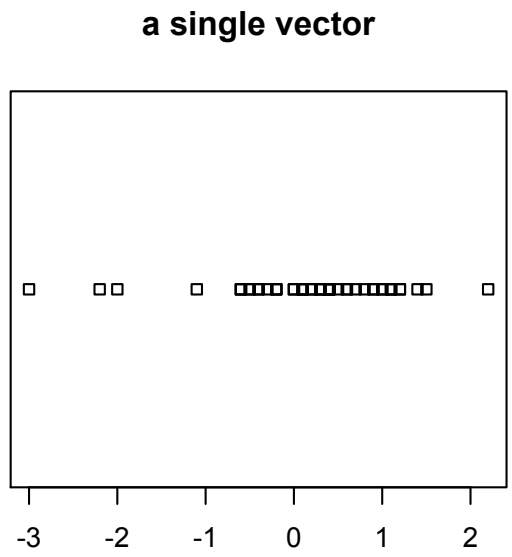
radius=0

Definition (stripchart)

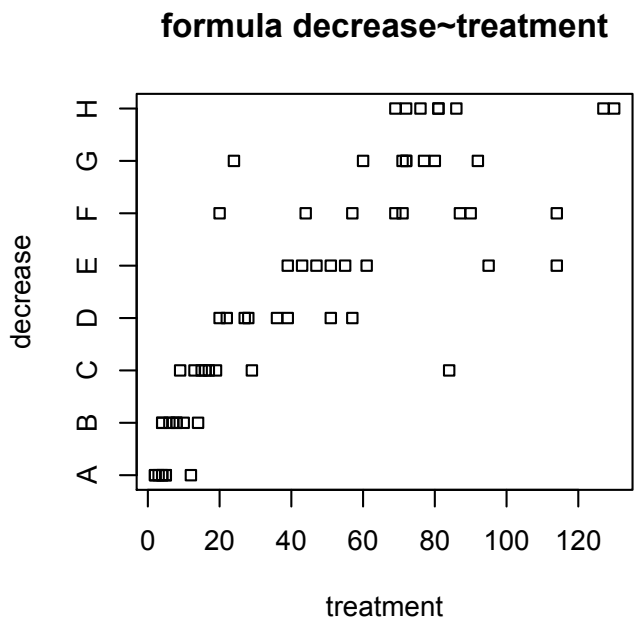
```
# stripchart produces one dimensional scatter plots (or dot plots) of the given
# data. These plots are a good alternative to boxplots when sample sizes are small.

stripchart(x, ...)

## S3 method for class 'formula'
stripchart(x, data = NULL, dlab = NULL, ...,
           subset, na.action = NULL)


## Default S3 method:
stripchart(x, method = "overplot", jitter = 0.1, offset = 1/3,
           vertical = FALSE, group.names, add = FALSE,
           at = NULL, xlim = NULL, ylim = NULL,
           ylab=NULL, xlab=NULL, dlab="", glab="",
           log = "", pch = 0, col = par("fg"), cex = par("cex"),
           axes = TRUE, frame.plot = axes, ...)
```

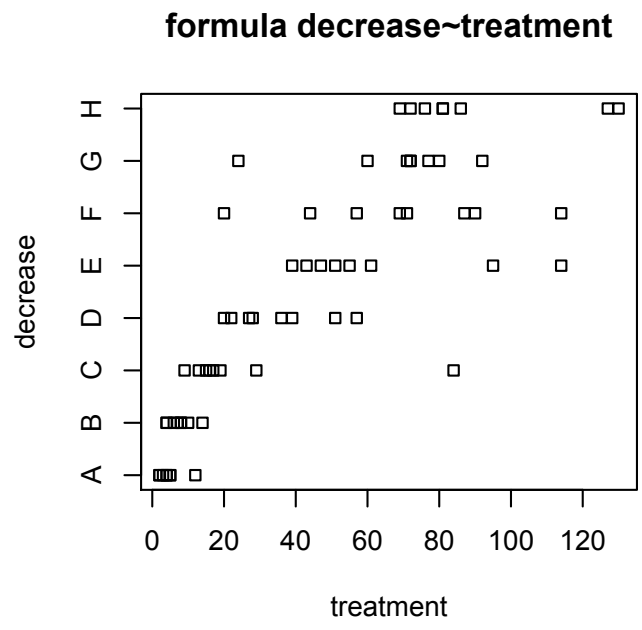
Example (stripchart 1)

```
> x <- round(rnorm(50), 1)
> y <- round(rnorm(50), 1)
> z <- round(rnorm(50), 1)
> strip.data <- list(x,y,z)
> par(mfrow=c(2,2))
> stripchart(x, main="a single vector")
> stripchart(strip.data, main="a list having 3 vectors")
> with(OrchardSprays, stripchart(decrease~treatment,
+ main="formula decrease~treatment", xlab="treatment", ylab="decrease"))
> stripchart(decrease~treatment, data=OrchardSprays,
+ main="formula decrease~treatment", xlab="treatment", ylab="decrease")
> str(OrchardSprays)
'data.frame': 64 obs. of  4 variables:
 $ decrease : num  57 95 8 69 92 90 15 2 84 6 ...
 $ rowpos   : num  1 2 3 4 5 6 7 8 1 2 ...
 $ colpos   : num  1 1 1 1 1 1 1 1 2 2 ...
 $ treatment: Factor w/ 8 levels "A","B","C","D",..: 4 5 2 8 7 6 3 1 3 2 ...
```


a single vector
-3
-2
-1
0
1
2

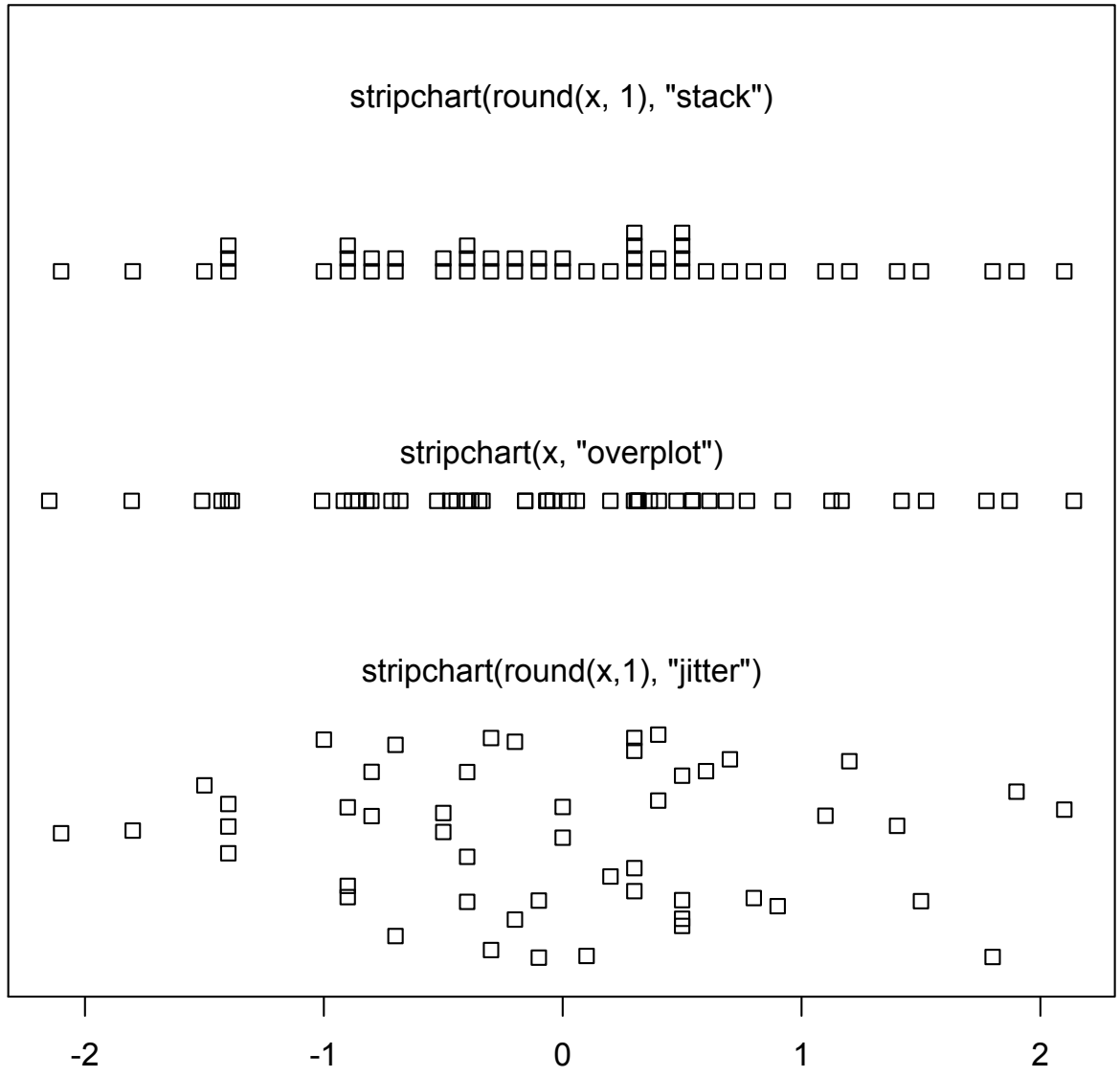

a list having 3 vectors
1
2
3
-3
-2
-1
0
1
2
3


formula decrease~treatment
decrease
A
B
C
D
E
F
G
H
0
20
40
60
80
100
120
treatment


formula decrease~treatment
decrease
A
B
C
D
E
F
G
H
0
20
40
60
80
100
120
treatment

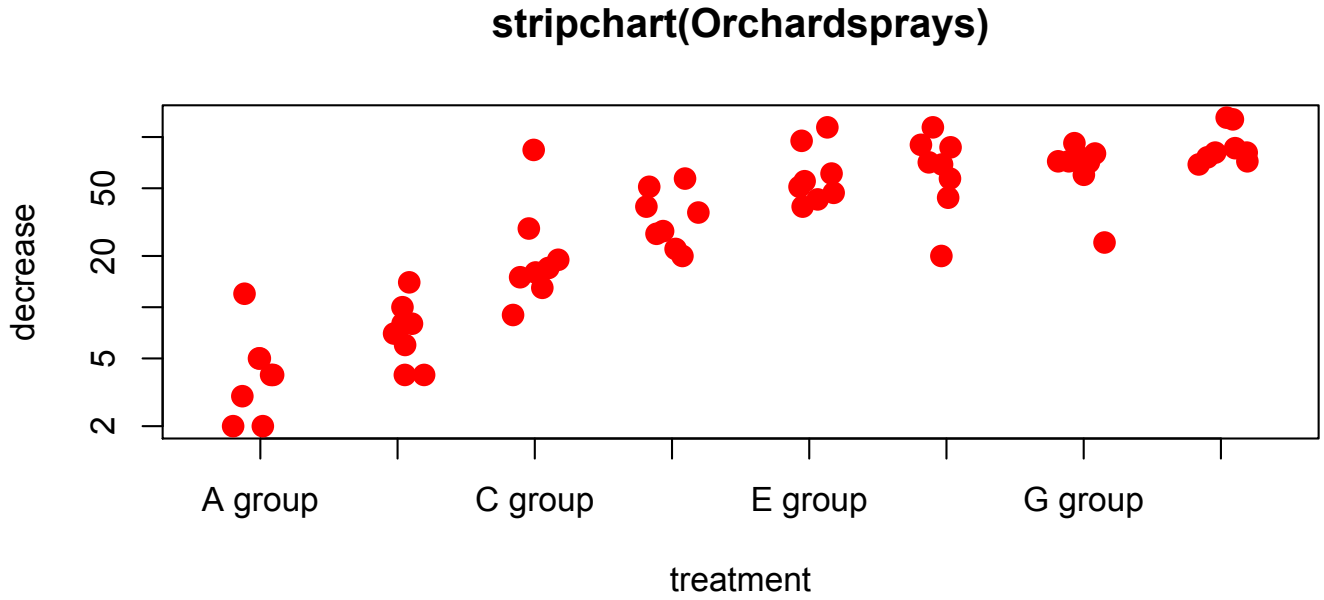
Example (stripchart 2)

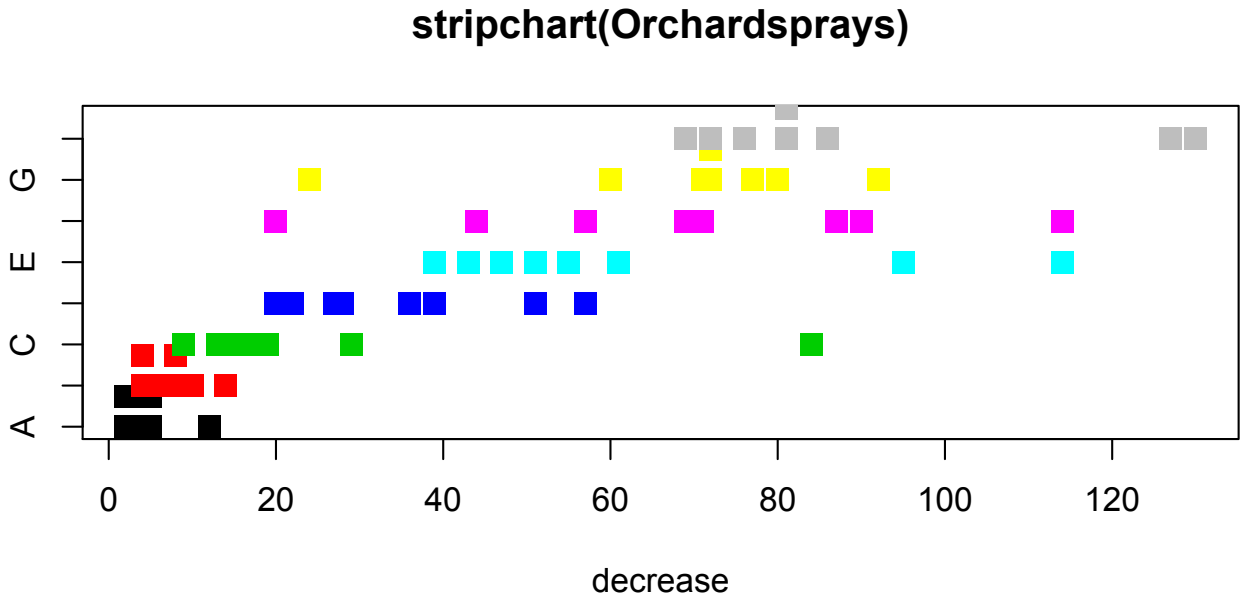
```
> x <- rnorm(50)
> xr <- round(x, 1)
> stripchart(x); m <- mean(par("usr")[1:2])
> text(m, 1.04, "stripchart(x, \"overplot\")")
> stripchart(xr, method="stack", add=TRUE, at=1.2)
> text(m, 1.35, "stripchart(round(x, 1), \"stack\")")
> stripchart(xr, method="jitter", add=TRUE, at=0.7)
> text(m, 0.85, "stripchart(round(x,1), \"jitter\")")
```


stripchart(round(x, 1), "stack")
stripchart(x, "overplot")
stripchart(round(x,1), "jitter")
-2
-1
0
1
2

Example (stripchart 3)

```
> par(mfrow=c(2,1))
> with(OrchardSprays, stripchart(decrease~treatment, method="jitter", jitter=0.2,
+ col="red", pch=16, cex=1.5, vertical=TRUE, log="y", main="stripchart(Orchardsprays)",
+ xlab="treatment", ylab="decrease", group.names=paste(LETTERS[1:8], "group")))
> with(OrchardSprays, stripchart(decrease~treatment, method="stack", offset=1/2,
+ col=1:8, pch=15, cex=1.5, main="stripchart(Orchardsprays)"))
```


stripchart(Orchardsprays)
decrease
2
5
20
50
A group
C group
E group
G group
treatment

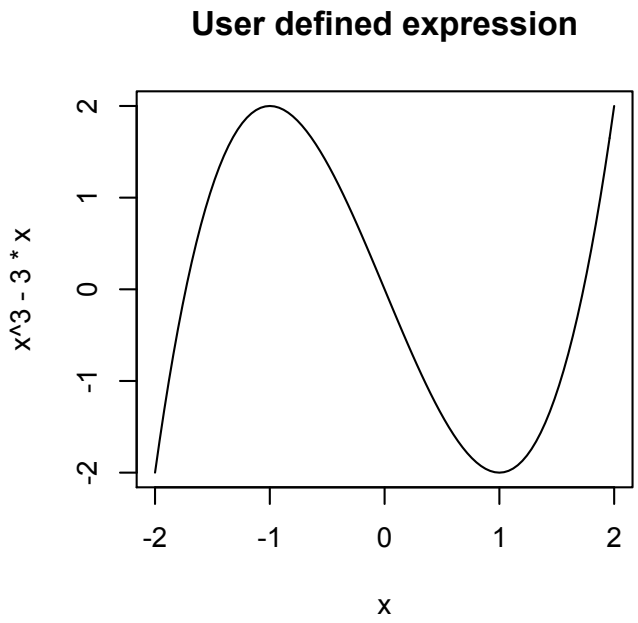

stripchart(Orchardsprays)
A
C
E
G
0
20
40
60
80
100
120
decrease

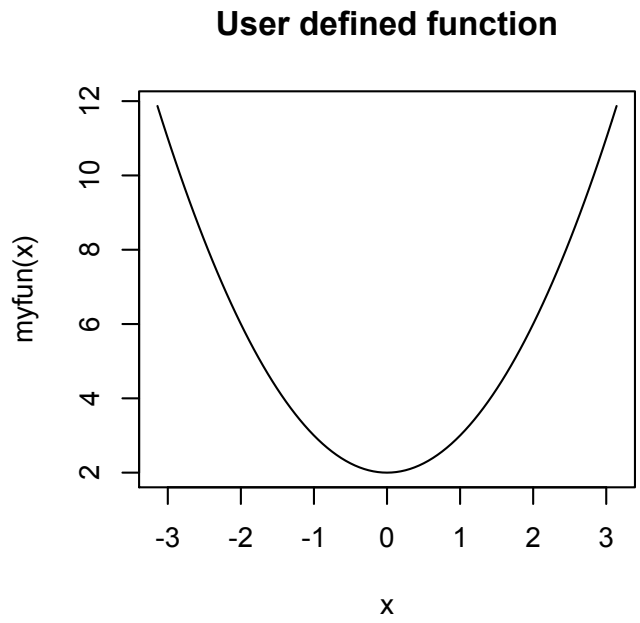
Definition (curve)

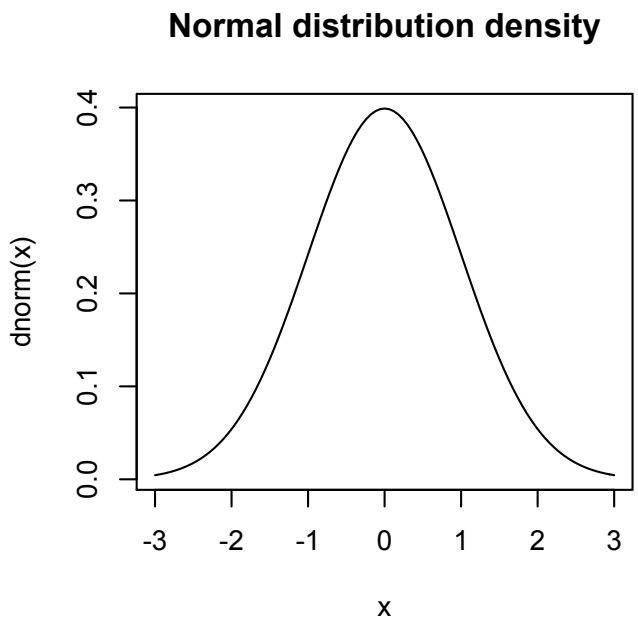
```
# Draws a curve corresponding to a function over the interval [from, to].
# curve can plot also an expression in the variable xname, default x.

curve(expr, from = NULL, to = NULL, n = 101, add = FALSE,
      type = "l", xname = "x", xlab = xname, ylab = NULL,
      log = NULL, xlim = NULL, ...)

## S3 method for class 'function'
plot(x, y = 0, to = 1, from = y, xlim = NULL, ylab = NULL, ...)
```

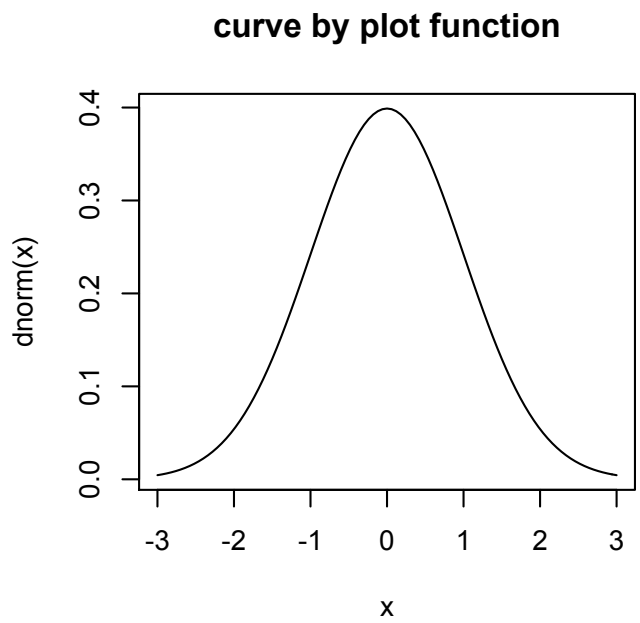
Example (curve 1)

```
> par(mfrow=c(2,2))
> curve(x^3-3*x, -2, 2)
> title(main="User defined expression")
> myfun <- function(x) x^2+2
> curve(myfun, -pi, pi)
> title(main="User defined function")
> curve(dnorm, from=-3, to=3)
> title(main="Normal distribution density")
> plot(dnorm, from=-3, to=3)
> title(main="curve by plot function")
```


