50. 高水準作図関数

種々の高水準作図関数を紹介する.

散布図：plot()

plot() の詳しい説明は 前々節 を御覧頂きたい. ここでは, plot() への引数の与え方による出力の違いを見てみる.

plot(x)

- ベクトル x の要素が実数ならば, x は時系列データとみなされ, 横軸を自然数, 縦軸をデータ x 要素とする時系列プロットが描かれる.
- ベクトル x が時系列データならば, そのまま時系列プロットが描かれる.
- ベクトル x の要素が複素数ならば, 横軸を実数, 縦軸を虚部とするプロットが描かれる.
- x が2列の行列ならば, 横軸を一列目, 縦軸を2列目とするプロットが描かれる.
- x が2次元リストならば, その要素を横軸, 縦軸としてプロットが描かれる. ただし names() を使ってどちらが x なのか y なのかラベルをつける必要がある.

```
plot(rnorm(10))
x <- list(1:5, 3:7);   names(x) <- c("X", "Y")
plot(x)
```

plot(x, y)

データが入っているベクトル x, y やリスト x, y を点の座標として与えると散布図を描く.

```
x <- rnorm(10)
y <- rnorm(10)
plot(x,y)
```

plot(y ~ x)

以下の様に回帰式として入力することも出来る.

```
x <- rnorm(10)
y <- rnorm(10)
plot(y ~ x)
```

plot(f)

f は因子オブジェクトである．f の棒グラフを描く．

```
f <- factor(c(rep("A",3), rep("B",5)))
plot(f)
```

plot(f, y)

f は因子オブジェクト，y は数値ベクトルである．f の各水準に対する y の箱ひげ図を描くときに使う．

```
x <- factor(c(rep("A",3), rep("B",5)))
y <- rnorm(8)
plot(f, y)
```

plot(df)

df はデータフレームである．データフレーム中の変量のプロットを行う．

plot(~ expr)

expr は "+" で仕切られたオブジェクト名のリスト (例えば a+b+c) である．名前が与えられたオブジェクトの分布関数のプロットを行う．

```
plot(~ group, data=sleep)
plot(~ extra, data=sleep)
```

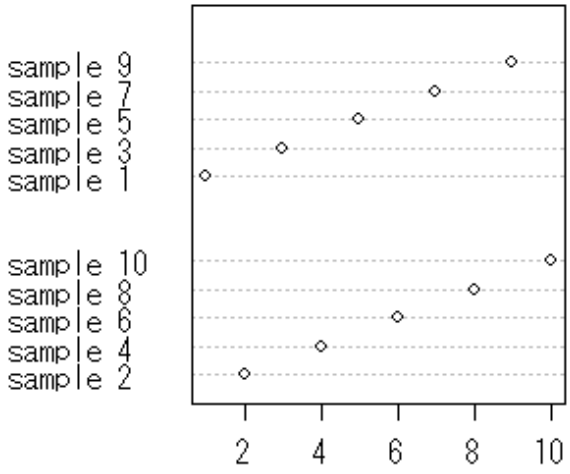
plot(y ~ expr)

y は任意のオブジェクト，expr は "+" で仕切られたオブジェクト名のリスト（例えば a+b+c ）である．expr に名前が与えられた全てのオブジェクトに対して y のプロットを行う．

関数 dotchart()

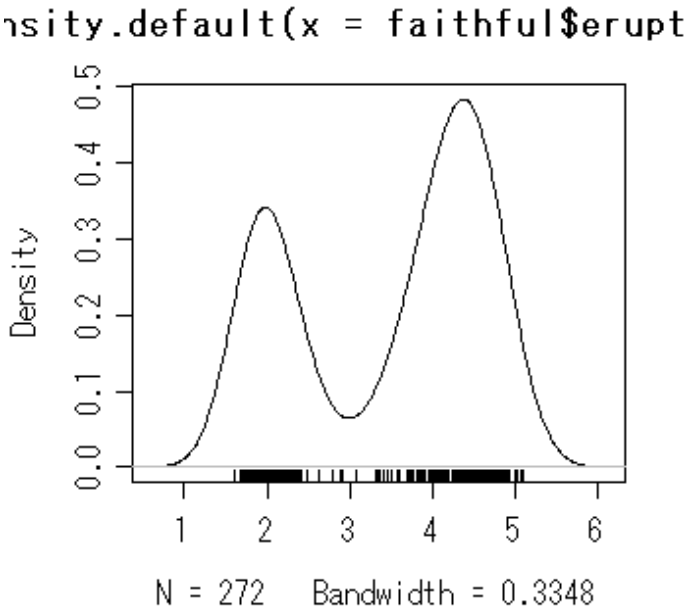
関数 dotchart() でドットチャートを描く棒グラフの棒の代わりに点でプロットする．

```
# dotchart(データ, groups=グループのベクトル, gdata=NULL, labels=ラベル, ...)

dotchart(1:10, group=c(rep(1:2,5)), labels=paste("sample", 1:10))
```