User defined expression
x^3 - 3 * x
-2
-1
0
1
2
x

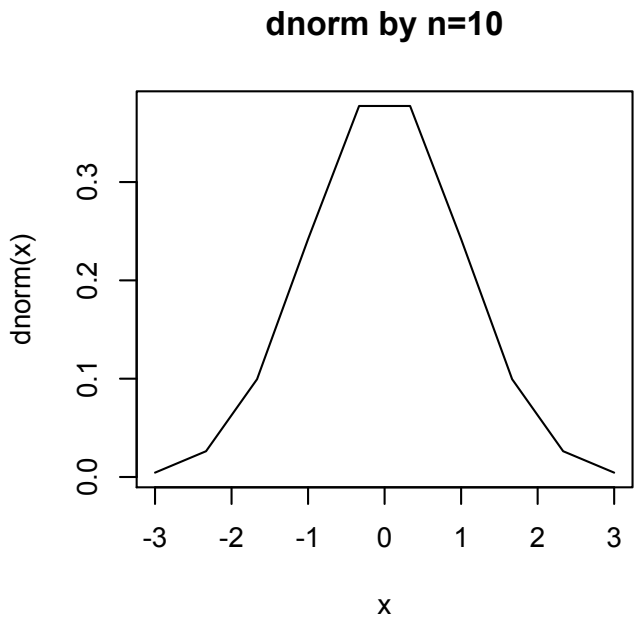

User defined function
myfun(x)
2
4
6
8
10
12
-3
-2
-1
0
1
2
3
x

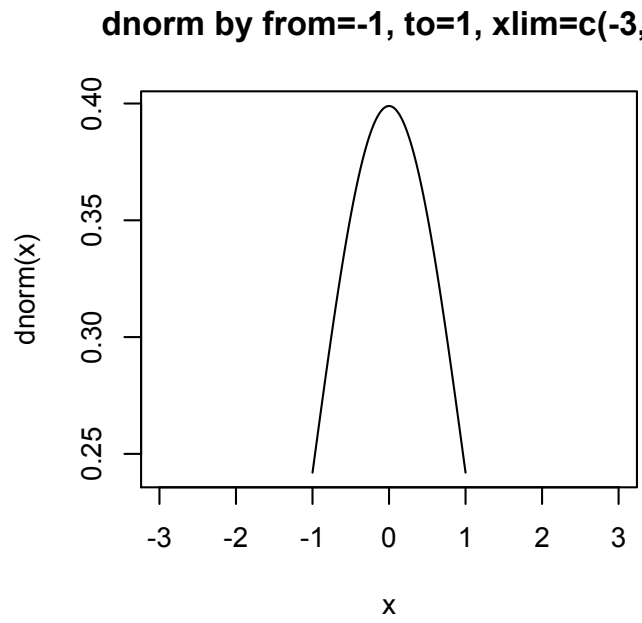

Normal distribution density
dnorm(x)
0.0
0.1
0.2
0.3
0.4
-3
-2
-1
0
1
2
3
x

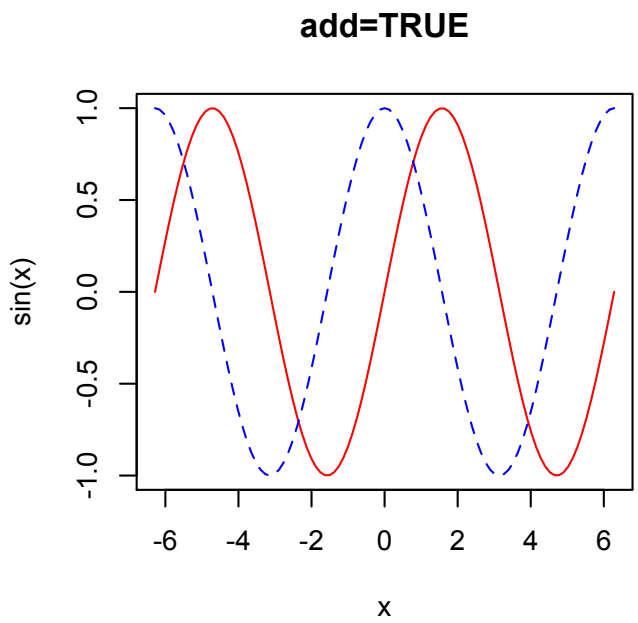

curve by plot function
dnorm(x)
0.0
0.1
0.2
0.3
0.4
-3
-2
-1
0
1
2
3
x

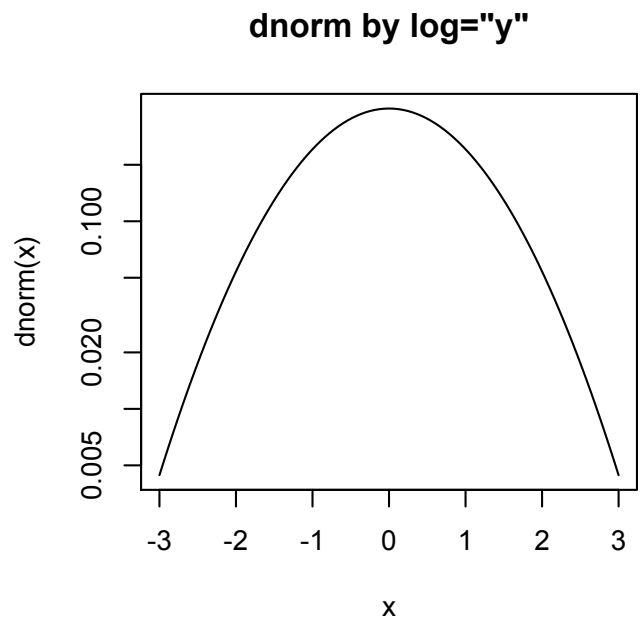
Example (curve 2)

```
> par(mfrow=c(2,2))
> curve(dnorm, from=-3, to=3, n=10)
> title(main="dnorm by n=10")
> curve(dnorm, from=-1, to=1, xlim=c(-3,3))
> title(main="dnorm by from=-1, to=1, xlim=c(-3,3)")
> curve(sin, from=-2*pi, to=2*pi, lty=1, col="red")
> curve(cos, from=-2*pi, to=2*pi, lty=2, col="blue", add=T)
> title(main="add=TRUE")
> curve(dnorm, from=-3, to=3, log="y")
> title(main="dnorm by log=\"y\"")
```


dnorm by n=10
dnorm(x)
0.0
0.1
0.2
0.3
-3
-2
-1
0
1
2
3
x


dnorm by from=-1, to=1, xlim=c(-3,3)
dnorm(x)
0.25
0.30
0.35
0.40
-3
-2
-1
0
1
2
3
x


add=TRUE
sin(x)
-1.0
-0.5
0.0
0.5
1.0
-6
-4
-2
0
2
4
6
x

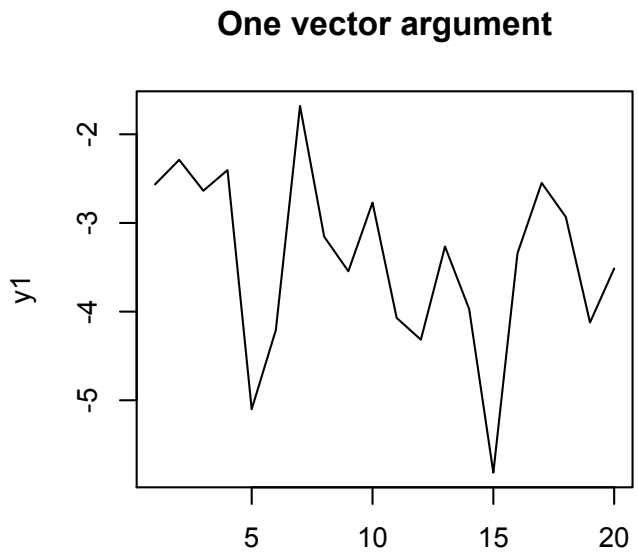

dnorm by log="y"
dnorm(x)
0.005
0.020
0.100
-3
-2
-1
0
1
2
3
x

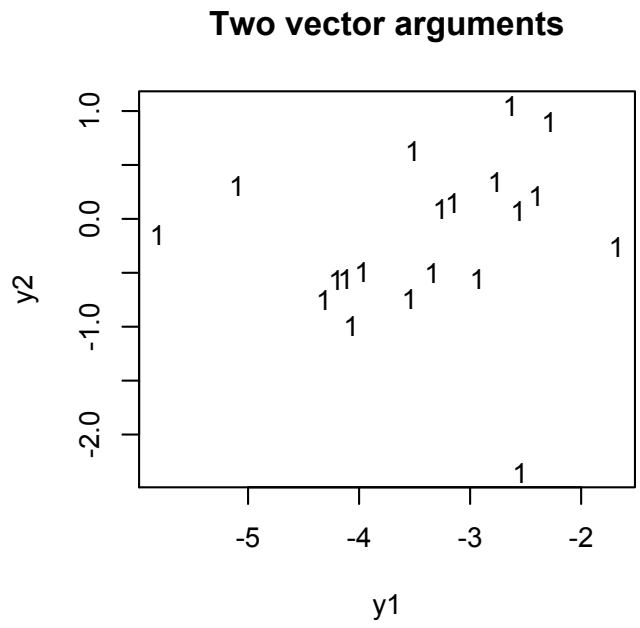
Definition (matplot, matpoints, matlines)

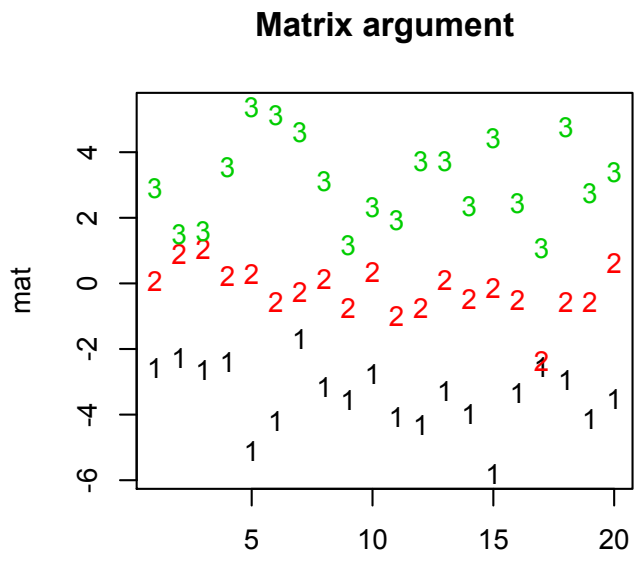
```
# Plot the columns of one matrix against the columns of another.

matplot(x, y, type = "p", lty = 1:5, lwd = 1, lend = par("lend"),
        pch = NULL,
        col = 1:6, cex = NULL, bg = NA,
        xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL,
        ..., add = FALSE, verbose = getOption("verbose"))

matpoints(x, y, type = "p", lty = 1:5, lwd = 1, pch = NULL,
          col = 1:6, ...)

matlines (x, y, type = "l", lty = 1:5, lwd = 1, pch = NULL,
          col = 1:6, ...)
```

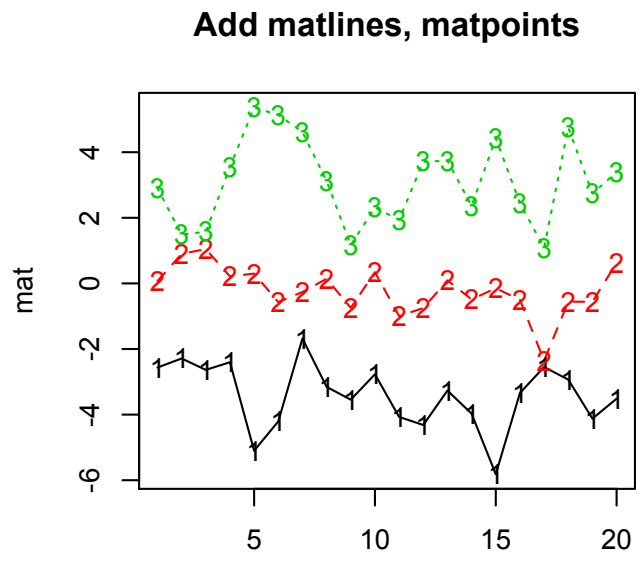
Example (matplot, matpoints, matlines 1)

```
> y1 <- rnorm(20, mean=-3, sd=1)
> y2 <- rnorm(20, mean=0, sd=1)
> y3 <- rnorm(20, mean=3, sd=1)
> mat <- cbind(y1, y2, y3)
> par(mfrow=c(2,2))
> matplot(y1, type="l", main="One vector argument")
> matplot(y1, y2, main="Two vector arguments")
> matplot(mat, main="Matrix argument")
> matplot(mat, type="n", main="Add matlines, matpoints")
> matlines(mat)
> matpoints(mat)
```


One vector argument
y1
-2
-3
-4
-5
5
10
15
20


Two vector arguments
y2
1.0
0.0
-1.0
-2.0
-5
-4
-3
-2
y1

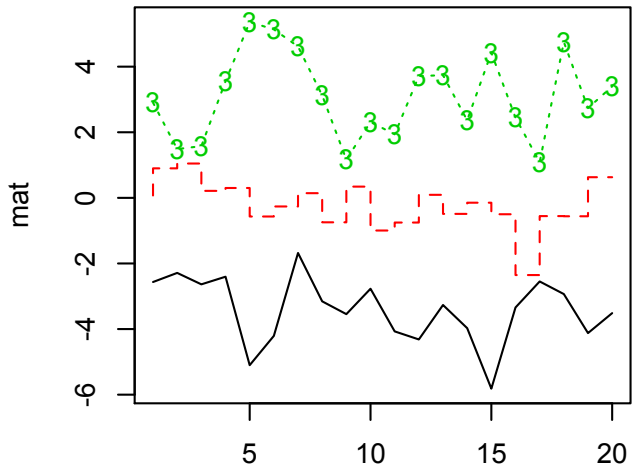

Matrix argument
mat
4
2
0
-2
-4
-6
5
10
15
20


Add matlines, matpoints
mat
4
2
0
-2
-4
-6
5
10
15
20

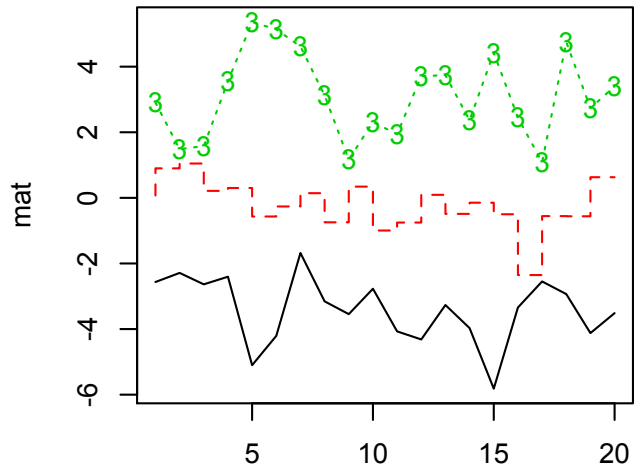
Example (matplot, matpoints, matlines 2)

```
> par(mfrow=c(2,2))
> matplot(mat, type="lSo", main="type=\"lSo\"")
> matplot(mat, type=c("l","S","o"), main="type=c(\"l\",\"S\",\"o\")")
> matplot(mat, col=c("red","blue","green"), cex=c(1,1.2,1.4))
> title(main="c(\"red\",\"blue\",\"green\"), cex=c(1,1.2,1.4)")
> matplot(mat, type="l", lty=3:5, lwd=1:3)
> title(main="lty=3:5, lwd=1:3")
```

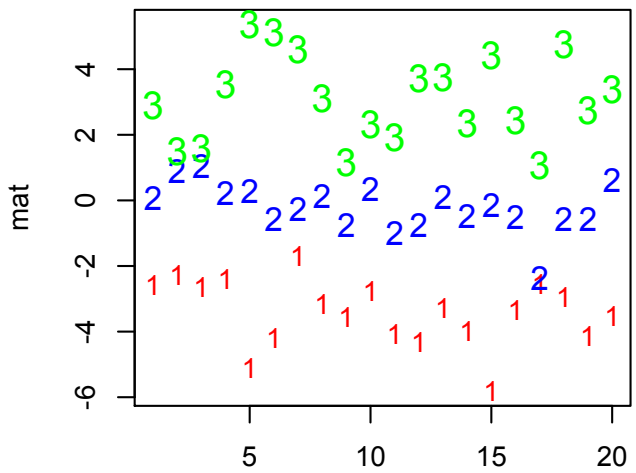
type="lSo"



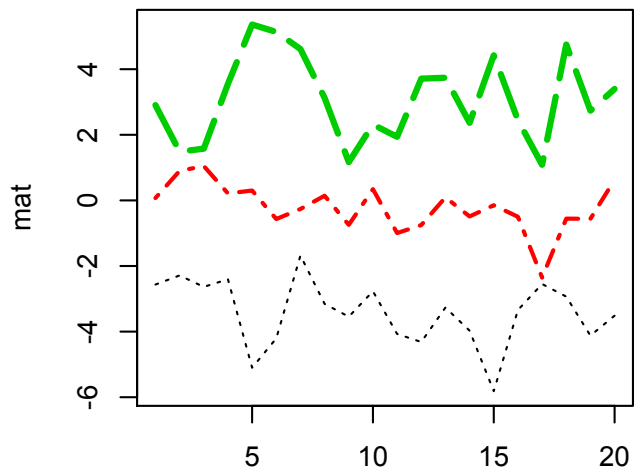
type=c("l","S","o")



c("red","blue","green"), cex=c(1,1.2,1.4)

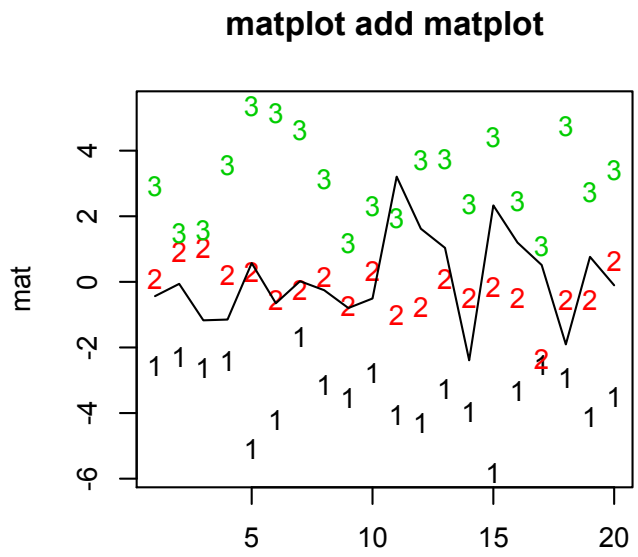


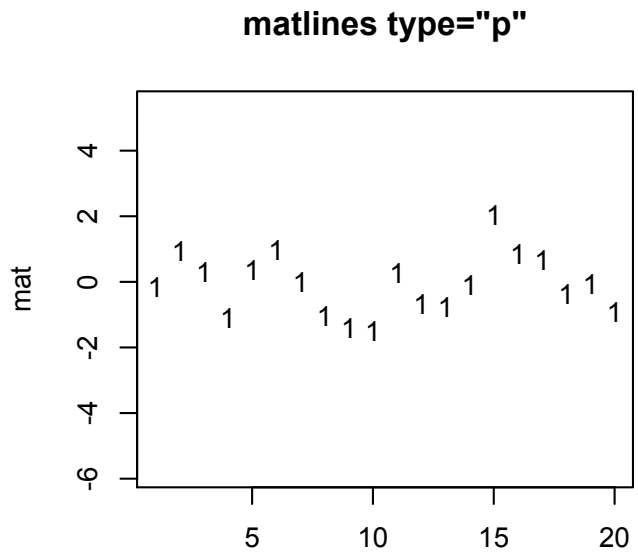
lty=3:5, lwd=1:3

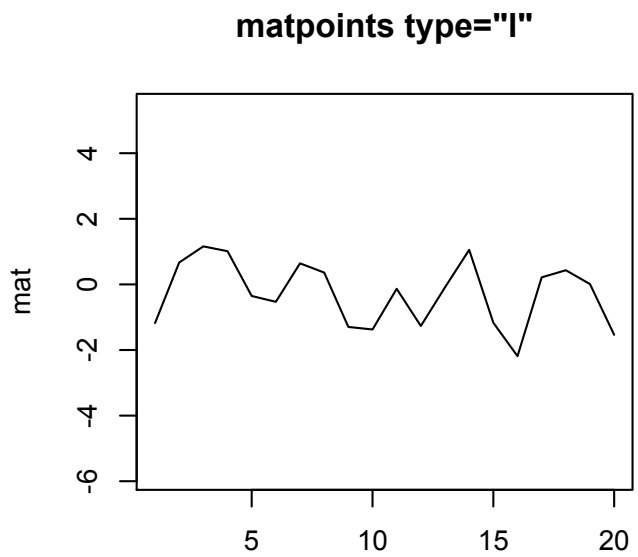


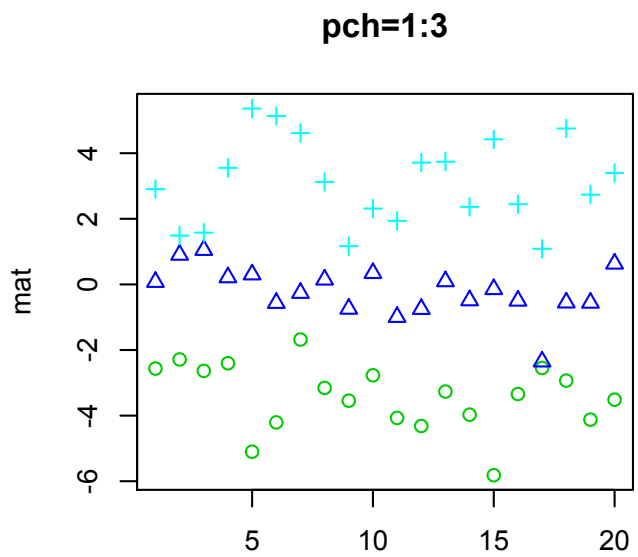
Example (matplot, matpoints, matlines 3)

```
> par(mfrow=c(2,2))
> matplot(mat)
> matplot(rnorm(20), type="l", add=TRUE)
> title(main="matplot add matplot")
> matplot(mat, type="n")
> matlines(rnorm(20), type="p")
> title(main="matlines type=\"p\"")
> matplot(mat, type="n")
> matpoints(rnorm(20), type="l")
> title(main="matpoints type=\"l\"")
> matplot(mat, pch=1:3, col=3:5, verbose=TRUE)
matplot: doing 3 plots with  col= ("3" "4" "5") pch= ("1" "2" "3") ...

> title(main="pch=1:3")
```


matplot add matplot
mat
5
10
15
20
-6
-4
-2
0
2
4


matlines type="p"
mat
5
10
15
20
-6
-4
-2
0
2
4


matpoints type="l"
mat
5
10
15
20
-6
-4
-2
0
2
4

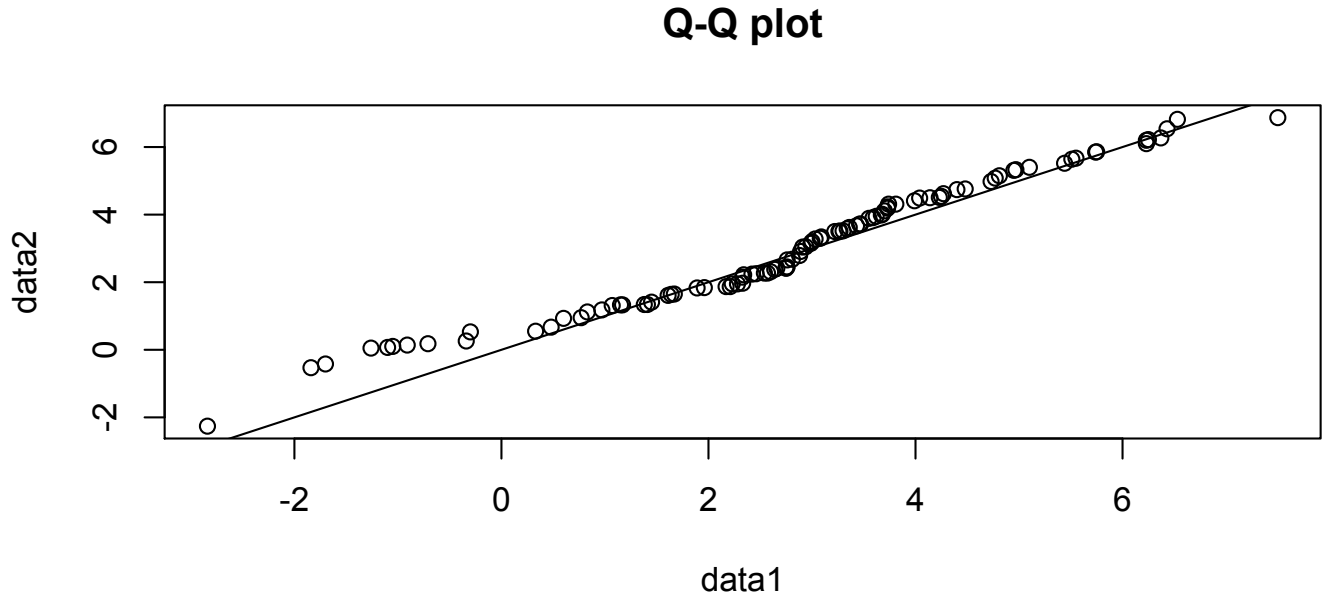

pch=1:3
mat
5
10
15
20
-6
-4
-2
0
2
4

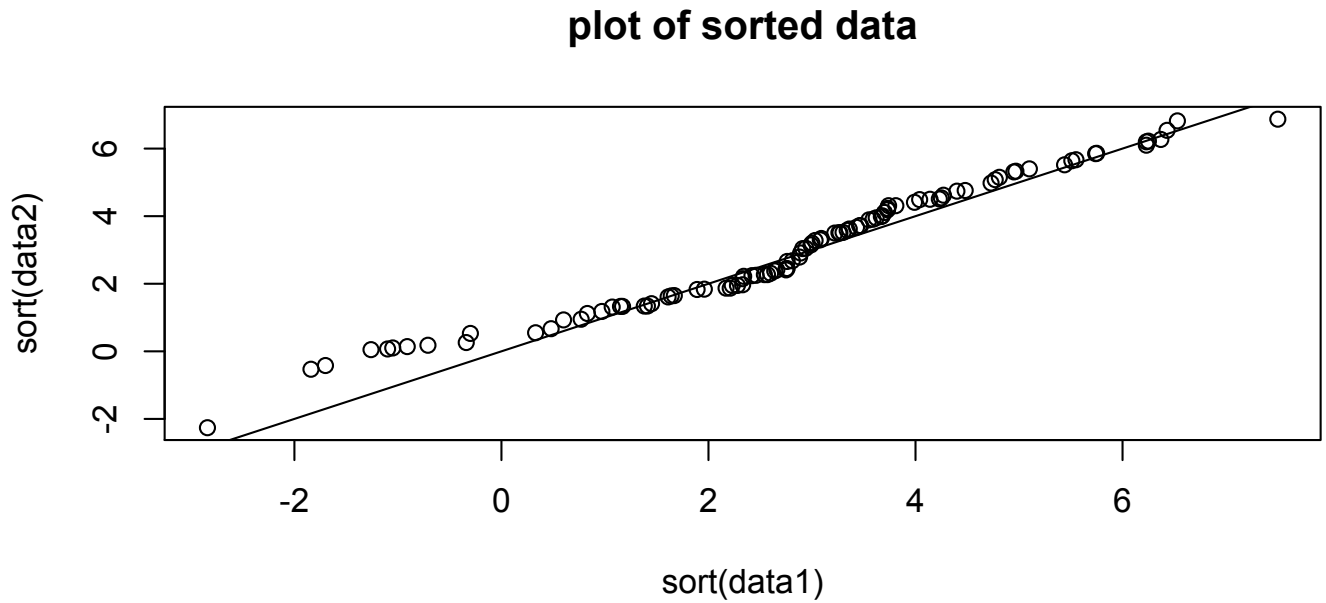
Definition (qqnorm, qqline, qqplot)

```
# qqnorm is a generic function the default method of which produces a normal
# QQ plot of the values in y. qqline adds a line to a normal quantile-quantile plot
# which passes through the first and third quartiles.
# qqplot produces a QQ plot of two datasets.
# Graphical parameters may be given as arguments to qqnorm, qqplot and qqline.

## Default S3 method:
qqnorm(y, ylim, main = "Normal Q-Q Plot",
       xlab = "Theoretical Quantiles", ylab = "Sample Quantiles",
       plot.it = TRUE, datax = FALSE, ...)

qqline(y, datax = FALSE, ...)

qqplot(x, y, plot.it = TRUE, xlab = deparse(substitute(x)),
       ylab = deparse(substitute(y)), ...)
```