- 似たような関数に rug() がある

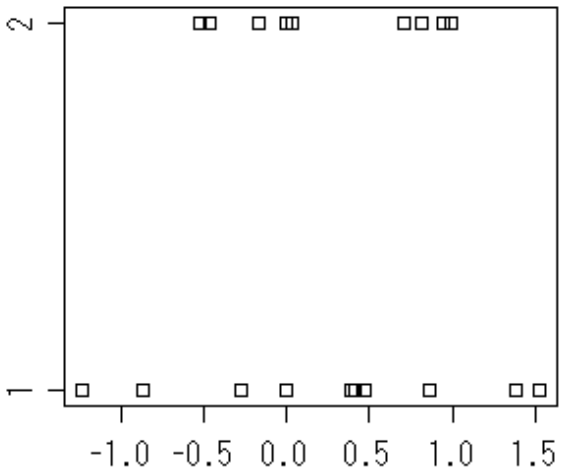
```
plot(density(faithful$eruptions))
rug(faithful$eruptions, side=1)
```



関数 stripchart()

関数 stripchart() で因子レベル別に一次元散布図を描く.

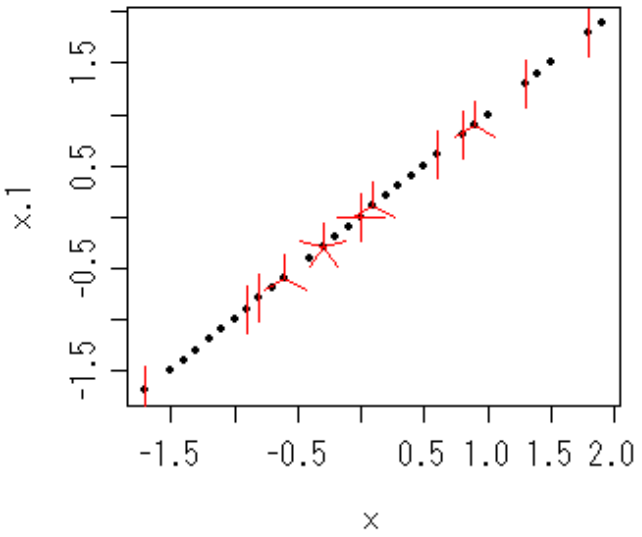
```
# stripchart(データ ~ 因子ベクトル, vertical = F, group.names=,...)

 y <- rnorm(20);   x <- factor(rep(1:2,10))
 stripchart(y ~ x)
```



関数 sunflowerplot()

関数 sunflowerplot() でヒマワリ図を描く．これは 1 点の周りに複数データが対応する際に，点の周りに重なった分だけ花弁を描く．

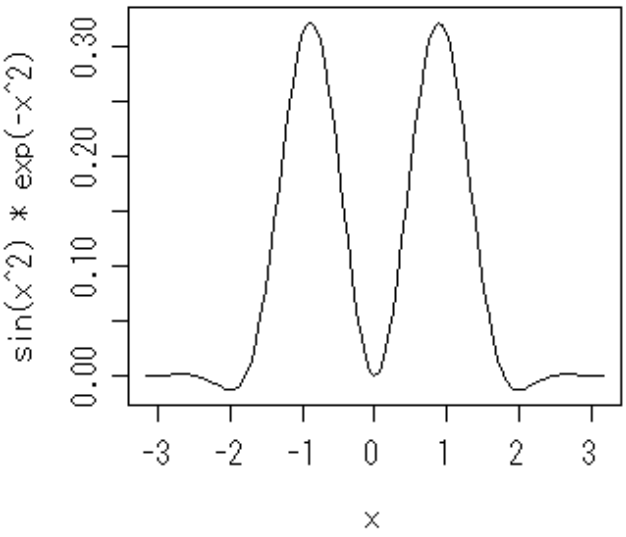
```
# sunflowerplot(x軸データ, y軸データ, digits=6, xlab = "", xlim = NULL, add = F, ...)

 x <- round(rnorm(50), d=1)
 y <- data.frame(x, x)
 sunflowerplot(y)
```



関数 curve()

関数 curve() を用いれば，関数を直接指定してグラフを出力することも出来る．引数 n でポイントの数を指定することが出来る．

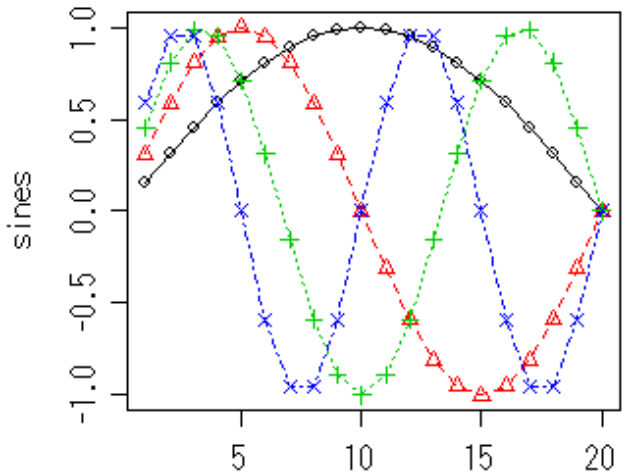
```
# curve(関数, from=下限, to=上限, n = 点の数, add = FALSE, type = "l", xlim = NULL, ...)

 curve(sin(x^2)*exp(-x^2),-pi,pi)
```



関数 matplot()

matplot() は行列を引数にとり，その各列についてプロットを行う．完成図は，座標設定や列ごとにマーカーの色や形を自動的に設定し，見た目に区別がつくように描かれる．また，上書き用に 関数 matpoints() ，matlines() が用意されている．

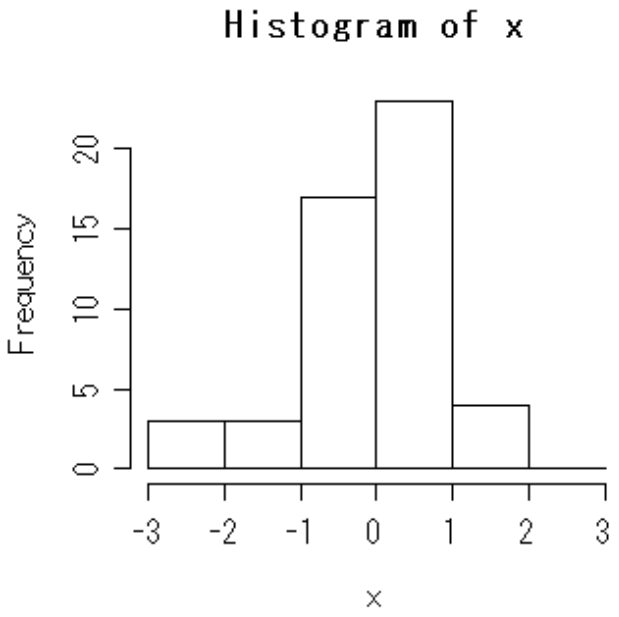
```
# matplot(行列（もしくは2つのベクトル）, type = "p", lty = 1:5, lwd = 1, pch = NULL, col = 1:6, ...)

 myfunc <- function(x,y) sin(x/20 * pi * y)
 sines  <- outer(1:20, 1:4, myfunc)
 matplot(sines, pch = 1:4, type = "o")
```

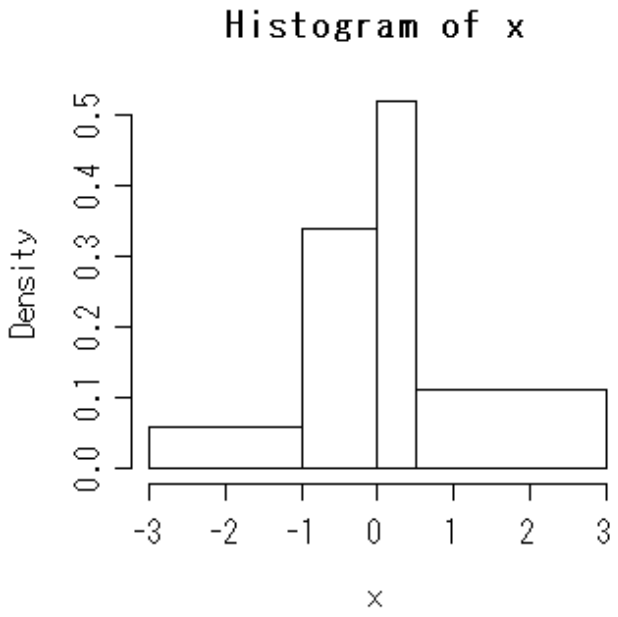


ヒストグラム：hist()