Example (qqnorm, qqline, qqplot 1)

```
> data1 <- round(rnorm(100,3,2), 2)
> data2 <- round(rnorm(100,3,2), 2)
> par(mfrow=c(2,1))
> qqplot(data1, data2, main='Q-Q plot')
> abline(0,1)
> plot(sort(data1), sort(data2), main='plot of sorted data')
> abline(0,1)
```


Q-Q plot
data2
-2
0
2
4
6
-2
0
2
4
6
data1

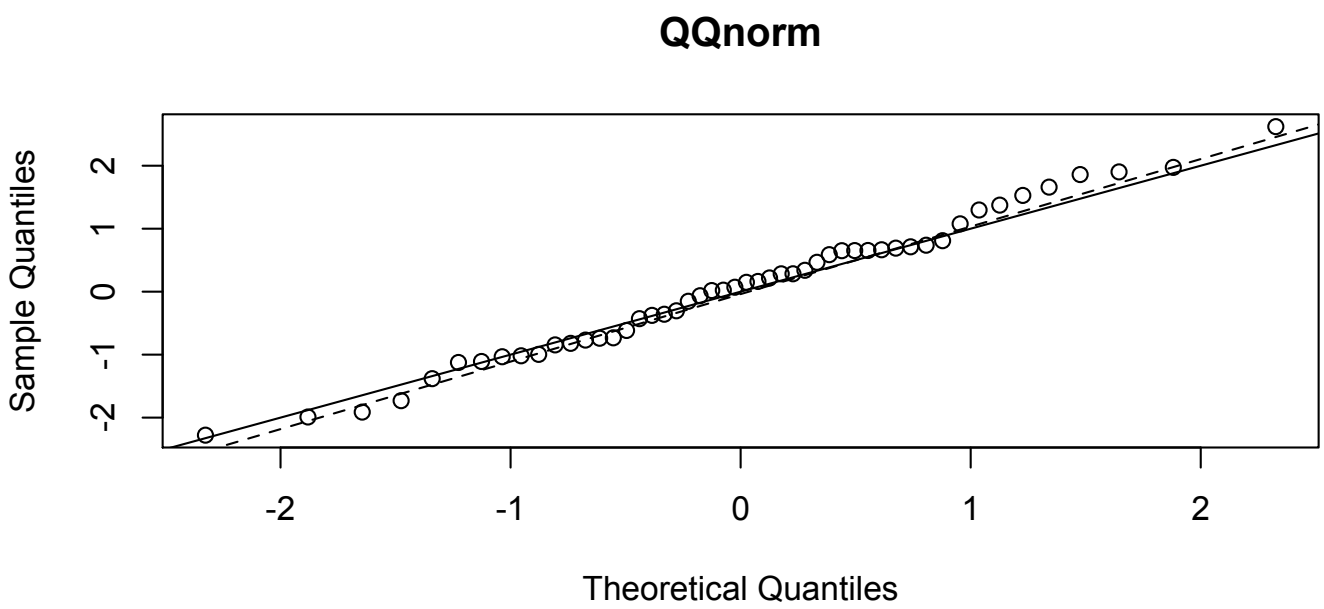

plot of sorted data
sort(data2)
-2
0
2
4
6
-2
0
2
4
6
sort(data1)

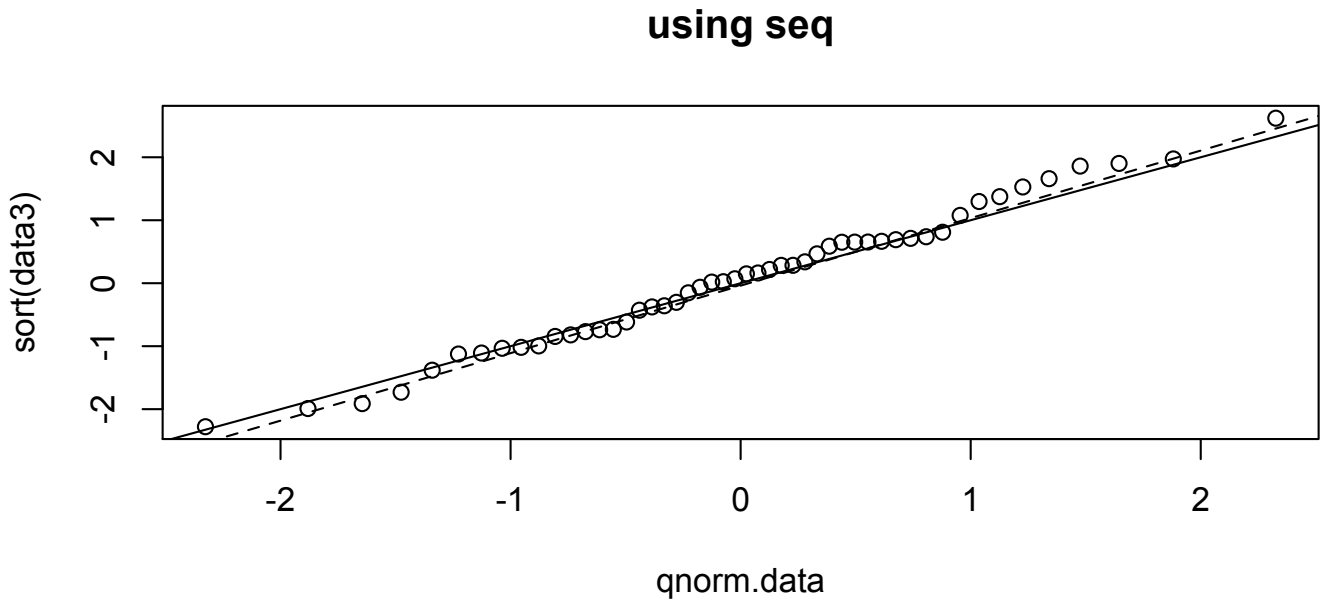
Example (qqnorm, qqline, qqplot 2)

```
> x <- round(rnorm(10), 2)
> y <- round(rnorm(10), 2)
> x
 [1] -0.17 -0.47  1.63 -0.42 -0.37 -2.31 -1.24 -1.23  0.53 -0.27
> y
 [1]  0.61 -1.40  1.70 -0.50  0.06  0.49 -1.27 -0.64  0.72  1.11
> qqplot(x, y, plot.it=FALSE)
> .Last.value
$x
 [1] -2.31 -1.24 -1.23 -0.47 -0.42 -0.37 -0.27 -0.17  0.53  1.63

$y
 [1] -1.40 -1.27 -0.64 -0.50  0.06  0.49  0.61  0.72  1.11  1.70

> qq<-qqplot(x, y, plot.it=FALSE)
> qq
$x
 [1] -2.31 -1.24 -1.23 -0.47 -0.42 -0.37 -0.27 -0.17  0.53  1.63

$y
 [1] -1.40 -1.27 -0.64 -0.50  0.06  0.49  0.61  0.72  1.11  1.70
```

Example (qqnorm, qqline, qqplot 3)

```
> data3 <- rnorm(50)
> seq.norm <- seq(1,99,2)/100
> qnorm.data <- qnorm(seq.norm)
> par(mfrow=c(2,1))
> qqnorm(data3, main="QQnorm")
> used.qqnorm <- .Last.value
> abline(0, 1)
> qqline(data3, lty=2)
> plot(qnorm.data, sort(data3), main="using seq")
> abline(0, 1)
> qqline(data3, lty=2)
> sort(data3)
 [1] -2.27859133 -1.99003400 -1.91212682 -1.73106850 -1.38142112 -1.12439939
...
[49]  1.97348842  2.62054769
> sort(used.qqnorm$y)
 [1] -2.27859133 -1.99003400 -1.91212682 -1.73106850 -1.38142112 -1.12439939
...
[49]  1.97348842  2.62054769
> qnorm.data
 [1] -2.32634787 -1.88079361 -1.64485363 -1.47579103 -1.34075503 -1.22652812
...
[49]  1.88079361  2.32634787
> sort(used.qqnorm$x)
 [1] -2.32634787 -1.88079361 -1.64485363 -1.47579103 -1.34075503 -1.22652812
...
[49]  1.88079361  2.32634787
```


QQnorm
Sample Quantiles
-2
-1
0
1
2
Theoretical Quantiles
-2
-1
0
1
2

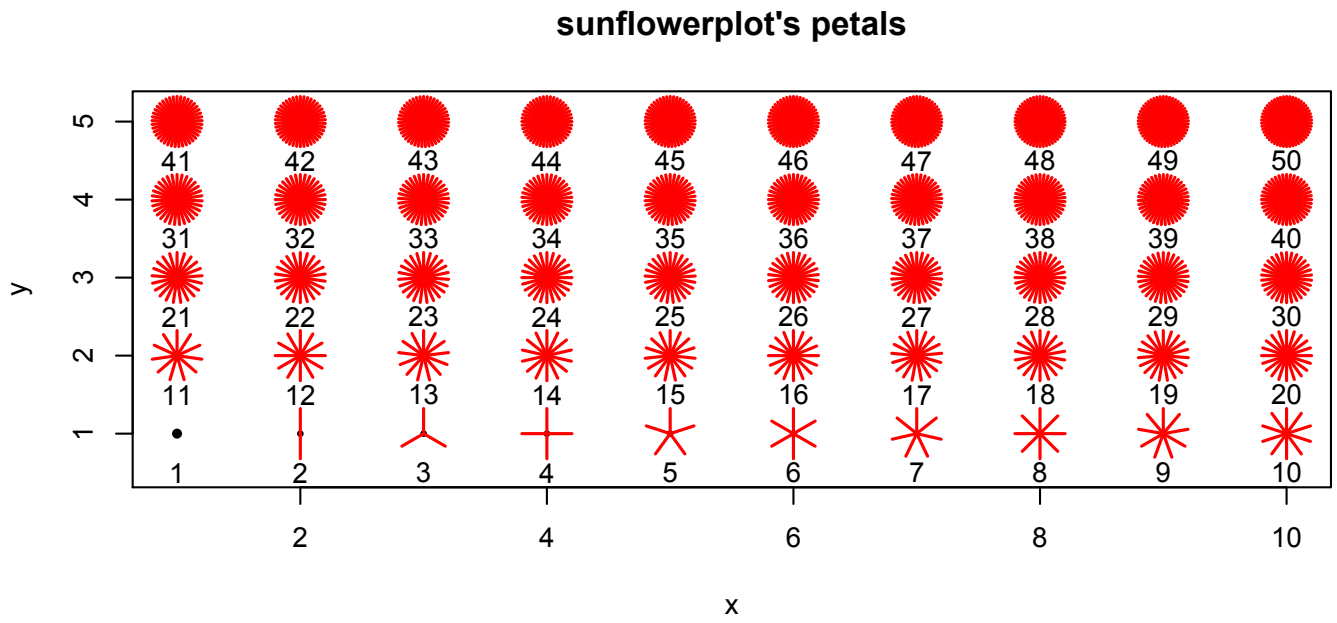

using seq
sort(data3)
-2
-1
0
1
2
qnorm.data
-2
-1
0
1
2

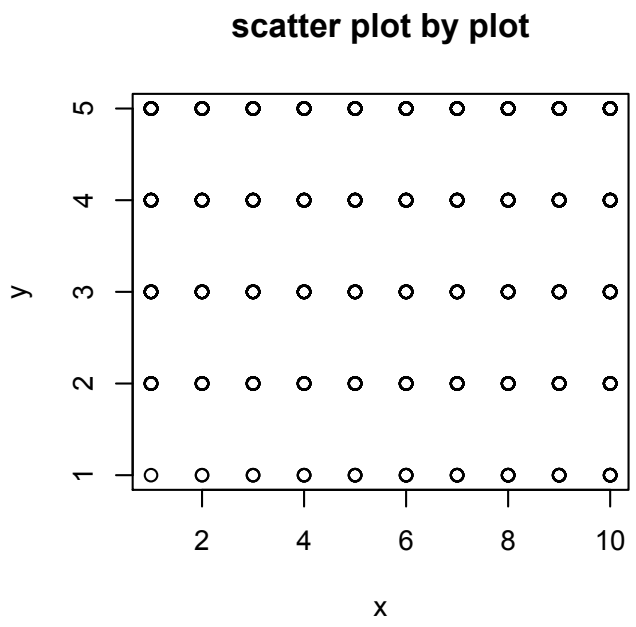
Definition (sunflowerplot)

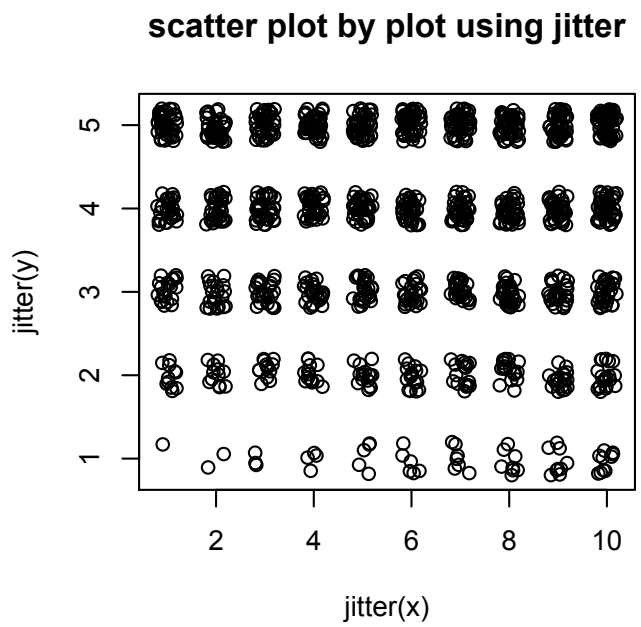
```
# Multiple points are plotted as ‘sunflowers’ with multiple leaves (‘petals’)
# such that overplotting is visualized instead of accidental and invisible.

## Default S3 method:
sunflowerplot(x, y = NULL, number, log = "", digits = 6,
              xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL,
              add = FALSE, rotate = FALSE,
              pch = 16, cex = 0.8, cex.fact = 1.5,
              col = par("col"), bg = NA, size = 1/8, seg.col = 2,
              seg.lwd = 1.5, ...)

## S3 method for class 'formula'
sunflowerplot(formula, data = NULL, xlab = NULL, ylab = NULL, ...,
             subset, na.action = NULL)
```

Example (sunflowerplot 1)

```
> x<-NULL
> y<-NULL
> for(i in 1:50){
+  x <- c(x, rep( ifelse( i%%10==0, 10, i%%10 ), i) )
+  y <- c(y, rep( (i-1)%/%10 + 1, i))
+ }
> t(table(x,y))
   x
y    1  2  3  4  5  6  7  8  9 10
  1  1  2  3  4  5  6  7  8  9 10
  2 11 12 13 14 15 16 17 18 19 20
  3 21 22 23 24 25 26 27 28 29 30
  4 31 32 33 34 35 36 37 38 39 40
  5 41 42 43 44 45 46 47 48 49 50
> layout(matrix(c(1,1,2,3), ncol=2, byrow=T))
> sunflowerplot(x, y, ylim=c(0.5, 5.2))
> title(main="sunflowerplot's petals")
> text(x=ifelse(1:50 %% 10 ==0, 10, 1:50%%10), y=((1:50 -1)%/%10+1) -0.5,
+ as.character(1:50))
> plot(x,y)
> title(main="scatter plot by plot")
> plot(jitter(x), jitter(y))
> title(main="scatter plot by plot using jitter")
```

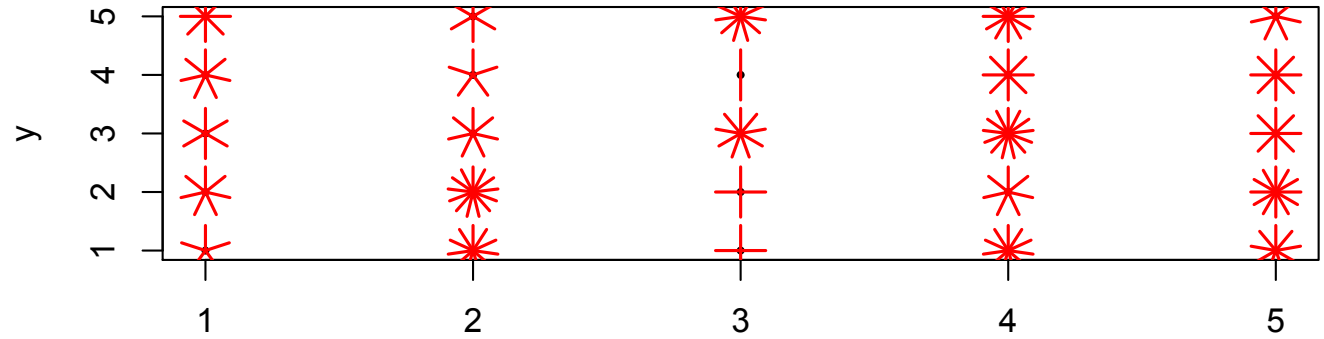

sunflowerplot's petals
41 42 43 44 45 46 47 48 49 50
31 32 33 34 35 36 37 38 39 40
21 22 23 24 25 26 27 28 29 30
11 12 13 14 15 16 17 18 19 20
1 2 3 4 5 6 7 8 9 10
y
x

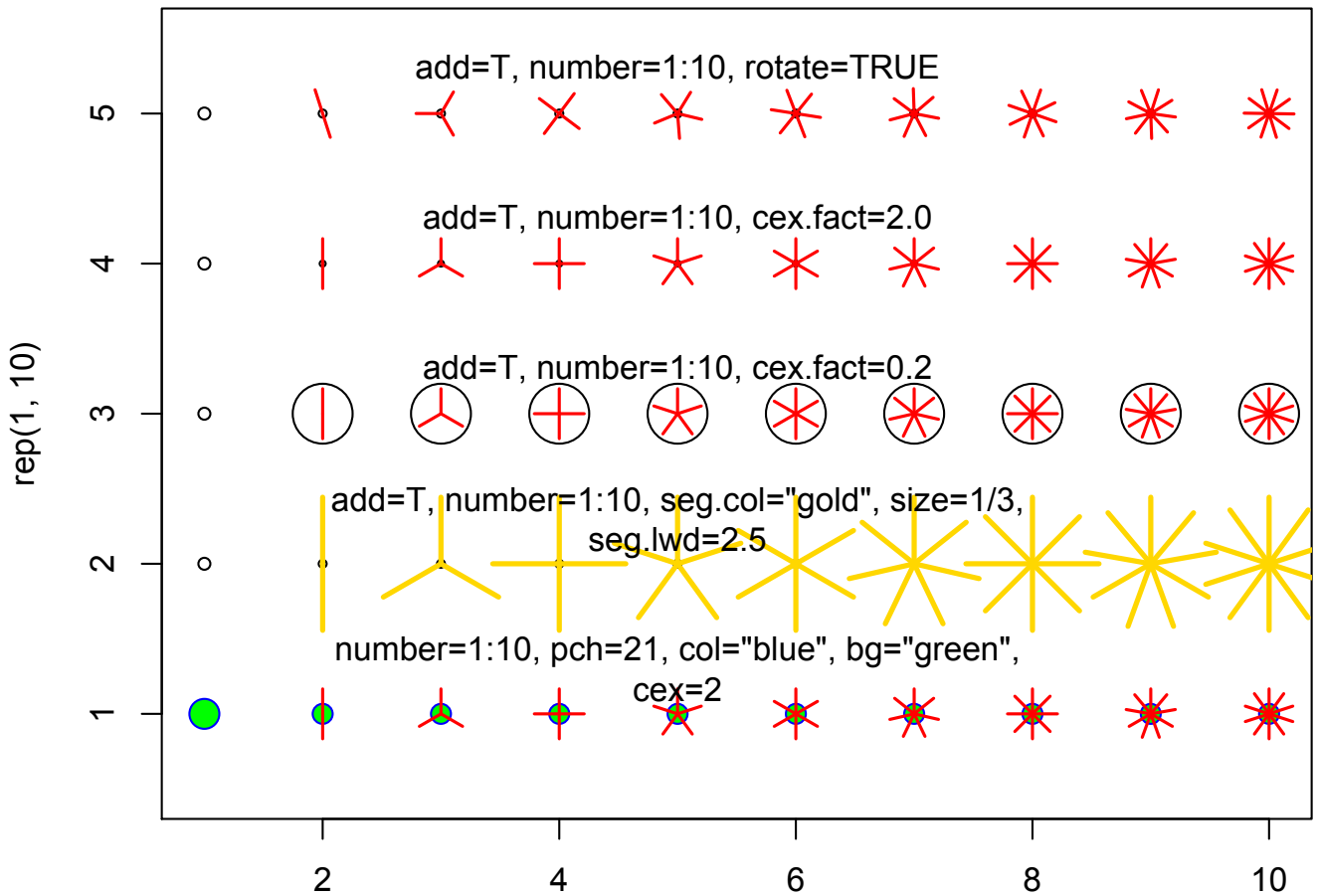

scatter plot by plot
y
x


scatter plot by plot using jitter
jitter(y)
jitter(x)

Example (sunflowerplot 2)

```
> x<-sample(1:5, 200, replace=T)
> y<-sample(1:5, 200, replace=T)
> par(mfrow=c(2,1),mar=c(2.1,4.1,2.1,2.1),fig=c(0,1,0.7,1))
> sunflowerplot(x,y)
> par(fig=c(0,1,0,0.7), new=T)
> sunflowerplot(1:10, rep(1,10), ylim=c(0.5,5.5), number=1:10,
+ pch=21, col="blue", bg="green", cex=2)
> text(5, 1.3, "number=1:10, pch=21, col=\"blue\", bg=\"green\",
+ cex=2", adj=0.5)
> sunflowerplot(1:10, rep(2,10), add=T, number=1:10,
+ seg.col="gold", size=1/3, seg.lwd=2.5, pch=21)
> text(5, 2.3, "add=T, number=1:10, seg.col=\"gold\", size=1/3,
+ seg.lwd=2.5", adj=0.5)
> sunflowerplot(1:10, rep(3,10), add=T, number=1:10, cex.fact=0.2, pch=21)
> text(5, 3.3, "add=T, number=1:10, cex.fact=0.2", adj=0.5)
> sunflowerplot(1:10, rep(4,10), add=T, number=1:10, cex.fact=2.0, pch=21)
> text(5, 4.3, "add=T, number=1:10, cex.fact=2.0", adj=0.5)
> sunflowerplot(1:10, rep(5,10), add=T, number=1:10, rotate=TRUE, pch=21)
> text(5, 5.3, "add=T, number=1:10, rotate=TRUE", adj=0.5)
```


y
1
2
3
4
5

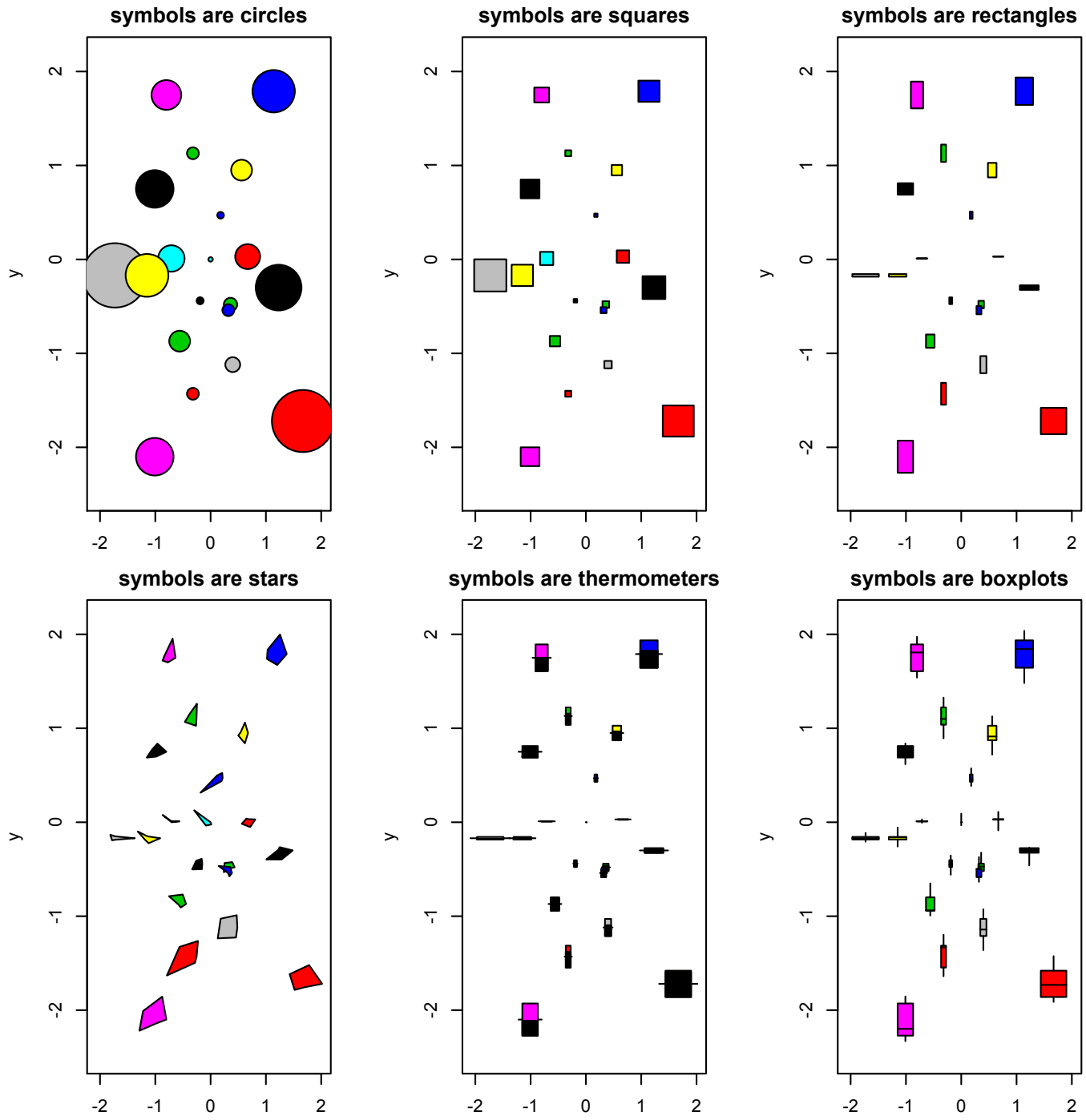

rep(1, 10)
add=T, number=1:10, rotate=TRUE
add=T, number=1:10, cex.fact=2.0
add=T, number=1:10, cex.fact=0.2
add=T, number=1:10, seg.col="gold", size=1/3,
seg.lwd=2.5
number=1:10, pch=21, col="blue", bg="green",
cex=2
2
4
6
8
10

Definition (symbols)

```
# This function draws symbols on a plot. One of six symbols; circles, squares,
# rectangles, stars, thermometers, and boxplots, can be plotted at a specified
# set of x and y coordinates. Specific aspects of the symbols, such as relative
# size, can be customized by additional parameters.

symbols(x, y = NULL, circles, squares, rectangles, stars,
        thermometers, boxplots, inches = TRUE, add = FALSE,
        fg = par("col"), bg = NA,
        xlab = NULL, ylab = NULL, main = NULL,
        xlim = NULL, ylim = NULL, ...)
```

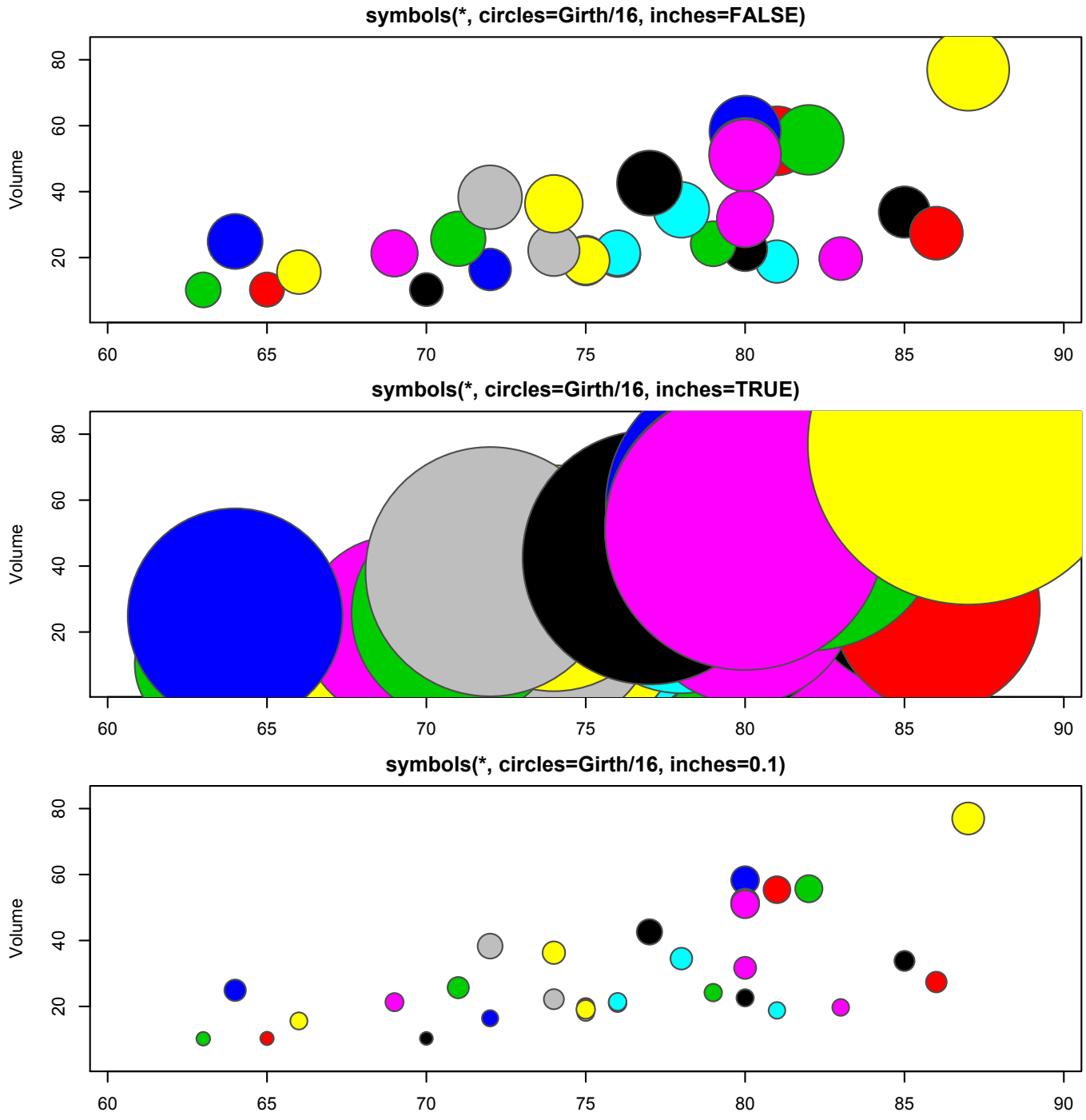
Example (symbols 1)

```
> x <- round(rnorm(20),2)
> y <- round(rnorm(20),2)
> z1 <- abs(round(rnorm(20),2))
> z2 <- abs(round(rnorm(20),2))
> z3 <- round(runif(20),2)
> z4 <- round(runif(20),2)
> z5 <- round(runif(20),2)
> par(mfrow=c(2,3))
> symbols(x, y, circles=abs(x), inches=0.2, bg=1:20)
> title("symbols are circles")
> symbols(x, y, squares=abs(x), inches=0.2, bg=1:20)
> title("symbols are squares")
> symbols(x, y, rectangles=cbind(abs(x),abs(y)), inches=0.2, bg=1:20)
> title("symbols are rectangles")
> symbols(x, y, stars=cbind(abs(x),abs(y),z1,z2,z3), inches=0.2, bg=1:20)
> title("symbols are stars")
> symbols(x, y, thermometers=cbind(abs(x),abs(y),z4), inches=0.2, bg=1:20)
> title("symbols are thermometers")
> symbols(x, y, boxplots=cbind(abs(x),abs(y),z3,z4,z5), inches=0.2, bg=1:20)
> title("symbols are boxplots")
```


symbols are circles
symbols are squares
symbols are rectangles
symbols are stars
symbols are thermometers
symbols are boxplots
y
-2
-1
0
1
2

Example (symbols 2)

```
> head(trees)
  Girth Height Volume
1   8.3     70   10.3
2   8.6     65   10.3
3   8.8     63   10.2
4  10.5     72   16.4
5  10.7     81   18.8
6  10.8     83   19.7
> N <- nrow(trees)
> palette(rainbow(N, end=0.9))
> par(mfrow=c(3,1))
> with(trees, {
+  symbols(Height, Volume, circles=Girth/16, inches=FALSE, bg=1:N,
+  fg="gray30", main="symbols(*, circles=Girth/16, inches=FALSE)")
+  symbols(Height, Volume, circles=Girth/16, inches=TRUE, bg=1:N,
+  fg="gray30", main="symbols(*, circles=Girth/16, inches=TRUE)")
+  symbols(Height, Volume, circles=Girth/16, inches=0.1, bg=1:N,
+  fg="gray30", main="symbols(*, circles=Girth/16, inches=0.1)")
+ })
> palette("default")
```

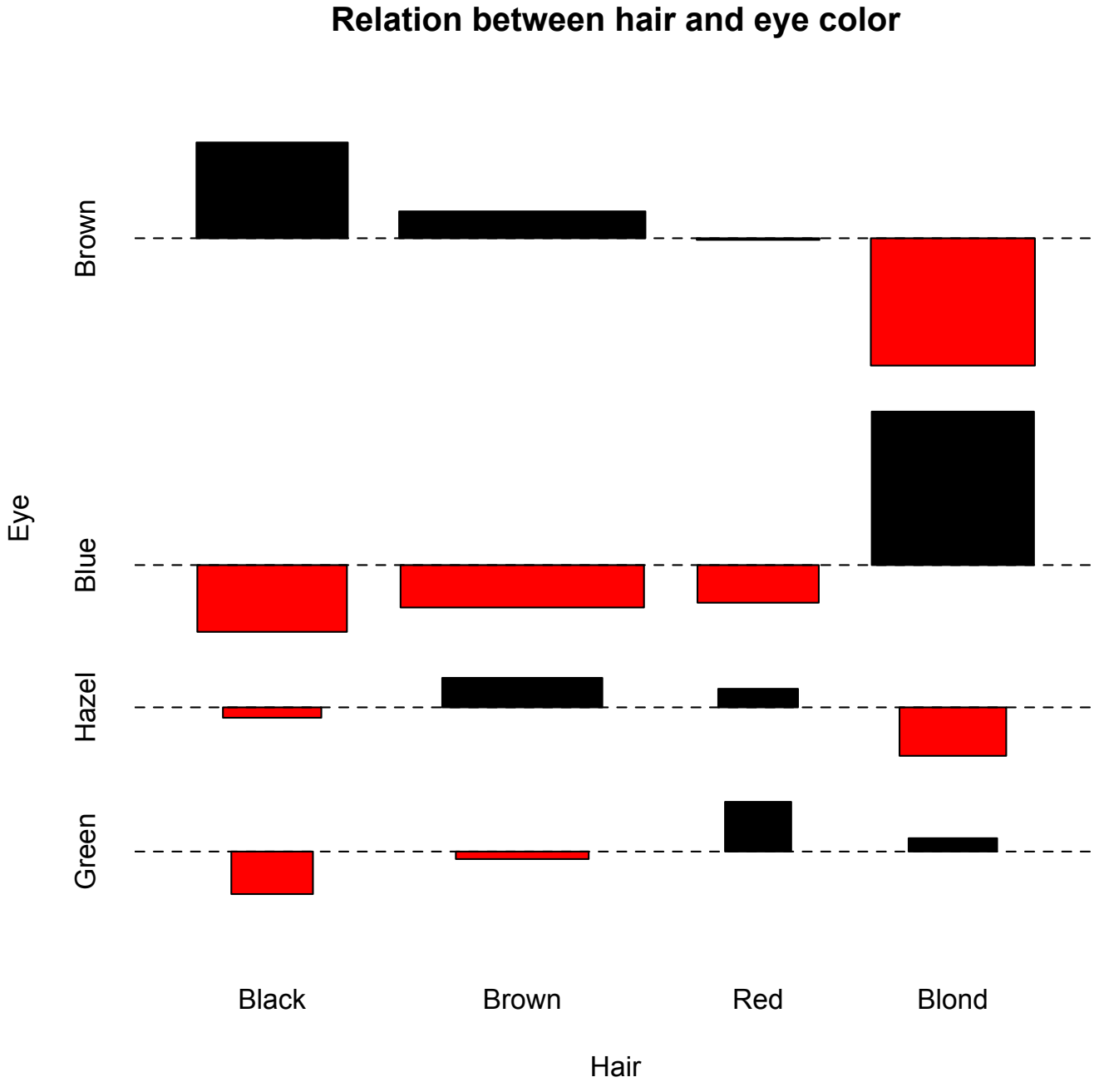

symbols(*, circles=Girth/16, inches=FALSE)
Volume
20
40
60
80
60
65
70
75
80
85
90
symbols(*, circles=Girth/16, inches=TRUE)
Volume
20
40
60
80
60
65
70
75
80
85
90
symbols(*, circles=Girth/16, inches=0.1)
Volume
20
40
60
80
60
65
70
75
80
85
90

Definition (assocplot)

```
# Produce a Cohen-Friendly association plot indicating deviations from
# independence of rows and columns in a 2-dimensional contingency table.

assocplot(x, col = c("black", "red"), space = 0.3,
          main = NULL, xlab = NULL, ylab = NULL)
```

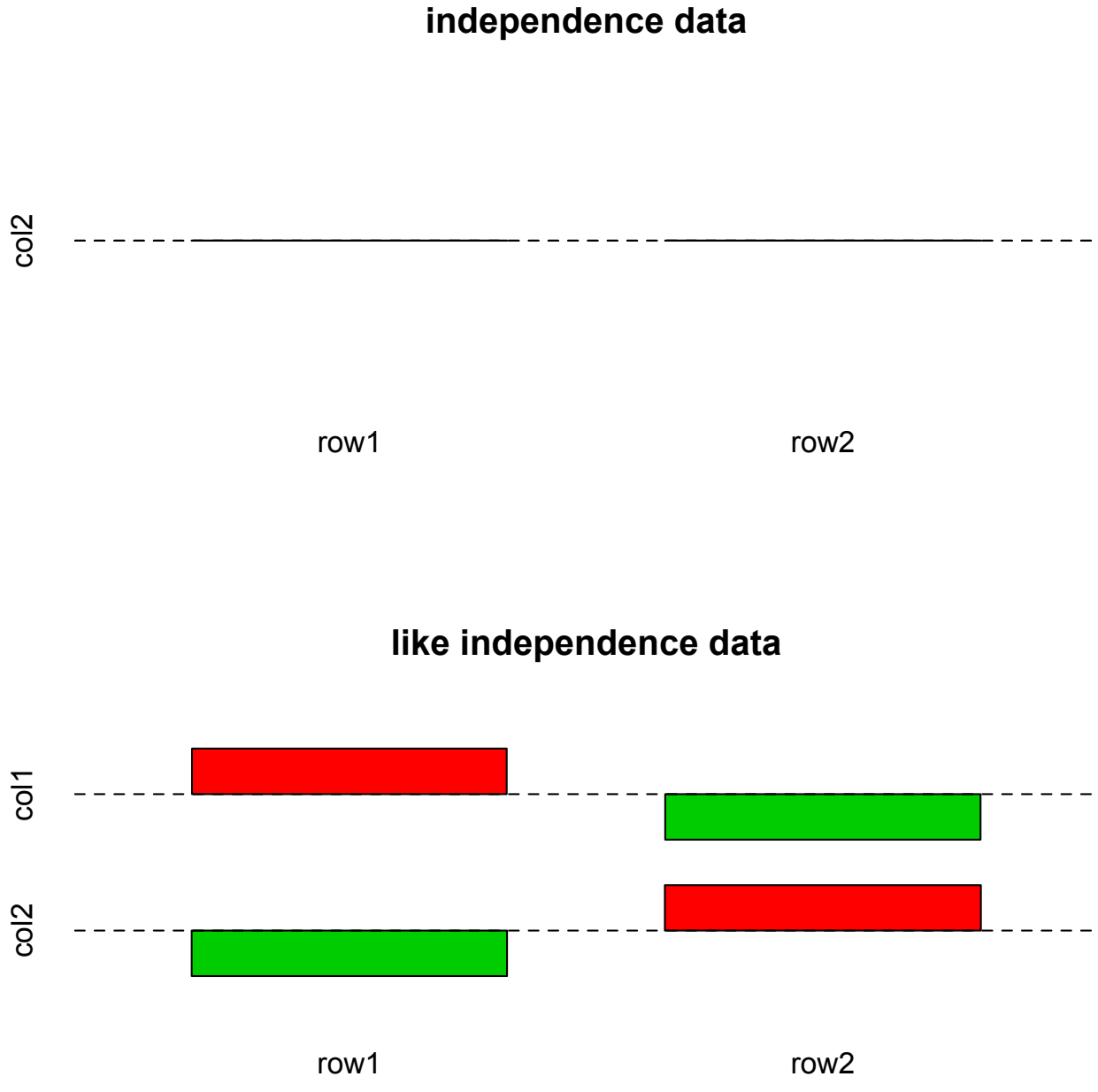
Example (assocplot 1)

```
> x <- margin.table(HairEyeColor, c(1,2))
> x
       Eye
Hair    Brown Blue Hazel Green
  Black    68   20    15     5
  Brown   119   84    54    29
  Red      26   17    14    14
  Blond     7   94    10    16
> assocplot(x, main="Relation between hair and eye color")
> chisq.test(x)

Pearson's Chi-squared test

data:  x
X-squared = 138.2898, df = 9, p-value < 2.2e-16
```


Relation between hair and eye color
Eye
Brown
Blue
Hazel
Green
Black
Brown
Red
Blond
Hair

Example (assocplot 2)

```
> x <- matrix(rep(120,4),ncol=2,
+ dimnames=list(c("row1","row2"),c("col1","col2")))
> x
     col1 col2
row1  120  120
row2  120  120
> y <- matrix(c(120,120,120,121),ncol=2,
+ dimnames=list(c("row1","row2"),c("col1","col2")))
> y
     col1 col2
row1  120  120
row2  120  121
> par(mfrow=c(2,1))
> assocplot(x, col=2:3, space=0.5)
> title(main="independence data")
> assocplot(y, col=2:3, space=0.5)
> title(main="like independence data")
> chisq.test(x)

Pearson's Chi-squared test
data:  x
X-squared = 0, df = 1, p-value = 1

> chisq.test(y)

Pearson's Chi-squared test with Yates' continuity correction
data:  y
X-squared = 0, df = 1, p-value = 1
```