関数 hist() でヒストグラムを描くことが出来る．異なる区切り幅のヒストグラムを描くことも出来るが，主観的なバイアス (偏り) が入る危険があるのでお勧めは出来ない．

```
x <- rnorm(50)                          # 一次元データ
hist(x, breaks = seq(-3,3,1) )          # -3 から -3 まで 1 ずつの幅で描く
hist(x, breaks = c(-3,-1,0,0.5,3) )     # 異なる区切り幅(出力例は省略)
```

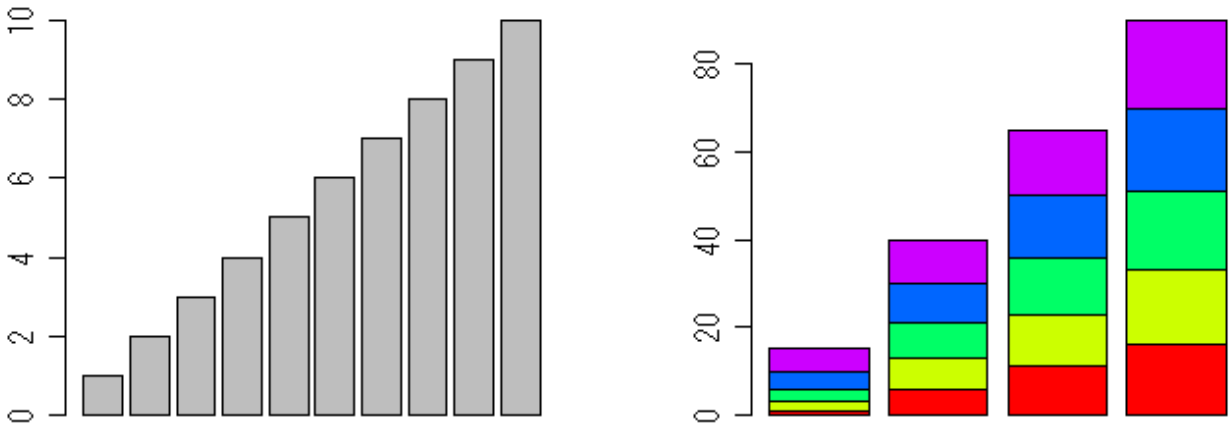




棒グラフ：barplot()

関数 barplot() で棒グラフ描くことが出来る．データにベクトルを指定すると各要素の長さについて棒グラフが描かれ，データに行列を指定すると，一本（一列）の棒に各行の要素が層別で表示される．

```
# barplot(ベクトルデータ, ...) , barplot(行列データ, ...)

 barplot(1:10)
 barplot(matrix(1:20, 5), col=rainbow(5))
```



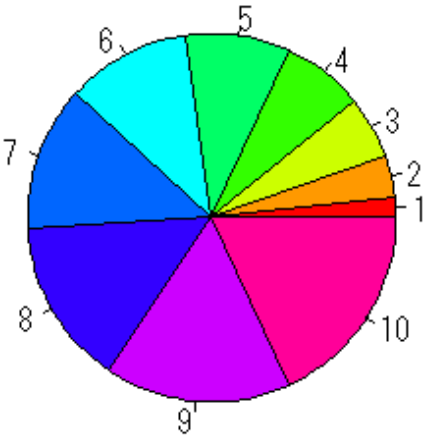
関数 barplot() のオプションは以下の通り．ところで，barplot() は各棒の中心位置の x 座標を返すので，この値と lines などの低水準作図関数を使うことで，例えば信頼区間を描くことが出来る．

引数	機能
angle=45	棒を斜線で塗る際の線分の角度を指定する．
beside=F	（行列データで層別表示する場合）T にするとグループ毎に棒が並べて表示され，F にすると一本の棒に層別で表示される．
col=rainbow(10)	棒を塗りつぶす際の色を指定する．（ベクトルでも可）
density=30	棒を斜線で塗る際の線分の密度を指定する．
horiz=F	T にすると棒が水平に表示される．
legend = rownames(x)	凡例を描く．（例はデータ名が x の場合）
names=文字型ベクトル	各棒のラベルを文字型ベクトルで指定する．
space=1	各棒の間の間隔を指定する．
width=数値型ベクトル	各棒の相対的な幅を指定する．

円グラフ：pie()