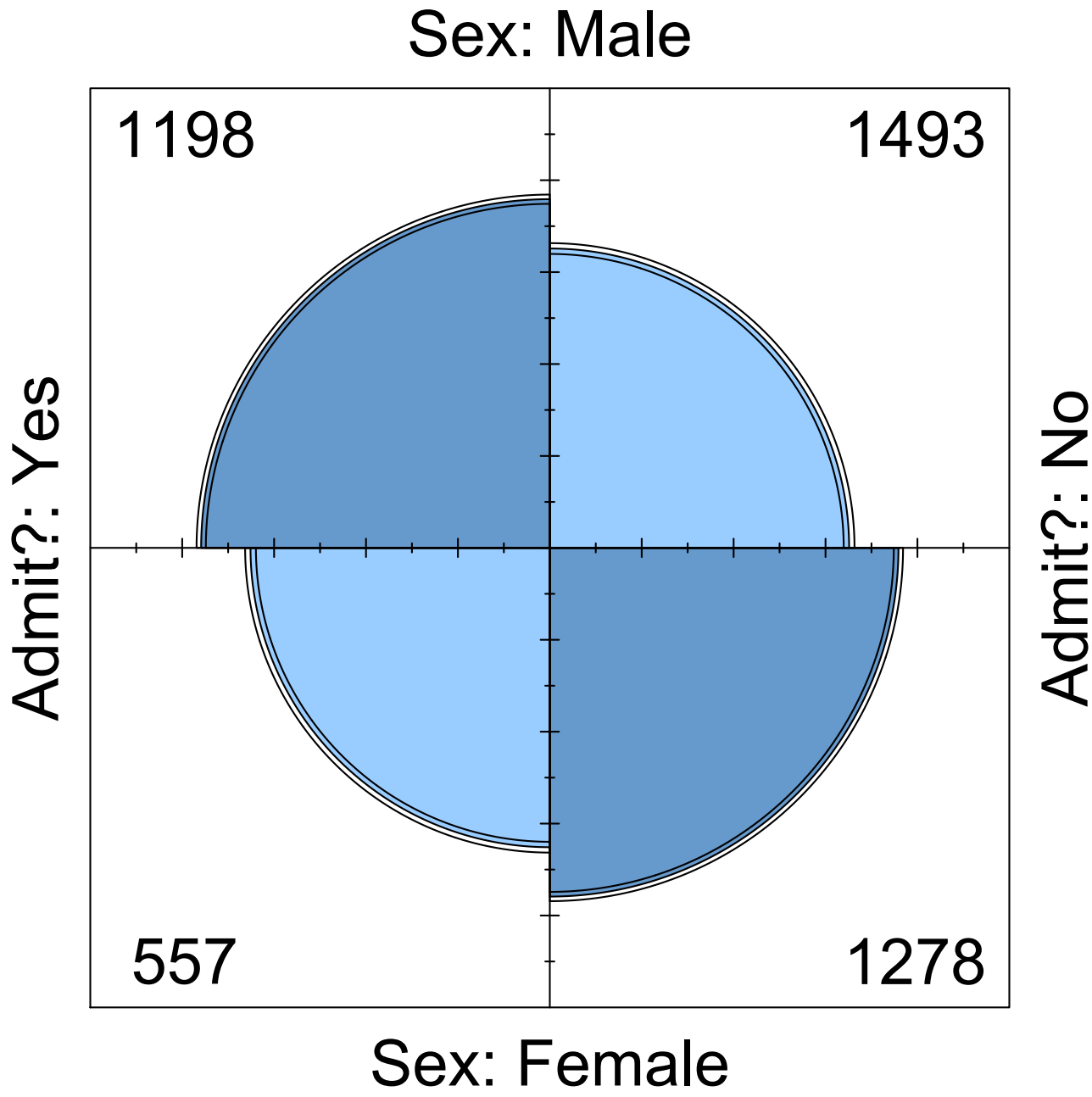

independence data
col2
row1
row2
like independence data
col1
col2
row1
row2

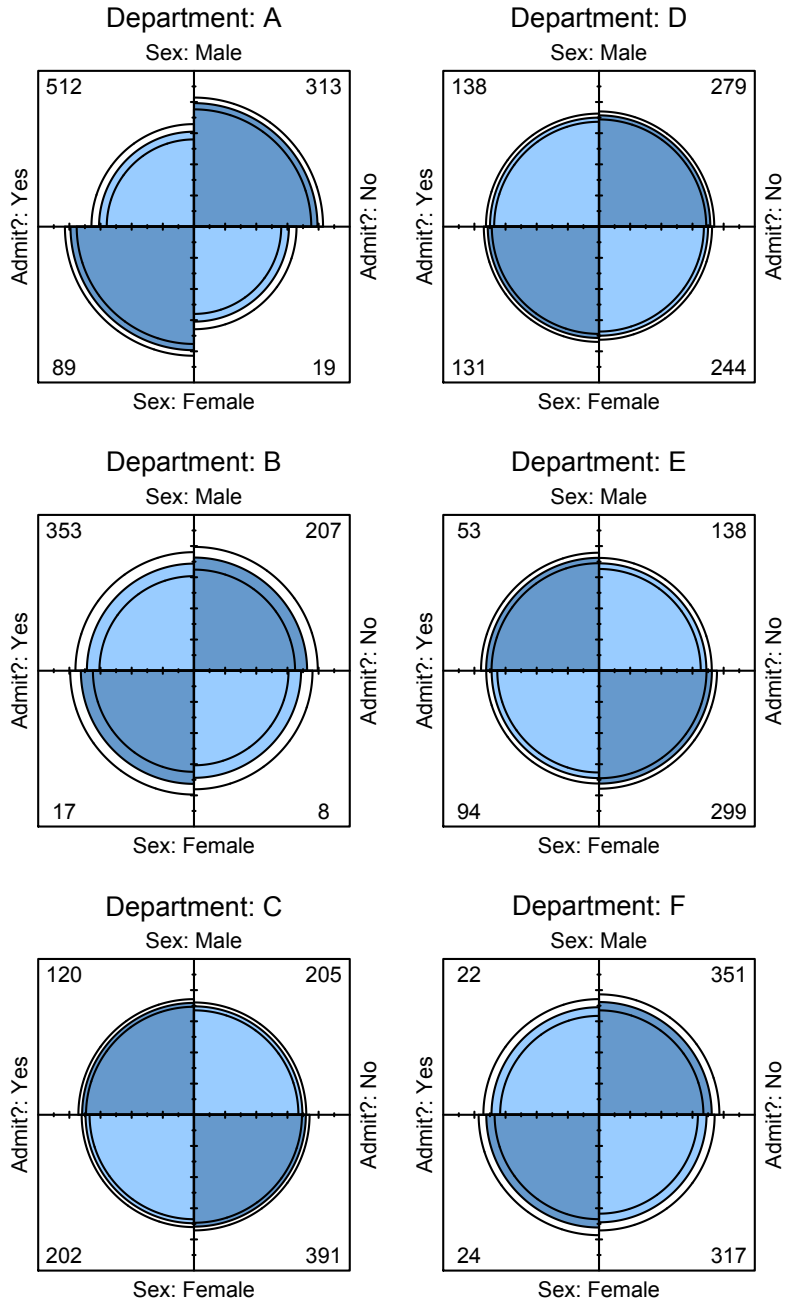
Definition (fourfoldplot)

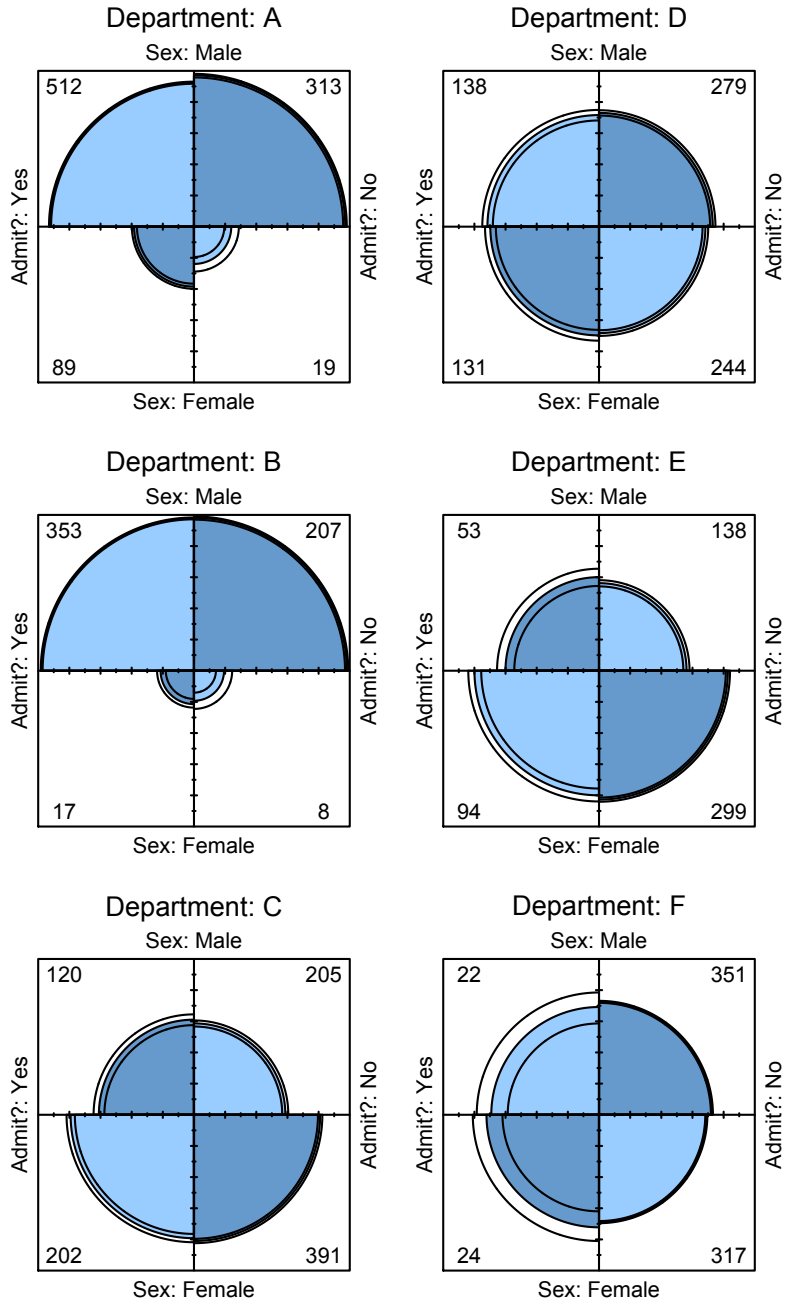
```
# Creates a fourfold display of a 2 by 2 by k contingency table on the current
# graphics device, allowing for the visual inspection of the association between
# two dichotomous variables in one or several populations (strata).

fourfoldplot(x, color = c("#99CCFF", "#6699CC"),
             conf.level = 0.95,
             std = c("margins", "ind.max", "all.max"),
             margin = c(1, 2), space = 0.2, main = NULL,
             mfrow = NULL, mfcol = NULL)
```

Example (fourfoldplot)

```
> admis <- aperm(UCBAdmissions, c(2,1,3))
> dimnames(admis)[[2]] <- c("Yes","No")
> names(dimnames(admis)) <- c("Sex", "Admit?", "Department")
> ftable(admis)
              Department   A   B   C   D   E   F
Sex    Admit?
Male   Yes               512 353 120 138  53  22
       No                313 207 205 279 138 351
Female Yes                89  17 202 131  94  24
       No                 19   8 391 244 299 317
> admis.sex <- margin.table(admis, c(1,2))
> admis.sex
        Admit?
Sex       Yes   No
  Male   1198 1493
  Female  557 1278
> prop.table(admis.sex,1)
        Admit?
Sex             Yes        No
  Male   0.4451877 0.5548123
  Female 0.3035422 0.6964578
> fourfoldplot(admis.sex)
> fourfoldplot(admis)
> fourfoldplot(admis, margin=2)
```


Sex: Male
1198
1493
Admit?: Yes
Admit?: No
557
1278
Sex: Female


Department: A
Sex: Male
512
313
89
19
Admit?: Yes
Admit?: No
Sex: Female
Department: D
Sex: Male
138
279
131
244
Admit?: Yes
Admit?: No
Sex: Female
Department: B
Sex: Male
353
207
17
8
Admit?: Yes
Admit?: No
Sex: Female
Department: E
Sex: Male
53
138
94
299
Admit?: Yes
Admit?: No
Sex: Female
Department: C
Sex: Male
120
205
202
391
Admit?: Yes
Admit?: No
Sex: Female
Department: F
Sex: Male
22
351
24
317
Admit?: Yes
Admit?: No
Sex: Female


Department: A
Sex: Male
512
313
89
19
Admit?: Yes
Admit?: No
Sex: Female
Department: D
Sex: Male
138
279
131
244
Admit?: Yes
Admit?: No
Sex: Female
Department: B
Sex: Male
353
207
17
8
Admit?: Yes
Admit?: No
Sex: Female
Department: E
Sex: Male
53
138
94
299
Admit?: Yes
Admit?: No
Sex: Female
Department: C
Sex: Male
120
205
202
391
Admit?: Yes
Admit?: No
Sex: Female
Department: F
Sex: Male
22
351
24
317
Admit?: Yes
Admit?: No
Sex: Female

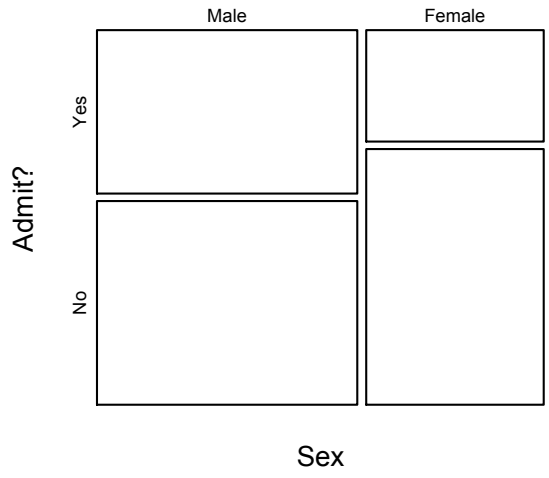
Definition (mosaicplot)

```
# Plots a mosaic on the current graphics device.

## Default S3 method:
mosaicplot(x, main = deparse(substitute(x)),
           sub = NULL, xlab = NULL, ylab = NULL,
           sort = NULL, off = NULL, dir = NULL,
           color = NULL, shade = FALSE, margin = NULL,
           cex.axis = 0.66, las = par("las"), border = NULL,
           type = c("pearson", "deviance", "FT"), ...)

## S3 method for class 'formula'
mosaicplot(formula, data = NULL, ...,
           main = deparse(substitute(data)), subset,
           na.action = stats::na.omit)
```

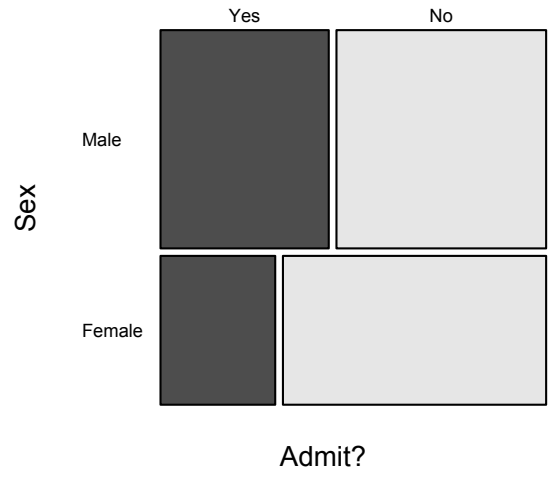
Example (mosaicplot 1)

```
> par(mfrow=c(2,2))
> mosaicplot(admis.sex, color=FALSE, las=0, main="color=FALSE, las=0")
> mosaicplot(admis.sex, color=TRUE, las=1, dir=c("h","v"),
+  xlab="Admit?", ylab="Sex",
+  main="color=T, las=1, dir=c(\"h\",\"v\"), xlab, ylab")
> mosaicplot(~Gender+Admit, data=UCBAdmissions, sort=c(2,1),
+  color=2:3, las=2, main="formula, sort=c(2,1), color=2:3, las=2")
> mosaicplot(admis.sex, off=c(5,20), las=3, shade=TRUE,
+  main="off=c(5,20), las=3, shade=TRUE")
```

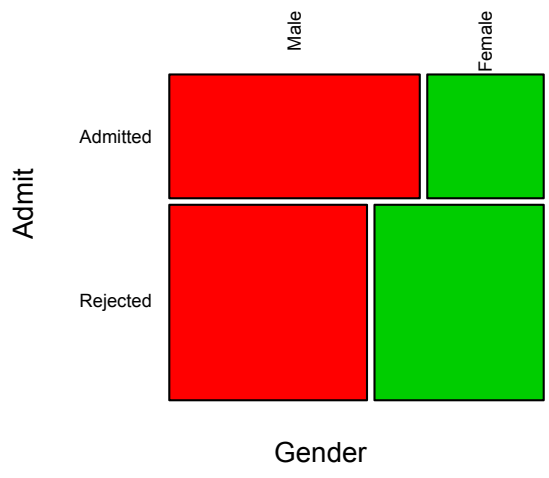
color=FALSE, las=0



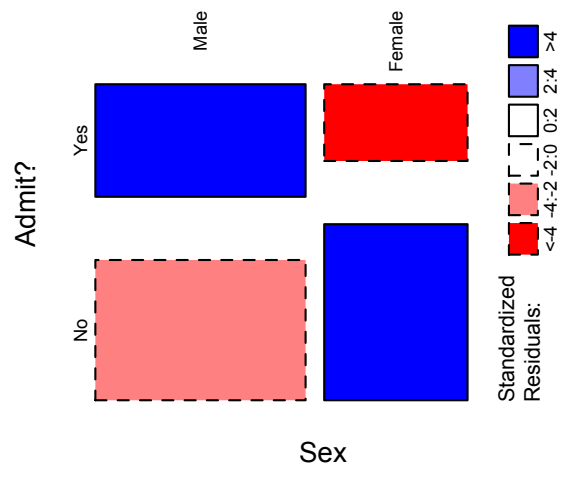
color=T, las=1, dir=c("h","v"), xlab, ylab



formula, sort=c(2,1), color=2:3, las=2



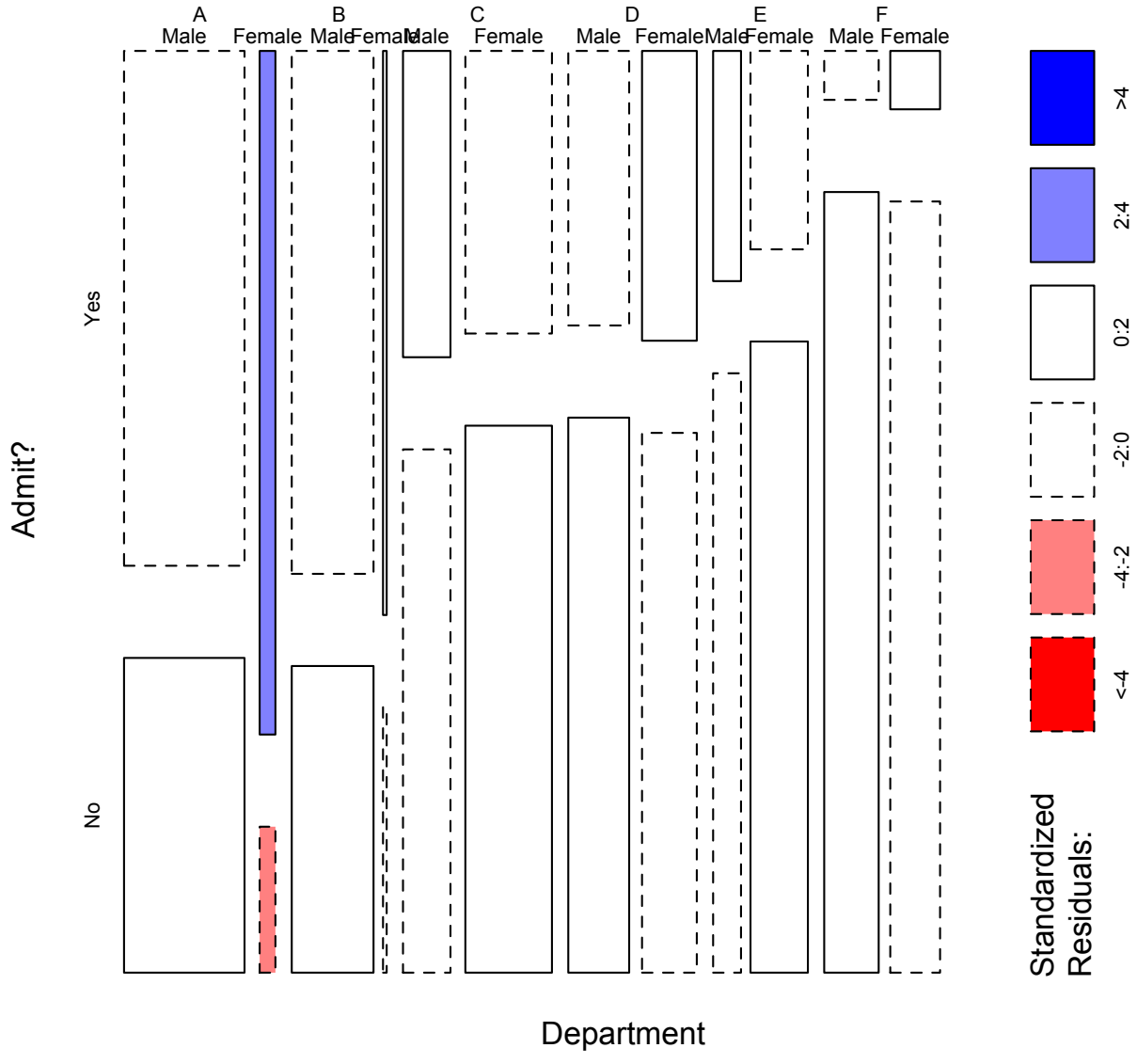
off=c(5,20), las=3, shade=TRUE



Example (mosaicplot 2)

```
> mosaicplot(admis, sort=c(3,1,2), shade=T, margin=list(c(1,3),c(2,3)),
+  xlab="Department", main="Sex, Admit?, Department Mosaic Plots")
```

Sex, Admit?, Department Mosaic Plots



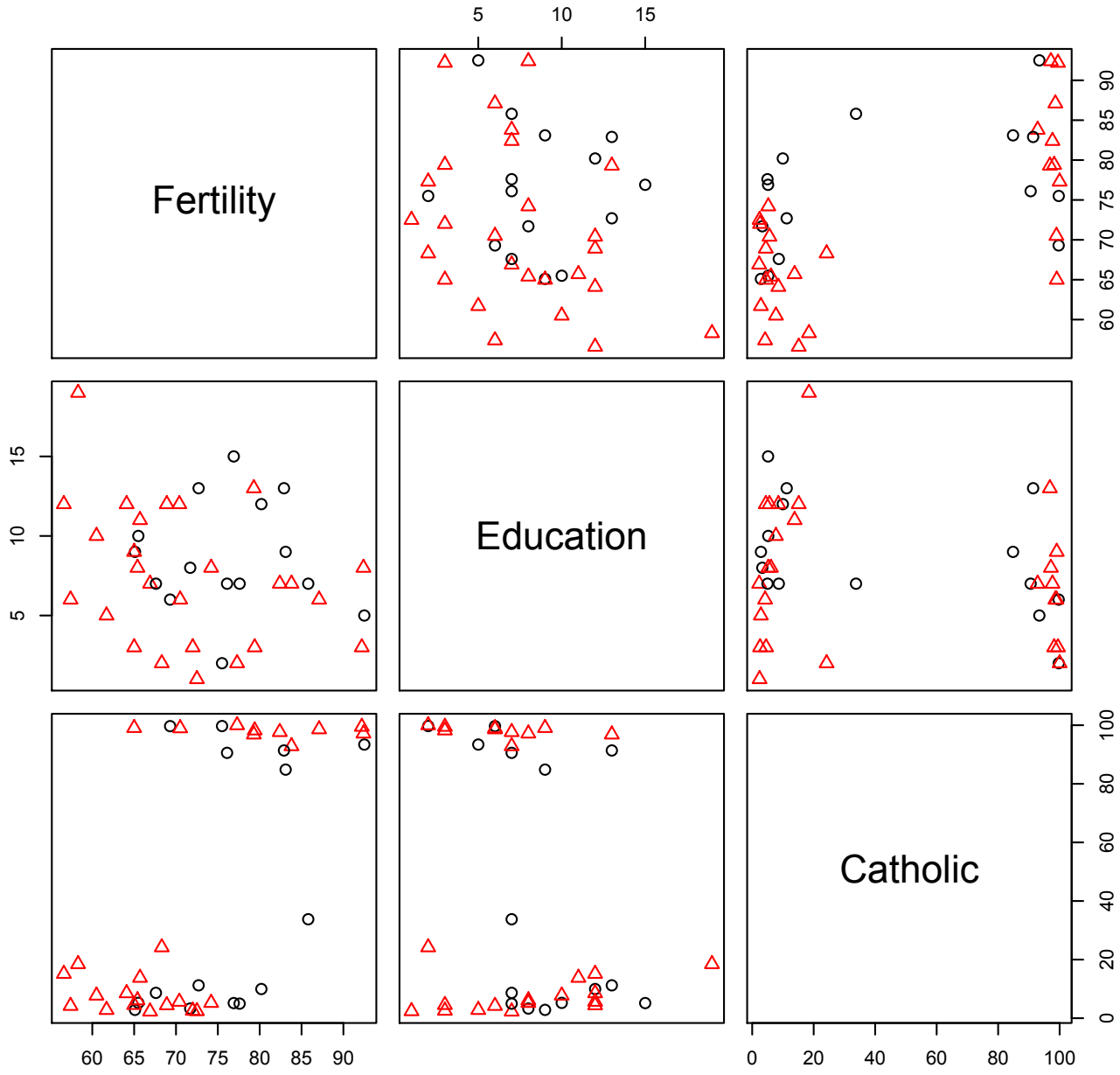
Definition (pairs)

```
# A matrix of scatterplots is produced.

## S3 method for class 'formula'
pairs(formula, data = NULL, ..., subset,
      na.action = stats::na.pass)

## Default S3 method:
pairs(x, labels, panel = points, ...,
      lower.panel = panel, upper.panel = panel,
      diag.panel = NULL, text.panel = textPanel,
      label.pos = 0.5 + has.diag/3,
      cex.labels = NULL, font.labels = 1,
      row1attop = TRUE, gap = 1)
```

Example (pairs 1)

```
> head(swiss)
             Fertility Agriculture Examination Education Catholic
Courtelary        80.2        17.0          15        12     9.96
Delemont          83.1        45.1           6         9    84.84
Franches-Mnt      92.5        39.7           5         5    93.40
Moutier           85.8        36.5          12         7    33.77
Neuveville        76.9        43.5          17        15     5.16
Porrentruy        76.1        35.3           9         7    90.57
             Infant.Mortality
Courtelary               22.2
Delemont                 22.2
Franches-Mnt             20.2
Moutier                  20.3
Neuveville               20.6
Porrentruy               26.6
> pairs(~Fertility+Education+Catholic, data=swiss,
+  subset=Education<20, main="Swiss data, Education<20",
+  col=1+(swiss$Agriculture > 50), cex=1.2,
+  pch=1+(swiss$Agriculture > 50))
```