関数 pie() で円グラフ描くことが出来る．

```
# pie(データ, labels = names(x), radius = 0.8, density = NULL, angle = 45, col = NULL, ...)
 pie(1:10, r=1,  col=rainbow(10))
```



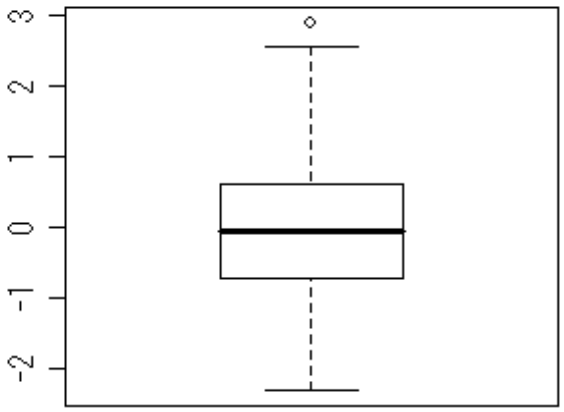
関数 pie() のオプションは以下の通り．

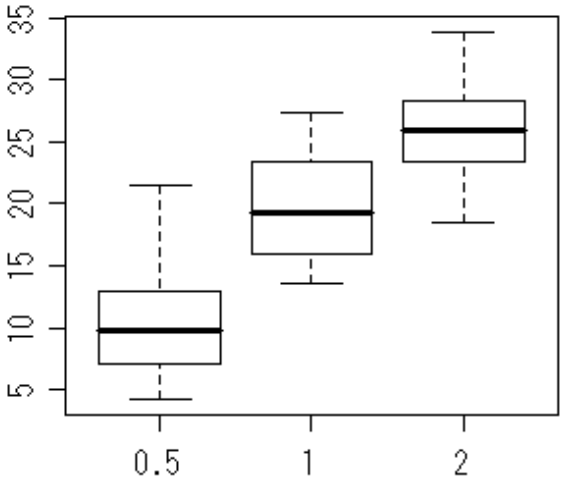
引数	機能
angle=45	棒を斜線で塗る際の線分の角度を指定する．
border=NULL	NA にすると仕切り線が描かれなくなる．
col=rainbow(10)	棒を塗りつぶす際の色を指定する．（ベクトルでも可）
density=30	棒を斜線で塗る際の線分の密度を指定する．
main="Title"	タイトルを指定する．
radius=0.8	円の半径を -1 〜 1 の範囲で指定する．負の値にすると，データを表示する開始位置が左端からとなる．

箱ひげ図：boxplot()

複数のデータの分布の違いを比較する際は，関数 boxplot() で箱ひげ図を描く．

```
# boxplot(x, range = 1.5, width = NULL, horizontal = FALSE, add = FALSE,...)
# boxplot(x ~ 因子ベクトル)

 x <- rnorm(100)
 boxplot(x)
 boxplot(len ~ dose, data = ToothGrowth)
```





関数 boxplot() のオプションは以下の通り．

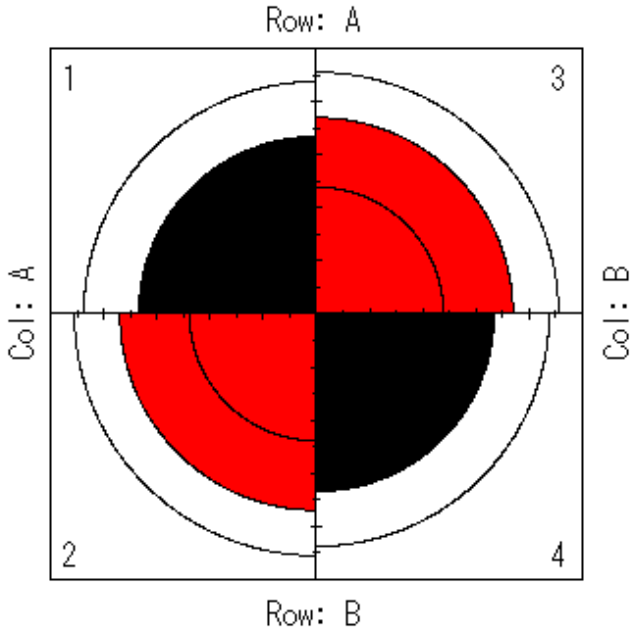
引数	機能
boxwex = 0.8	箱の幅を指定する．
horizontal=F	T にすると，箱が横向きに描かれる．
notch=F	箱に notch を入れるかどうかを指定する．
outwex = 0.5	outlier の幅を指定する．
range=1.5	「ひげ」が箱の端からどれだけ離れるかを指定する．range が正の値であるなら，「ひげ」は箱から四分位範囲の range 倍（デフォルトは 1.5 倍）である最も端にあるデータ点となる．例えば，上側の「ひげ」の場合は Q3 + 1.5×(Q3-Q1) となる．range が 0 ならば「ひげ」はデータ範囲の一番端の値になる．
staplewex = 0.5	ひげの幅を指定する．
width=数値ベクトル	箱の幅を指定する．