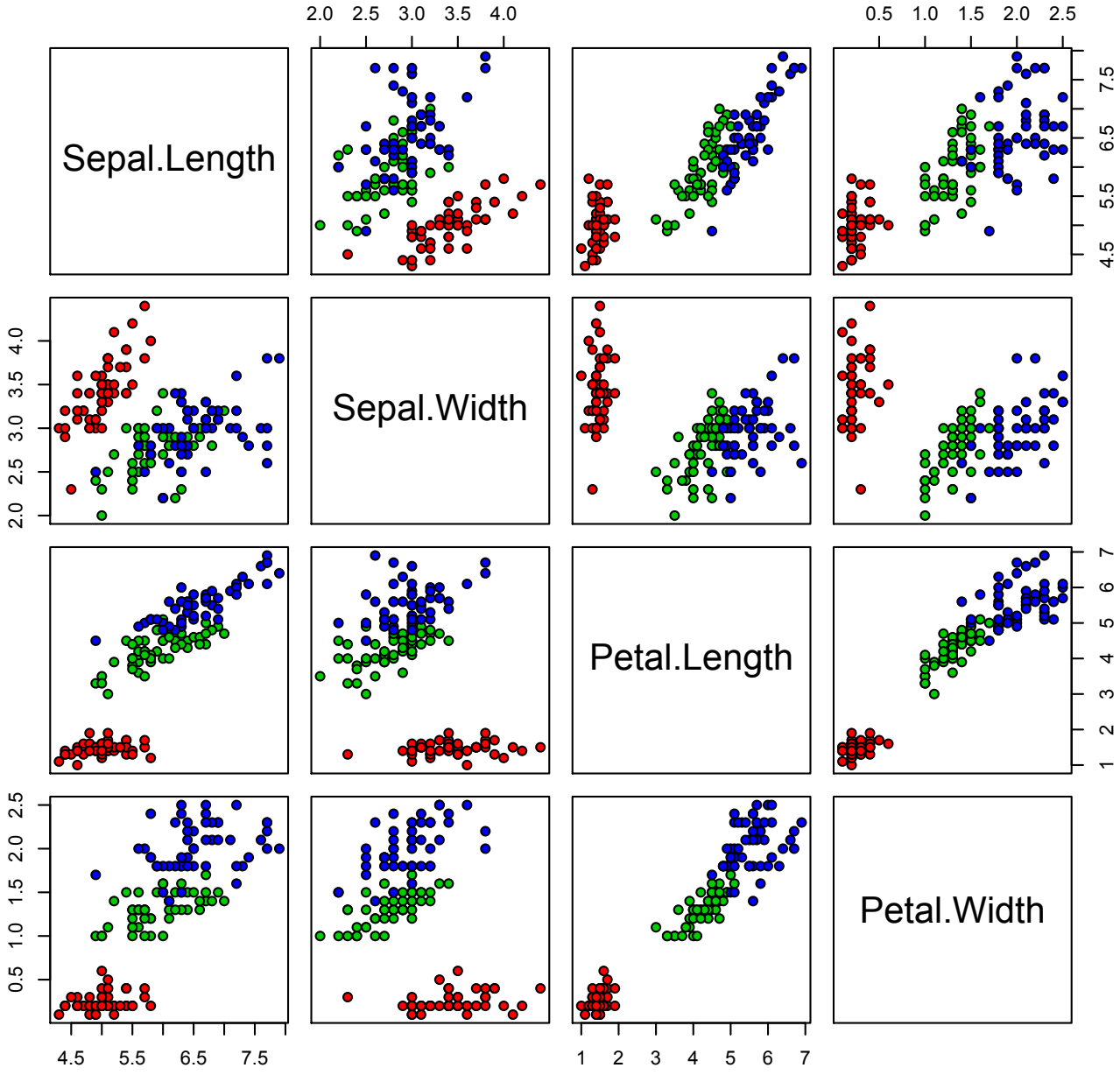
Swiss data, Education<20



Example (pairs 2)

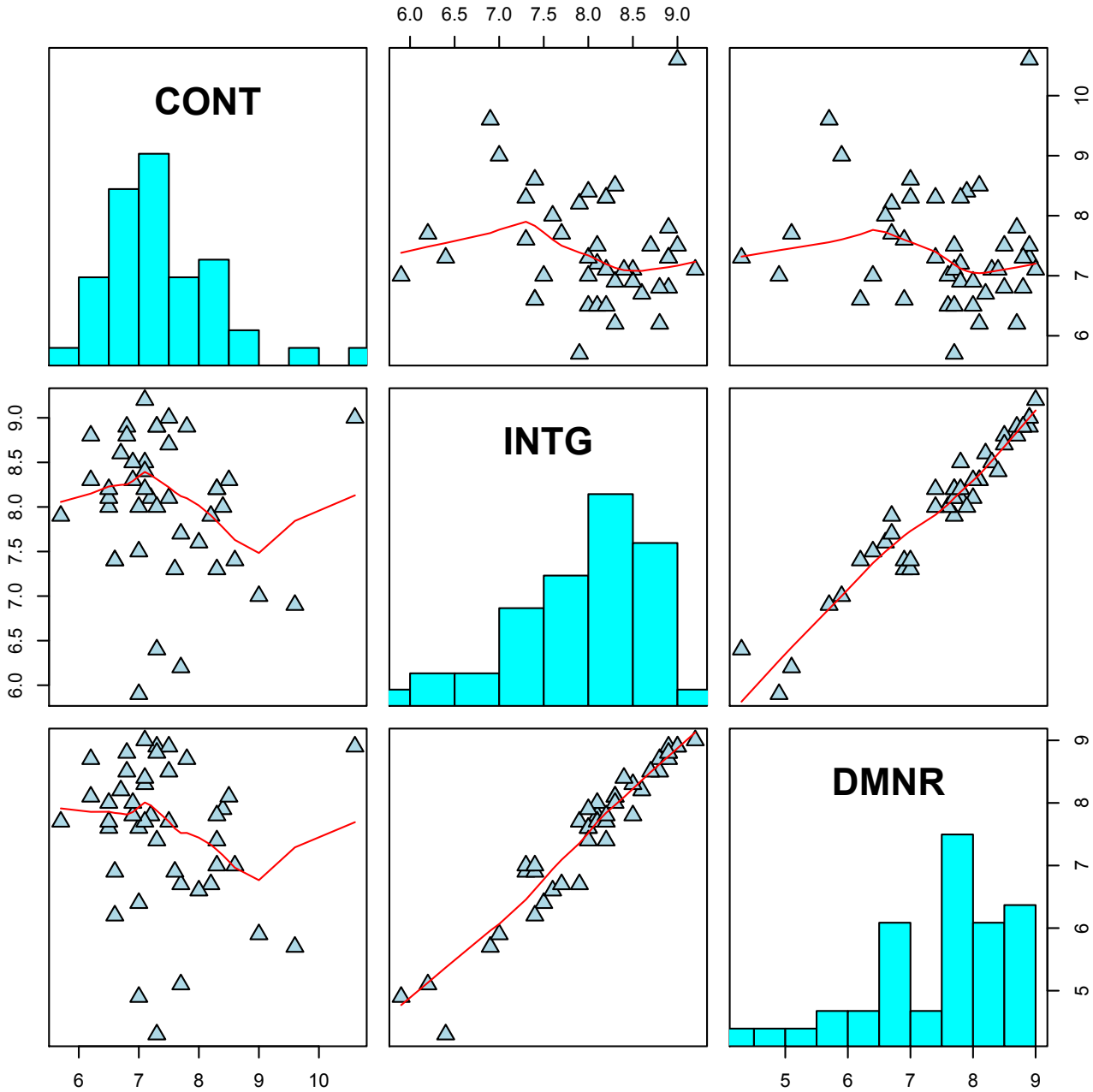
```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> 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



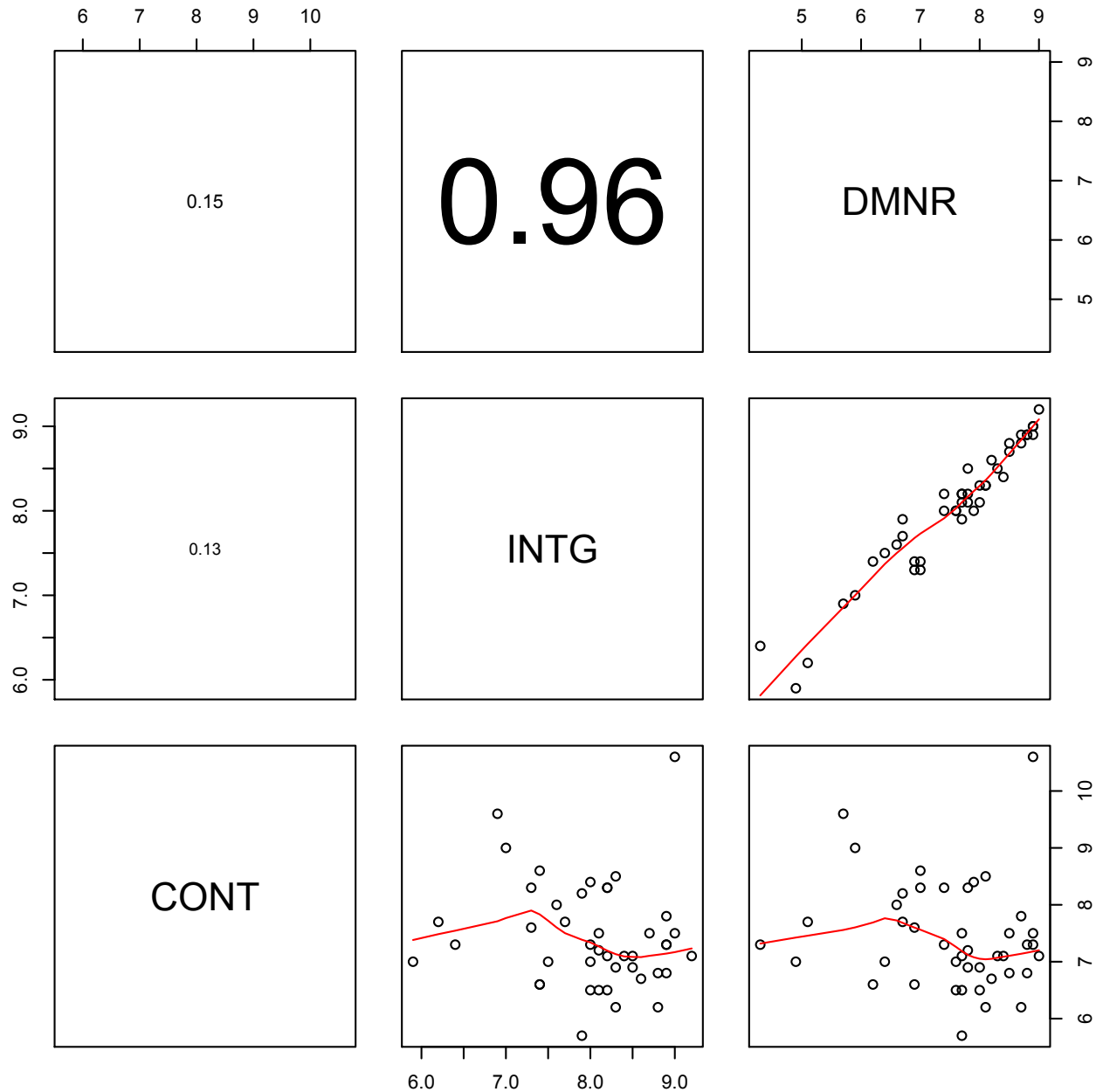
Example (pairs 3)

```
> panel.smooth
function (x, y, col = par("col"), bg = NA, pch = par("pch"),
    cex = 1, col.smooth = "red", span = 2/3, iter = 3, ...)
{
    points(x, y, pch = pch, col = col, bg = bg, cex = cex)
    ok <- is.finite(x) & is.finite(y)
    if (any(ok))
        lines(stats::lowess(x[ok], y[ok], f = span, iter = iter),
            col = col.smooth, ...)
}
<bytecode: 0x100eb0980>
<environment: namespace:graphics>
> 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", ...)
+ }
> head(USJudgeRatings)
               CONT INTG DMNR DILG CFMG DECI PREP FAMI ORAL WRIT PHYS RTEN
AARONSON,L.H.   5.7  7.9  7.7  7.3  7.1  7.4  7.1  7.1  7.1  7.0  8.3  7.8
ALEXANDER,J.M.  6.8  8.9  8.8  8.5  7.8  8.1  8.0  8.0  7.8  7.9  8.5  8.7
ARMENTANO,A.J.  7.2  8.1  7.8  7.8  7.5  7.6  7.5  7.5  7.3  7.4  7.9  7.8
BERDON,R.I.     6.8  8.8  8.5  8.8  8.3  8.5  8.7  8.7  8.4  8.5  8.8  8.7
BRACKEN,J.J.    7.3  6.4  4.3  6.5  6.0  6.2  5.7  5.7  5.1  5.3  5.5  4.8
BURNS,E.B.      6.2  8.8  8.7  8.5  7.9  8.0  8.1  8.0  8.0  8.0  8.6  8.6
> pairs(USJudgeRatings[,1:3], panel=panel.smooth, cex=1.5, pch=24,
+  bg="lightblue", diag.panel=panel.hist, cex.labels=2, font.labels=2)
```


CONT
INTG
DMNR
6.0 6.5 7.0 7.5 8.0 8.5 9.0
6 7 8 9 10
5 6 7 8 9

Example (pairs 4)

```
> 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 <- 0.8 / strwidth(txt)
+  text(0.5, 0.5, txt, cex=cex*r)
+ }
> pairs(USJudgeRatings[,1:3], row1attop=FALSE, gap=2,
+  lower.panel=panel.smooth, upper.panel=panel.cor)
```

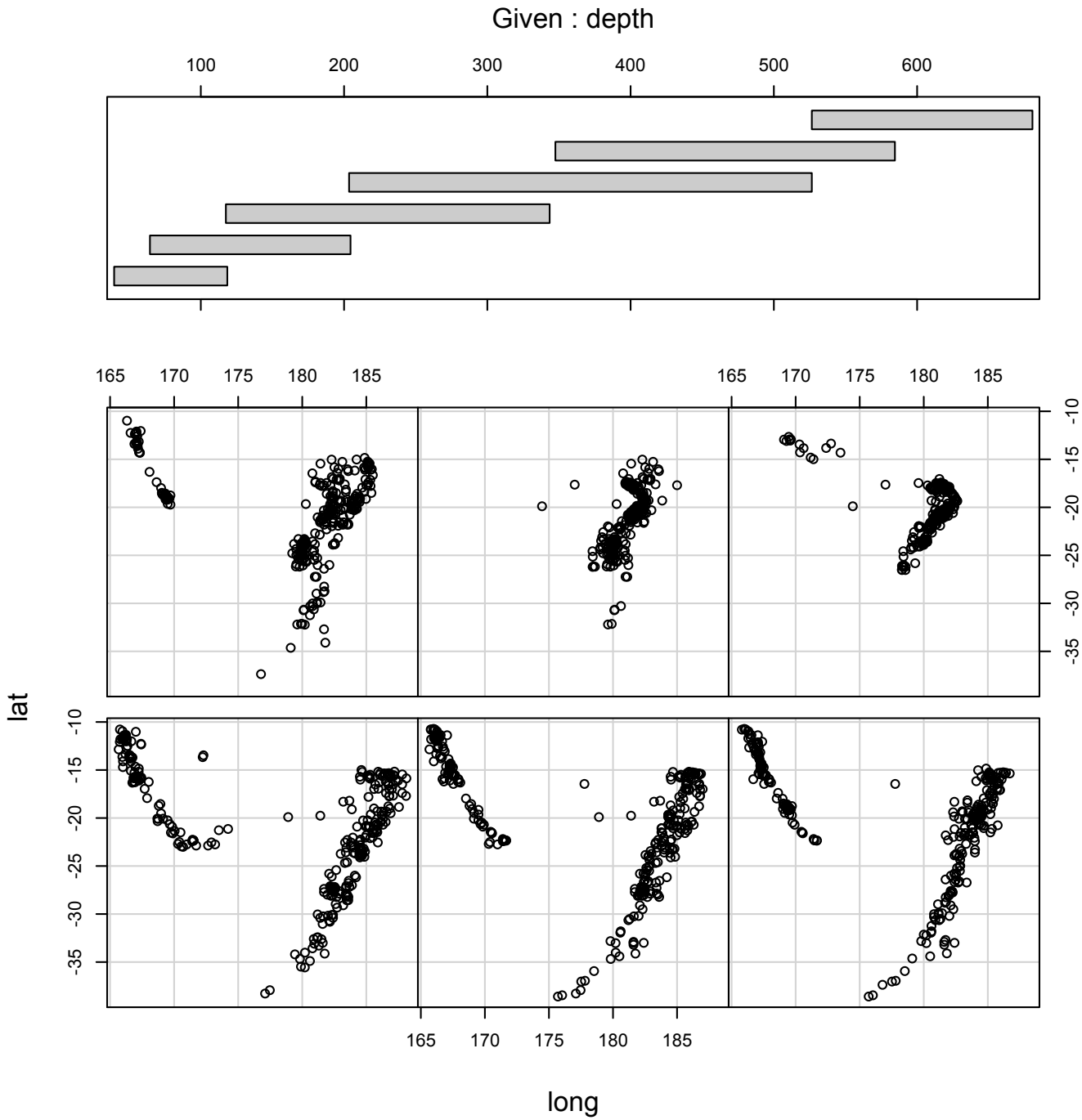

6 7 8 9 10
0.15
0.96
5 6 7 8 9
DMNR
9.0 8.0 7.0 6.0
0.13
INTG
CONT
6.0 7.0 8.0 9.0
10 9 8 7 6

Definition (coplot)

```
# This function produces two variants of the conditioning plots
# discussed in the reference below.

coplot(formula, data, given.values, panel = points, rows, columns,
       show.given = TRUE, col = par("fg"), pch = par("pch"),
       bar.bg = c(num = gray(0.8), fac = gray(0.95)),
       xlab = c(x.name, paste("Given :", a.name)),
       ylab = c(y.name, paste("Given :", b.name)),
       subscripts = FALSE,
       axlabels = function(f) abbreviate(levels(f)),
       number = 6, overlap = 0.5, xlim, ylim, ...)
co.intervals(x, number = 6, overlap = 0.5)
```

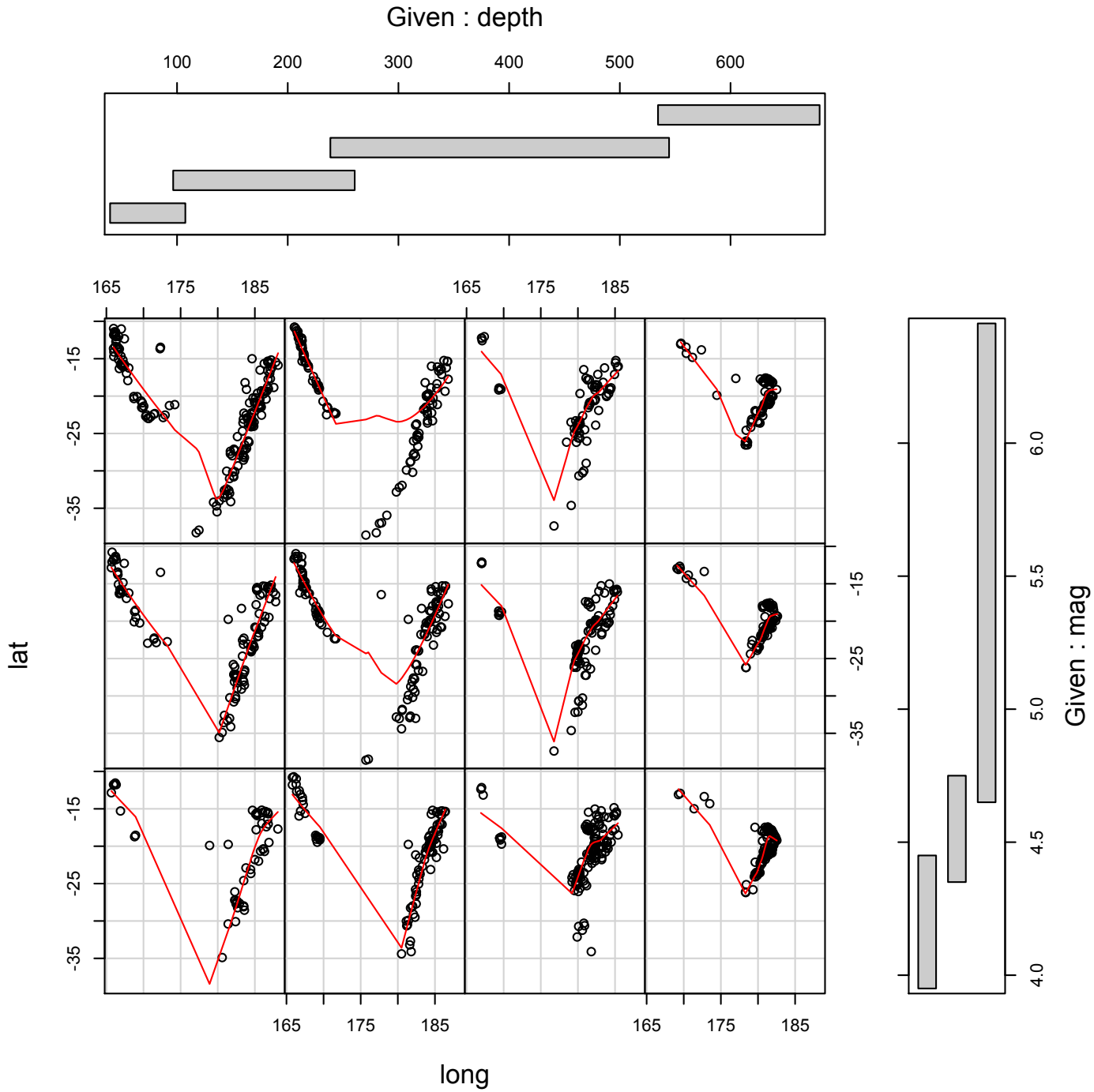
Example (coplot 1)

```
> head(quakes)
     lat   long depth mag stations
1 -20.42 181.62   562 4.8       41
2 -20.62 181.03   650 4.2       15
3 -26.00 184.10    42 5.4       43
4 -17.97 181.66   626 4.1       19
5 -20.42 181.96   649 4.0       11
6 -19.68 184.31   195 4.0       12
> summary(quakes$depth)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   40.0    99.0   247.0   311.4   543.0   680.0
> inter <- co.intervals(quakes$depth, number=4, overlap=0.1)
> inter
      [,1]  [,2]
[1,]  39.5 107.5
[2,]  96.5 260.5
[3,] 238.5 544.5
[4,] 534.5 680.5
> inter[,2]-inter[,1]
[1]  68 164 306 146
> is.data.frame(quakes)
[1] TRUE
> coplot(lat~long|depth, data=quakes)
```


Given : depth
100
200
300
400
500
600
165
170
175
180
185
-10
-15
-20
-25
-30
-35
lat
long

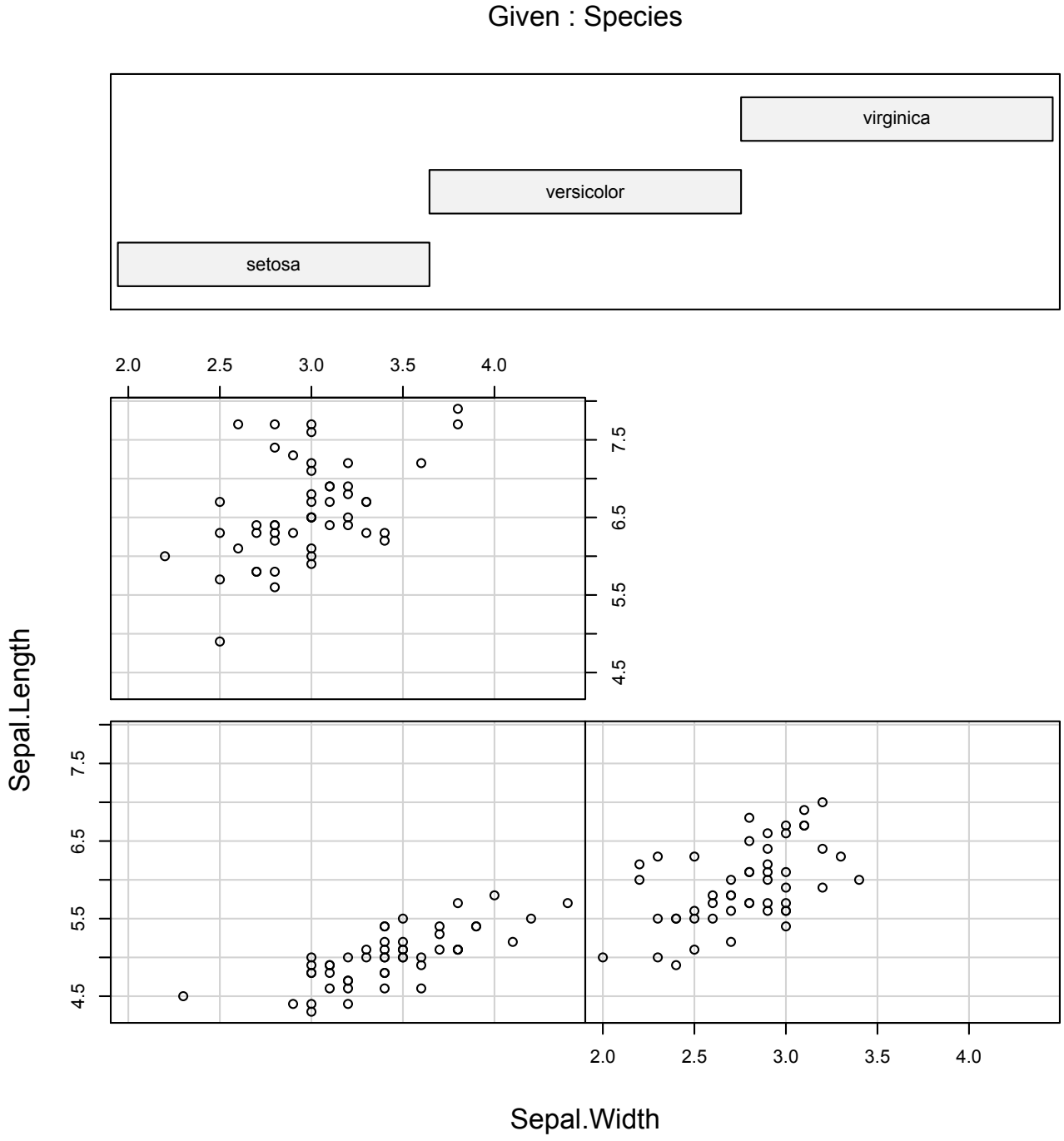
Example (coplot 2)