分割表データの図示：fourfoldplot()

関数 fourfoldplot() で 1 個以上の層について，2 変数間の関係を考慮に入れた 2×2 のグラフを生成する．

```
# fourfoldplot(2*2分割表データ)
```

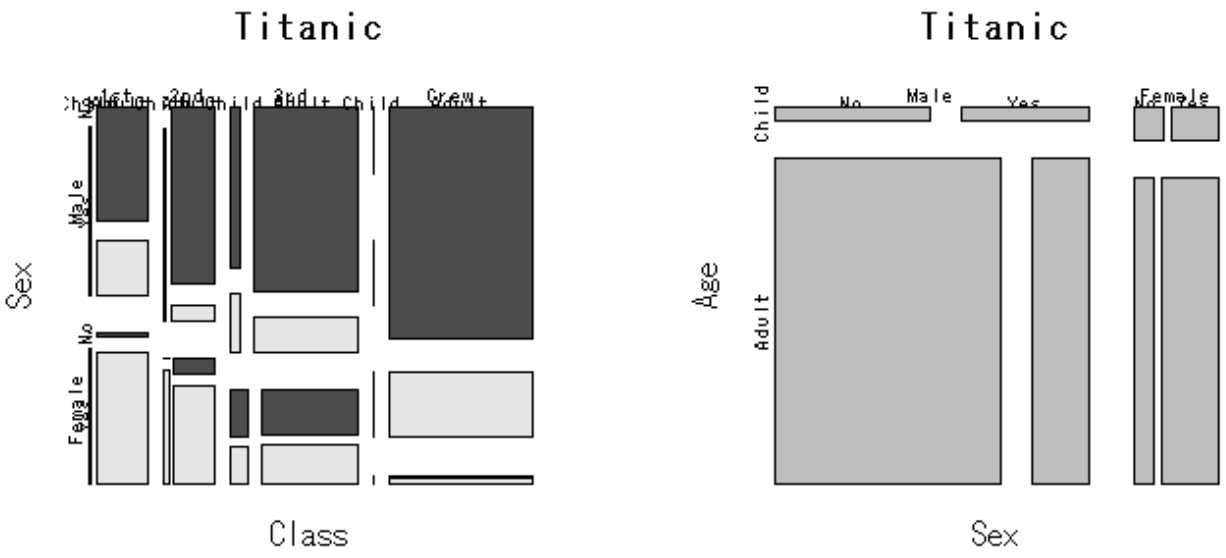
```
x <- matrix(1:4,2)
fourfoldplot(x, col=1:2)
```



分割表データの図示：mosaicplot()