```
> inter1 <- co.intervals(quakes$depth, number=4, overlap=0.1)
> inter2 <- co.intervals(quakes$mag, number=3, overlap=0.1)
> inter <- list(inter1, inter2)
> formulas <- lat ~ long | depth * mag
> coplot(formulas, data=quakes, panel=panel.smooth, given.values=inter, rows=2)
```


Given : depth
100
200
300
400
500
600
165
175
185
-15
-25
-35
lat
long
Given : mag
4.0
4.5
5.0
5.5
6.0

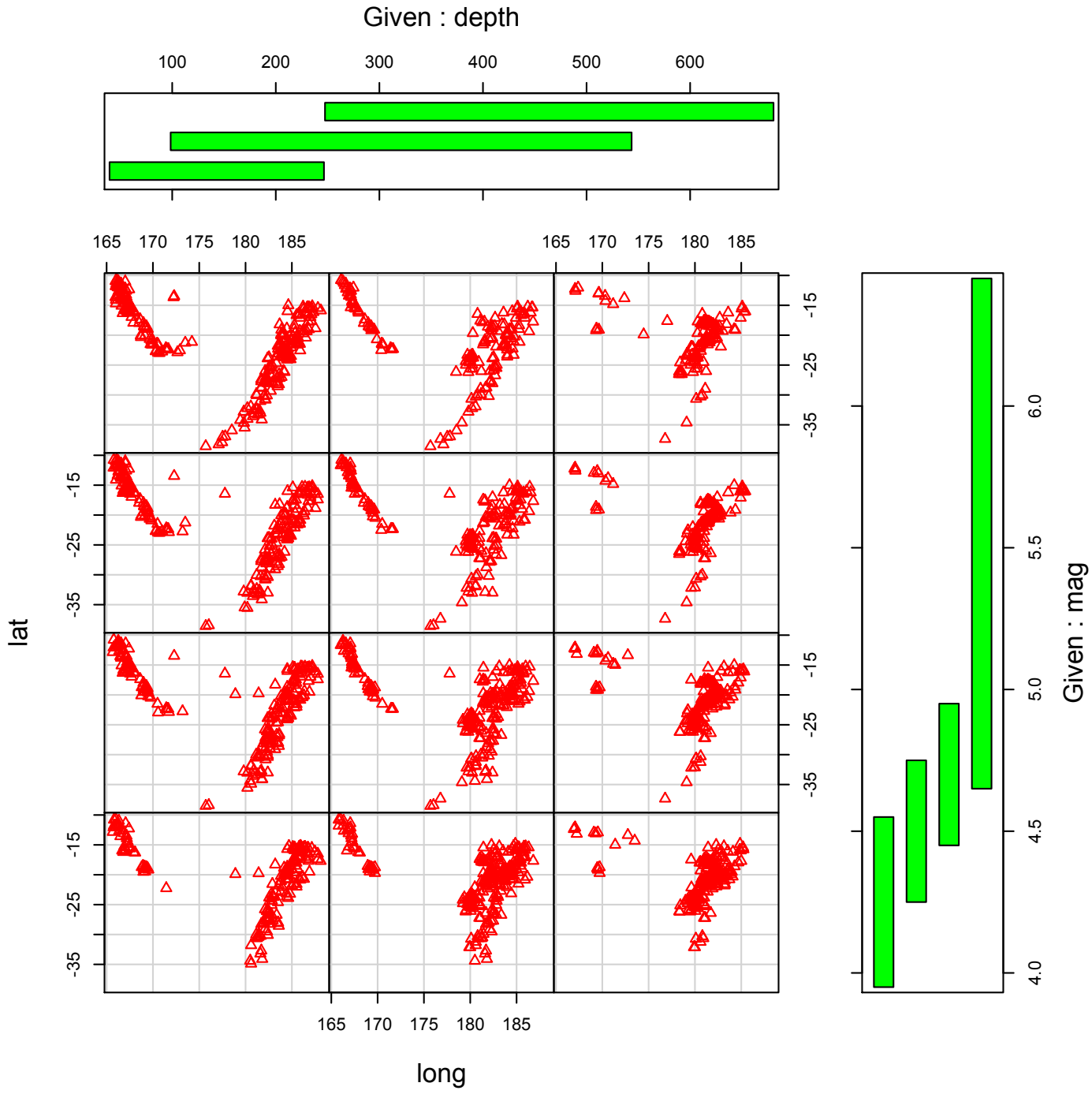
Example (coplot 3)

```
> dim(iris)
[1] 150   5
> is.data.frame(iris)
[1] TRUE
> names(iris)
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
> table(iris$Species)

    setosa versicolor  virginica
        50         50         50
> coplot(Sepal.Length~Sepal.Width|Species, data=iris)
```


Given : Species
virginica
versicolor
setosa
2.0
2.5
3.0
3.5
4.0
4.5
5.5
6.5
7.5
Sepal.Length
Sepal.Width

Example (coplot 4)

```
> formulas <- lat ~ long | depth * mag
> coplot(formulas, data=quakes, col="red", pch=2,
+  number=c(3,4), bar.bg=c(num="green", fac="blue"))
```

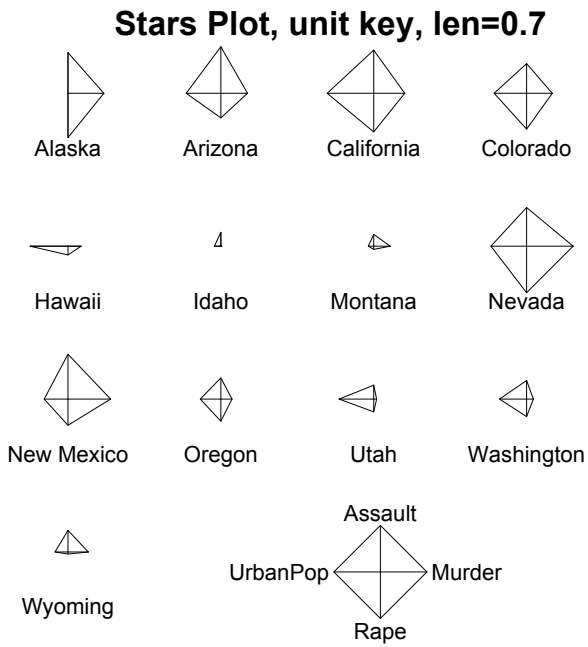

Given : depth
100
200
300
400
500
600
165
170
175
180
185
-15
-25
-35
lat
long
Given : mag
4.0
4.5
5.0
5.5
6.0

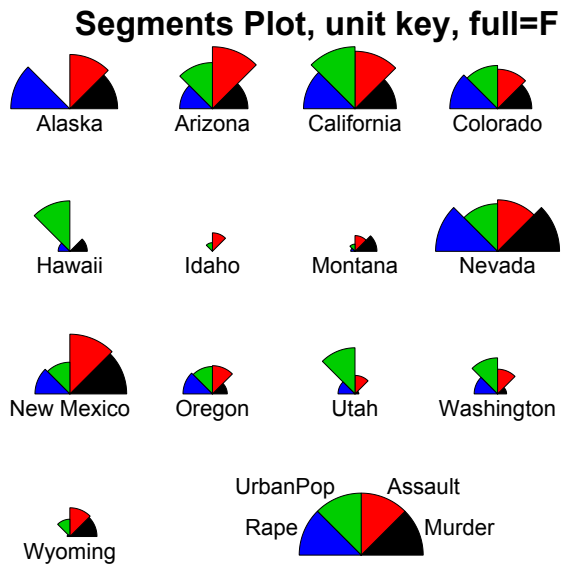
Definition (stars)

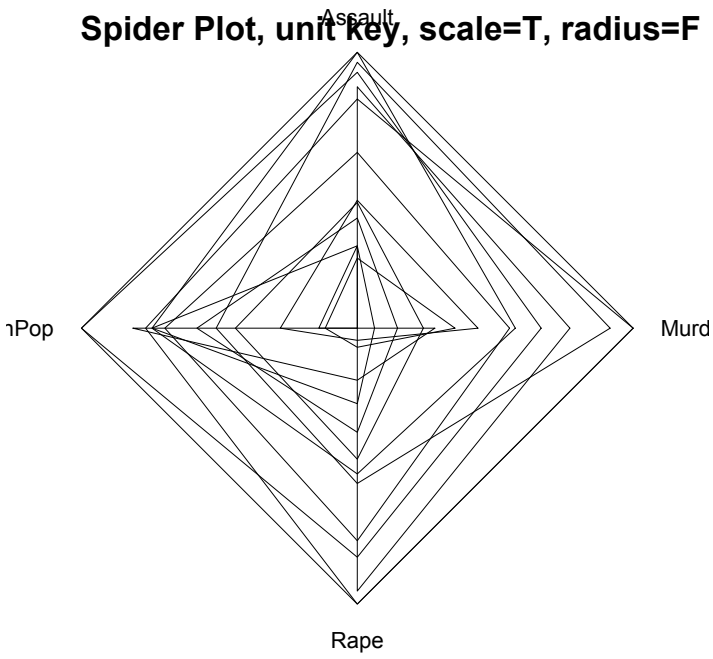
```
# Draw star plots or segment diagrams of a multivariate data set.
# With one single location, also draws ‘spider’ (or ‘radar’) plots.

stars(x, full = TRUE, scale = TRUE, radius = TRUE,
      labels = dimnames(x)[[1]], locations = NULL,
      nrow = NULL, ncol = NULL, len = 1,
      key.loc = NULL, key.labels = dimnames(x)[[2]],
      key.xpd = TRUE,
      xlim = NULL, ylim = NULL, flip.labels = NULL,
      draw.segments = FALSE,
      col.segments = 1:n.seg, col.stars = NA, col.lines = NA,
      axes = FALSE, frame.plot = axes,
      main = NULL, sub = NULL, xlab = "", ylab = "",
      cex = 0.8, lwd = 0.25, lty = par("lty"), xpd = FALSE,
      mar = pmin(par("mar"),
                 1.1+ c(2*axes+ (xlab != ""),
                 2*axes+ (ylab != ""), 1,0)),
      add = FALSE, plot = TRUE, ...)
```

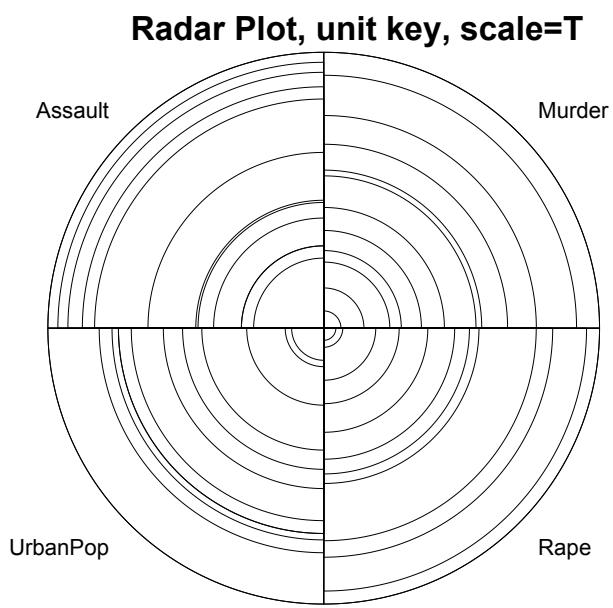
Example (stars)

```
> head(USArrests)
           Murder Assault UrbanPop Rape
Alabama      13.2     236       58 21.2
Alaska       10.0     263       48 44.5
Arizona       8.1     294       80 31.0
Arkansas      8.8     190       50 19.5
California    9.0     276       91 40.6
Colorado      7.9     204       78 38.7
> state.region
 [1] South         West          West          South         West
 [6] West          Northeast     South         South         South
...
[41] North Central South         South         West          Northeast
[46] South         West          South         North Central West
Levels: Northeast South North Central West
> WESTarrests <- USArrests[state.region == "West", ]
> par(mfrow=c(2,2))
> stars(WESTarrests, draw.segments=FALSE, len=0.7, key.loc=c(7,2))
> title(main="Stars Plot, unit key, len=0.7")
> stars(WESTarrests, draw.segments=TRUE, full=FALSE, key.loc=c(7,2))
> title(main="Segments Plot, unit key, full=F")
> stars(WESTarrests, locations=c(0,0), scale=TRUE, radius=FALSE,
+  col.stars=0, key.loc=c(0,0))
> title(main="Spider Plot, unit key, scale=T, radius=F")
> stars(WESTarrests, locations=0:1, scale=TRUE, draw.segments=TRUE,
+  col.segments=0, col.stars=0, key.loc=0:1)
> title(main="Radar Plot, unit key, scale=T")
```


Stars Plot, unit key, len=0.7
Alaska
Arizona
California
Colorado
Hawaii
Idaho
Montana
Nevada
New Mexico
Oregon
Utah
Washington
Wyoming
Assault
UrbanPop
Murder
Rape


Segments Plot, unit key, full=F
Alaska
Arizona
California
Colorado
Hawaii
Idaho
Montana
Nevada
New Mexico
Oregon
Utah
Washington
Wyoming
UrbanPop
Assault
Rape
Murder


Spider Plot, unit key, scale=T, radius=F
Assault
ıPop
Murd
Rape


Radar Plot, unit key, scale=T
Assault
Murder
UrbanPop
Rape

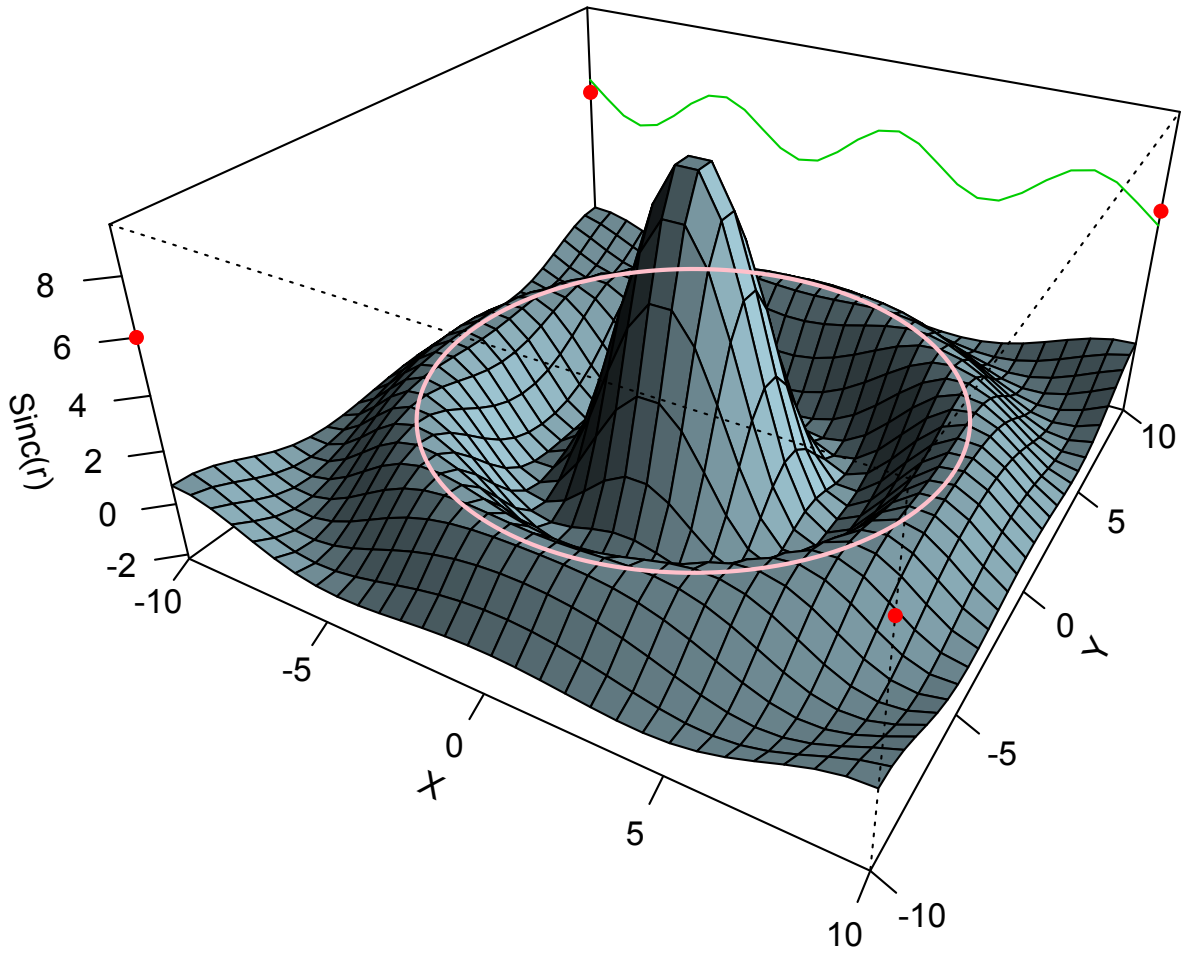
Definition (persp)

```
# This function draws perspective plots of a surface over the x-y plane.
# persp is a  generic function.

## Default S3 method:
persp(x = seq(0, 1, length.out = nrow(z)),
      y = seq(0, 1, length.out = ncol(z)),
      z, xlim = range(x), ylim = range(y),
      zlim = range(z, na.rm = TRUE),
      xlab = NULL, ylab = NULL, zlab = NULL,
      main = NULL, sub = NULL,
      theta = 0, phi = 15, r = sqrt(3), d = 1,
      scale = TRUE, expand = 1,
      col = "white", border = NULL, ltheta = -135, lphi = 0,
      shade = NA, box = TRUE, axes = TRUE, nticks = 5,
      ticktype = "simple", ...)
```

Example (persp 1)

```
> 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
> 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
> title(main="Perspective Plots with Sinc Function")
> 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
> trans3d <- function(x, y, z, pmat){
+  tr <- cbind(x, y, z, 1) %*% pmat
+  list(x=tr[,1]/tr[,4], y=tr[,2]/tr[,4])
+ }
> xE <- c(-10, 10); xy <- expand.grid(xE, xE)
> points(trans3d(xy[,1], xy[,2], 6, pm=res), col=2, pch=16)
> lines(trans3d(x, y=10, z=6+sin(x), pm=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)
```

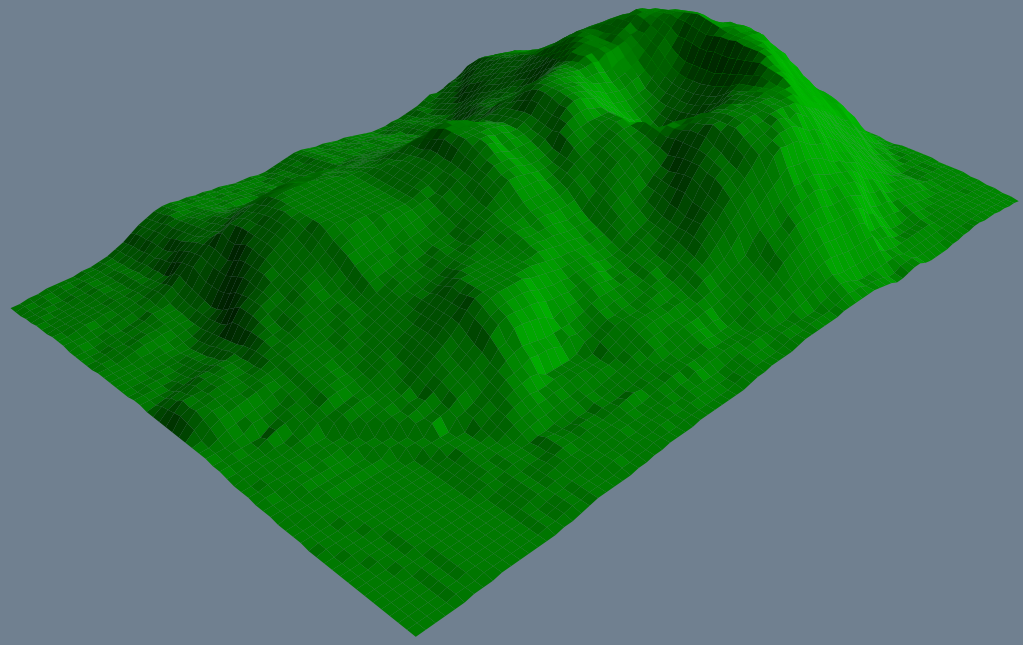
Perspective Plots with Sinc Function



Example (persp 2)

```
> dim(volcano)
[1] 87 61
> z <- 2*volcano
> x <- 10*(1:nrow(z)) # 10 meter spacing (S to N)
> y <- 10*(1:ncol(z)) # 10 meter spacing (E to W)
> par(bg="slategray")
> persp(x, y, z, theta=135, phi=30, col="green3", scale=FALSE,
+  ltheta=-120, shade=0.75, border=NA, box=FALSE)
> title("Perspective Plots with volcano")
```

Perspective Plots with volcano

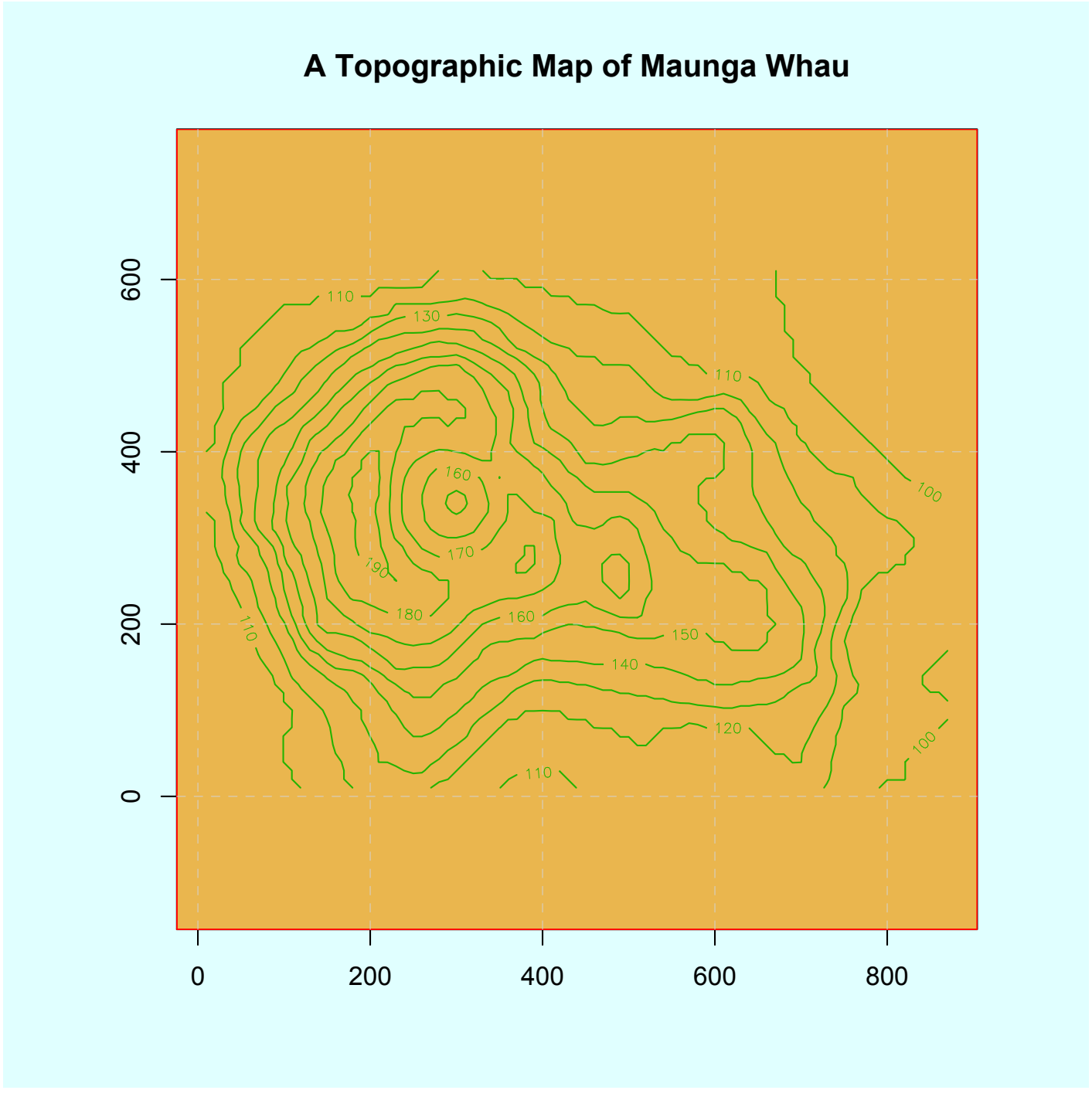


Definition (contour)

```
# contour: create a contour plot, or add contour lines to an existing plot.
# contourLines: calculate contour lines for a given set of data.

contour(x = seq(0, 1, length.out = nrow(z)),
        y = seq(0, 1, length.out = ncol(z)),
        z,
        nlevels = 10, levels = pretty(zlim, nlevels),
        labels = NULL,
        xlim = range(x, finite = TRUE),
        ylim = range(y, finite = TRUE),
        zlim = range(z, finite = TRUE),
        labcex = 0.6, drawlabels = TRUE, method = "flattest",
        vfont, axes = TRUE, frame.plot = axes,
        col = par("fg"), lty = par("lty"), lwd = par("lwd"),
        add = FALSE, ...)

contourLines(x = seq(0, 1, length.out = nrow(z)),
             y = seq(0, 1, length.out = ncol(z)),
             z, nlevels = 10,
             levels = pretty(range(z, na.rm=TRUE), nlevels))
```

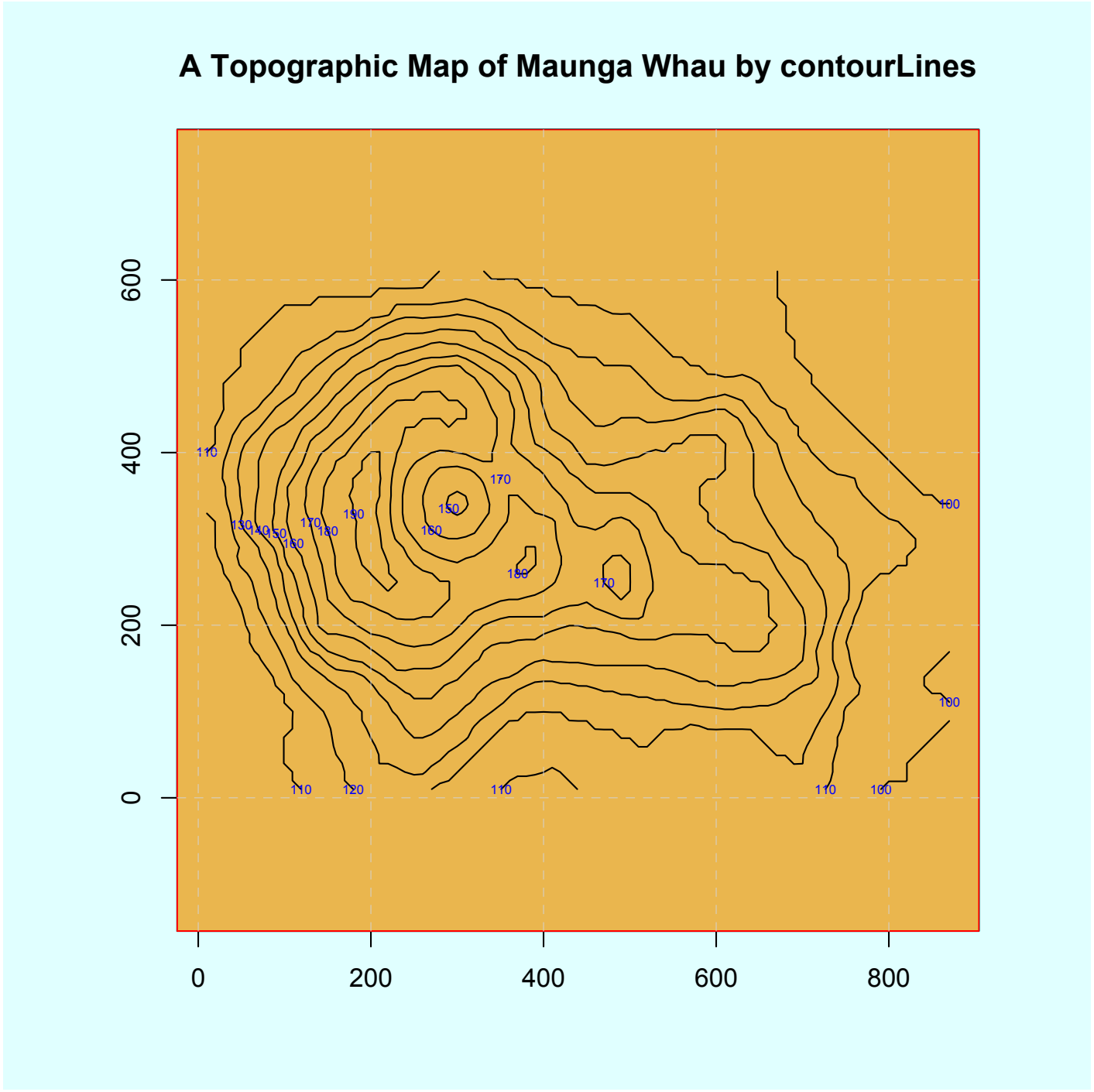
Example (contour)

```
> rx <- range(x <- 10 * 1:nrow(volcano))
> ry <- range(y <- 10 * 1:ncol(volcano))
> ry <- ry + c(-1,1) * (diff(rx)-diff(ry)) /2
> tcol <- terrain.colors(12)
> par(pty="s", bg="lightcyan")
> plot(x=0, y=0, type="n", xlim=rx, ylim=ry, xlab="", ylab="")
> u<-par("usr")
> rect(u[1],u[3],u[2],u[4],col=tcol[8],border="red")
> contour(x, y, volcano, col=tcol[2], lty="solid", add=TRUE,
+  vfont=c("sans serif", "plain"))
> title("A Topographic Map of Maunga Whau", font=4)
> abline(h=200* 0:4, v=200 * 0:4, col="lightgray", lty=2, lwd=0.1)
```


A Topographic Map of Maunga Whau
600
400
200
0
0
200
400
600
800
110
130
110
160
170
190
180
160
150
140
120
110
100
100

Example (contour)

```
> line.list <- contourLines(x, y, volcano)
> par(pty="s", bg="lightcyan")
> plot(x=0, y=0, type="n", xlim=rx, ylim=ry, xlab="", ylab="")
> rect(u[1], u[3], u[2], u[4], col=tcol[8], border="red")
> is.list(line.list)
[1] TRUE
> length(line.list)
[1] 20
> names(line.list[[1]])
[1] "level" "x"     "y"
> line.list[[1]]
$level
[1] 100
$x
 [1] 870.0000 860.0000 850.9173 850.0000 840.0000 830.9173 830.0000 820.9173
...
[41] 680.9173 680.9173 680.0000 670.9173 670.9173 670.9173 670.9173
$y
 [1] 340.9173 340.9173 350.0000 350.9173 350.9173 360.0000 360.9173 370.0000
...
[41] 560.0000 570.0000 570.9173 580.0000 590.0000 600.0000 610.0000
> tmplines <- function(clines){
+  lines(clines[[2]], clines[[3]])
+  text(clines[[2]][1], clines[[3]][1], clines[[1]][1], cex=0.5, col="blue")
+ }
> invisible(lapply(line.list, tmplines))
> title("A Topographic Map of Maunga Whau by contourLines", font=4)
> abline(h=200 * 0:4, v=200* 0:4, col="lightgray", lty=2, lwd=0.1)
```


A Topographic Map of Maunga Whau by contourLines
600
400
200
0
0
200
400
600
800
110
170
150
190
130
140
150
160
170
180
160
180
110
100
100
110
120
110
110
100

Definition (image)

```
# Creates a grid of colored or gray-scale rectangles with colors
# corresponding to the values in z. This can be used to display three-dimensional
# or spatial data aka images. This is a generic function.
#
# The functions heat.colors, terrain.colors and topo.colors create heat-spectrum
# (red to white) and topographical color schemes suitable for displaying
# ordered data, with n giving the number of colors desired.

image(x, y, z, zlim, xlim, ylim, col = heat.colors(12),
      add = FALSE, xaxs = "i", yaxs = "i", xlab, ylab,
      breaks, oldstyle = FALSE, useRaster, ...)
```

Example (image 1)

```
> image(volcano, zlim=c(150,200), xaxs="r", yaxs="r",
+  xlab="West to East", ylab="South to North")
> image(volcano, zlim=c(0,150), add=T, col=cm.colors(12),
+  xlab="0 to 1", ylab="0 to 1")
> title(main="image & add image")
```

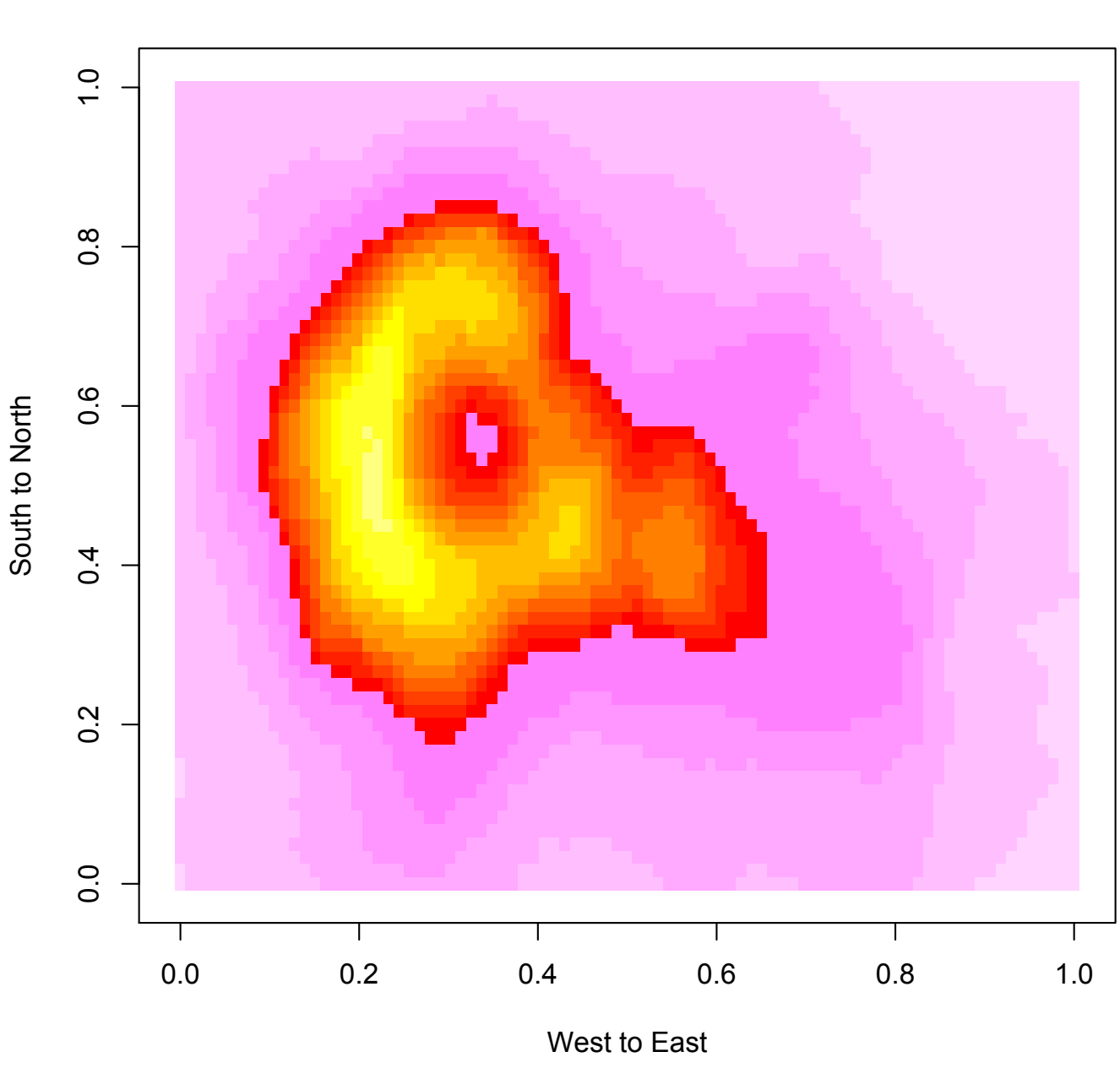
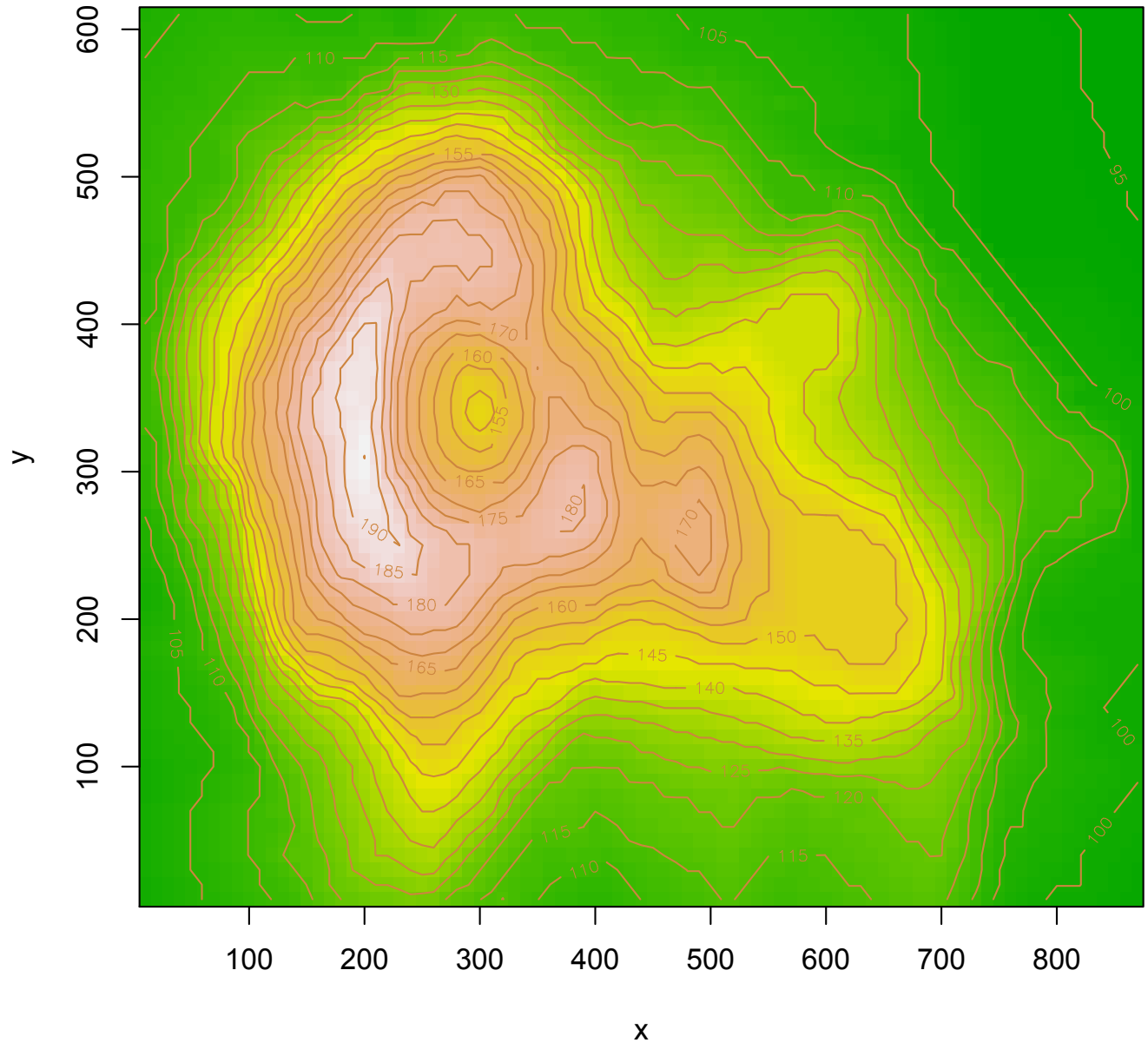


image & add image
South to North
0.0
0.2
0.4
0.6
0.8
1.0
West to East
0.0
0.2
0.4
0.6
0.8
1.0

Example (image 2)

```
> x <- 10 * (1:nrow(volcano))
> y <- 10 * (1:ncol(volcano))
> image(x, y, volcano, col=terrain.colors(100), axes=FALSE)
> contour(x, y, volcano, levels=seq(90, 200, by=5),
+  add=TRUE, col="peru")
> axis(1, at=seq(100,800,by=100))
> axis(2, at=seq(100,600,by=100))
> box()
> title(main="Maunga Whau Volcano", font.main=4)
```

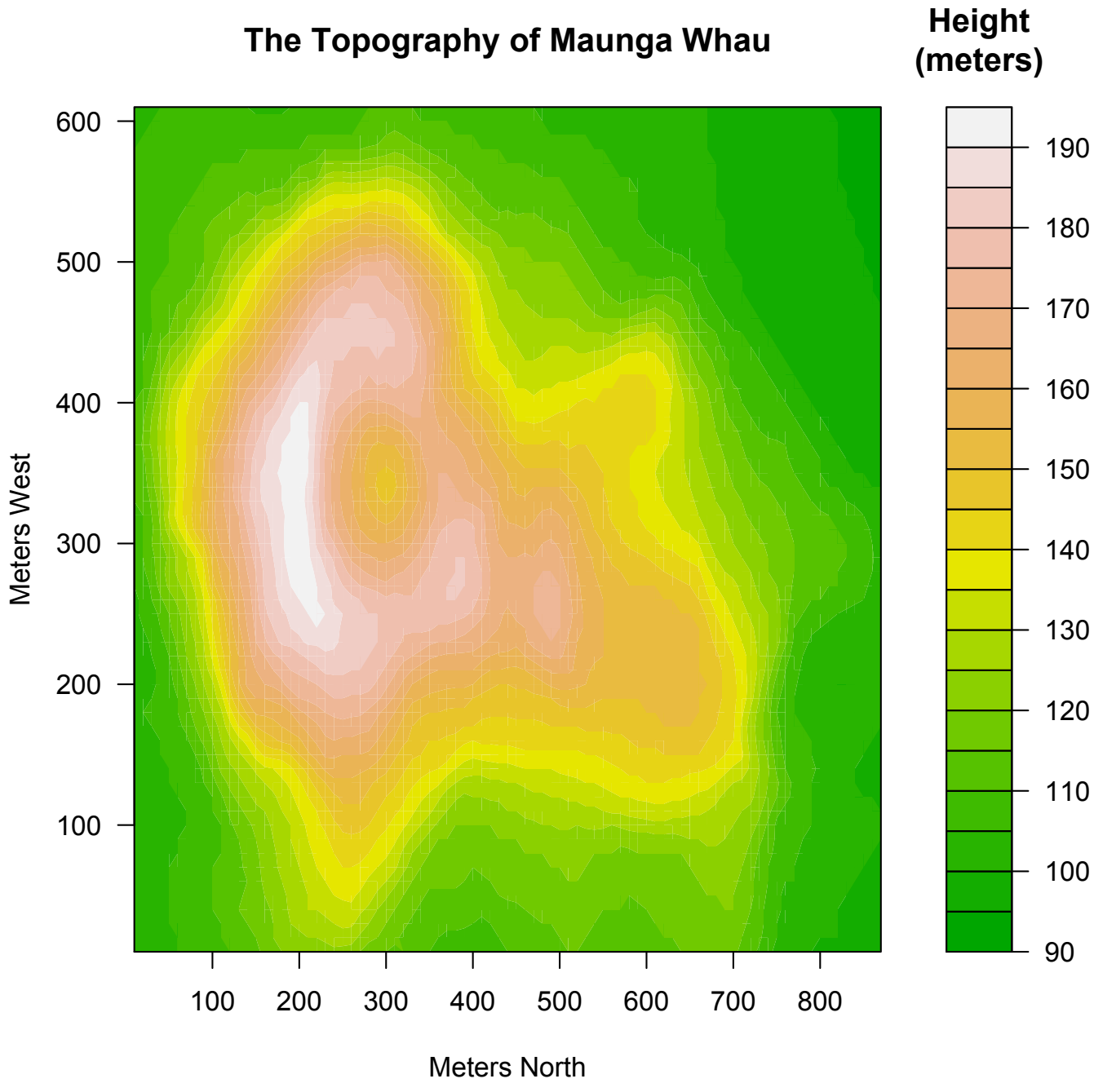

Maunga Whau Volcano
y
x
100
200
300
400
500
600
700
800
105
110
115
130
155
110
95
170
160
155
165
175
180
190
185
180
160
150
145
140
135
125
120
115
110
115
105
110
165
100
100
100

Definition (filled.contour)

```
# This function produces a contour plot with the areas between the contours
# filled in solid color (Cleveland calls this a level plot). A key showing
# how the colors map to z values is shown to the right of the plot.

filled.contour(x = seq(0, 1, length.out = nrow(z)),
               y = seq(0, 1, length.out = ncol(z)),
               z,
               xlim = range(x, finite=TRUE),
               ylim = range(y, finite=TRUE),
               zlim = range(z, finite=TRUE),
               levels = pretty(zlim, nlevels), nlevels = 20,
               color.palette = cm.colors,
               col = color.palette(length(levels) - 1),
               plot.title, plot.axes, key.title, key.axes,
               asp = NA, xaxs = "i", yaxs = "i", las = 1,
               axes = TRUE, frame.plot = axes, ...)
```

Example (fiiled.contour)

```
> x <- 10 * (1:nrow(volcano))
> y <- 10 * (1:ncol(volcano))
> filled.contour(x, y, volcano, color=terrain.colors,
+  plot.title=title(main="The Topography of Maunga Whau",
+  xlab="Meters North", ylab="Meters West"),
+  plot.axes={ axis(1, seq(100,800,by=100))
+  axis(2, seq(100,600,by=100))},
+  key.title=title(main="Height\n(meters)"),
+  key.axes=axis(4,seq(90,190,by=10)))
```


The Topography of Maunga Whau
Height (meters)
Meters West
Meters North
600
500
400
300
200
100
100
200
300
400
500
600
700
800
190
180
170
160
150
140
130
120
110
100
90