関数 mosaicplot() で 分割表データをモザイクプロットとして表示する．引数 dir="h" (="v") で分割方向が変わる．

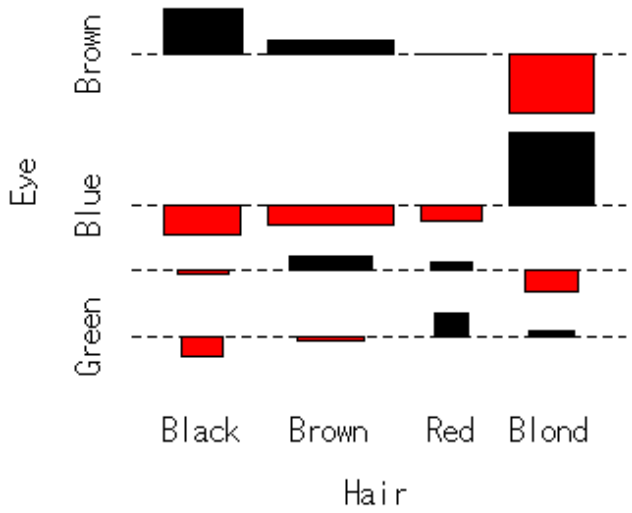
```
# mosaicplot(分割表データ)
# mosaicplot(モデル式, data=データ名)

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



分割表データの図示：assocplot()

関数 assocplot() で分割表のデータについて，Cohen-Friendly の連関プロット（Association Plots）を行う．

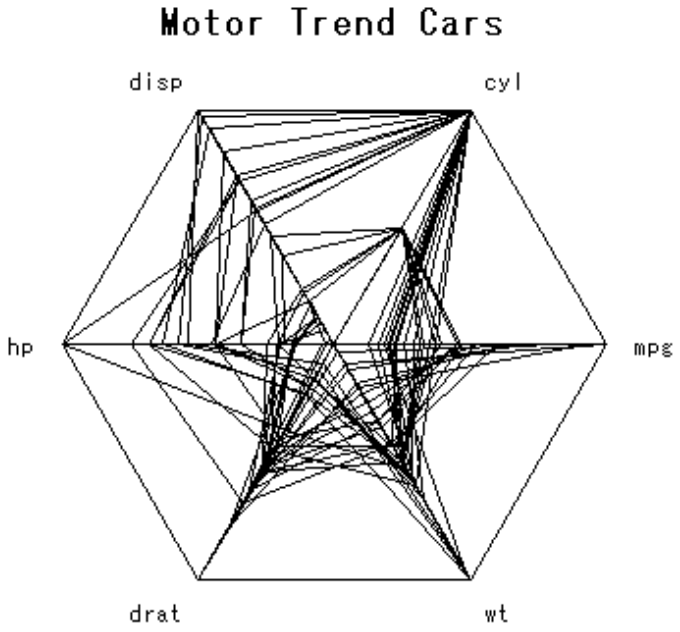
```
# assocplot(分割表データ)

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

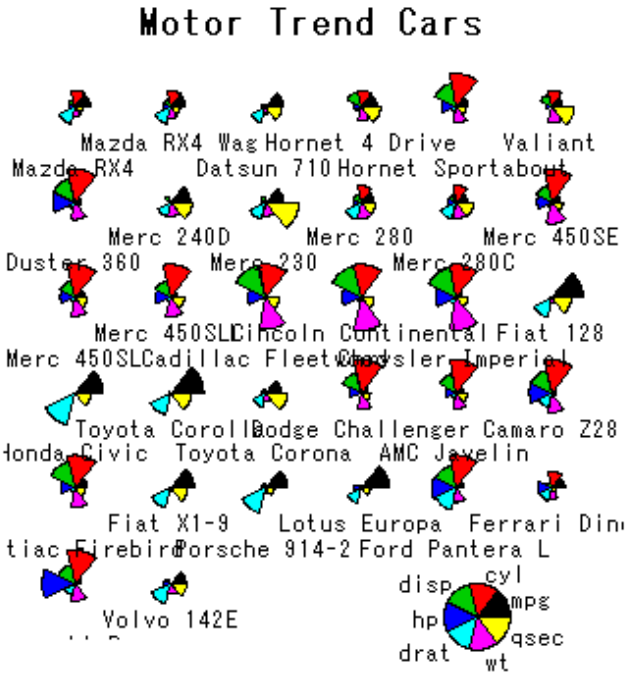


多変量データの図示：stars()

関数 stars() を用いることで，くもの巣プロットを描いて全体の傾向を見たり，星形図を描いてサンプルの分類を行うことが出来る．

```
stars(mtcars[, 1:6], locations = c(0,0), radius = T,
      key.loc=c(0,0.0), main="Motor Trend Cars")        # くもの巣プロット
stars(mtcars[, 1:7], len = 0.8, key.loc = c(12, 1.5),
      main = "Motor Trend Cars", draw.segments = TRUE)  # 星形図
```

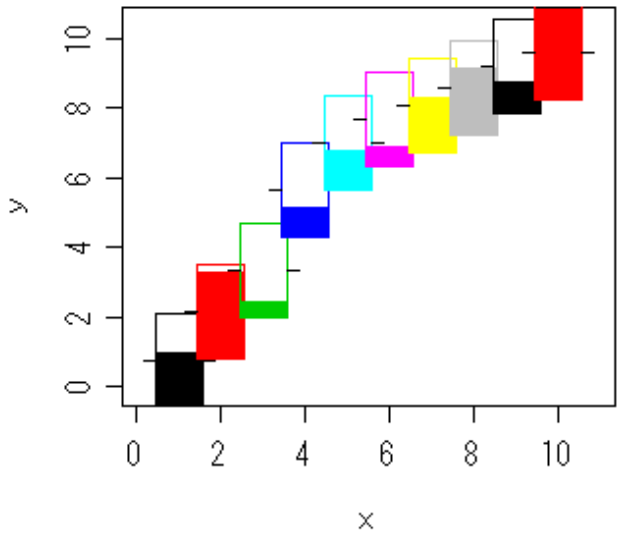




多変量データの図示：symbols()

関数 symbols() で多変量データを図示する散布図を描くことが出来る．ただし，点や線の代わりに，円や星，箱ひげ図で散布図が描かれる．

```
x <- 1:10;  y <- sort(10*runif(10));  z <- runif(10)
symbols(x, y, thermometers=cbind(0.5,1,z), inches=0.5, fg=1:10)
```



温度計の記号以外にも様々な記号で表すことが出来る．

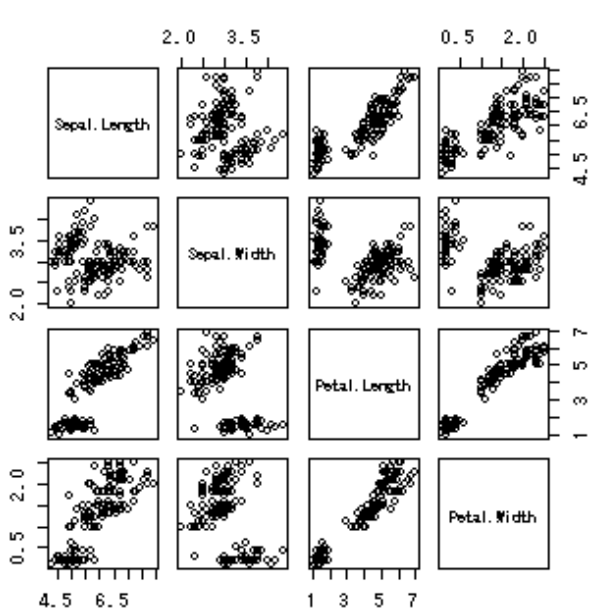
記号・機能	引数の指定方法
円	circles=数値ベクトル（半径）
正方形	squares=数値ベクトル（半径）
長方形	rectangles=数値ベクトル（半径）
星	stars=3列以上の行列（星の中央からの線の長さ，0 以上 1 以下）
温度計	thermometers=c(幅，高さ，塗りつぶす高さの比率) または thermometers=c(幅，高さ，塗りつぶす比率1，塗りつぶす比率2)
箱ひげ図	boxplots=c(幅，高さ，下ひげの長さ1，上ひげの長さ2，中央線の位置)
シンボルの色	fg=数値ベクトル
塗りつぶす色	bg=数値ベクトル
単位	inches=F ならば単位は x 軸の単位が使われる．inches=T ならば最大のシンボルが高さ1インチであるようにシンボルが指定される．inches=数値ならば最大のシンボルが（インチ単位で）指定した数値の高さになるように調節される．

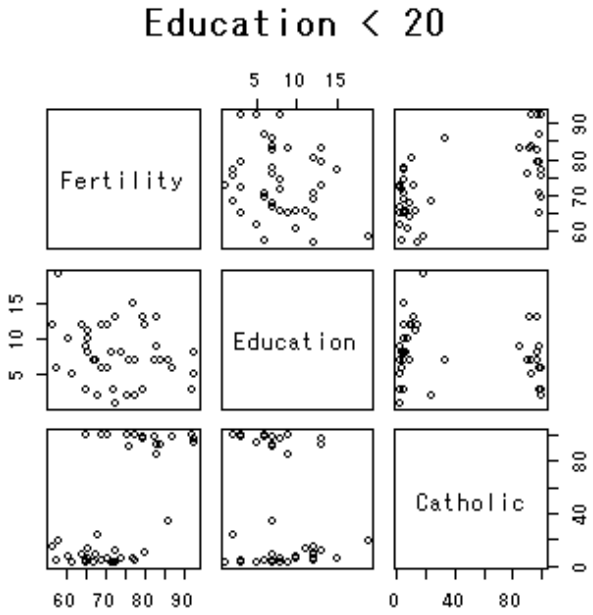
多変量データの図示：pairs()

関数 pairs(行列) で各列同士の組合せ全てについて散布図を描く．関数 pairs(モデル式) で，細かい

条件を指定した上で，各列同士の組合せ全てについて散布図を描く．

```
pairs(iris[1:4])                                              # 行列で指定

pairs(~ Fertility + Education + Catholic, data = swiss,
      subset = Education < 20, main = "Education < 20")       # モデル式で指定
```





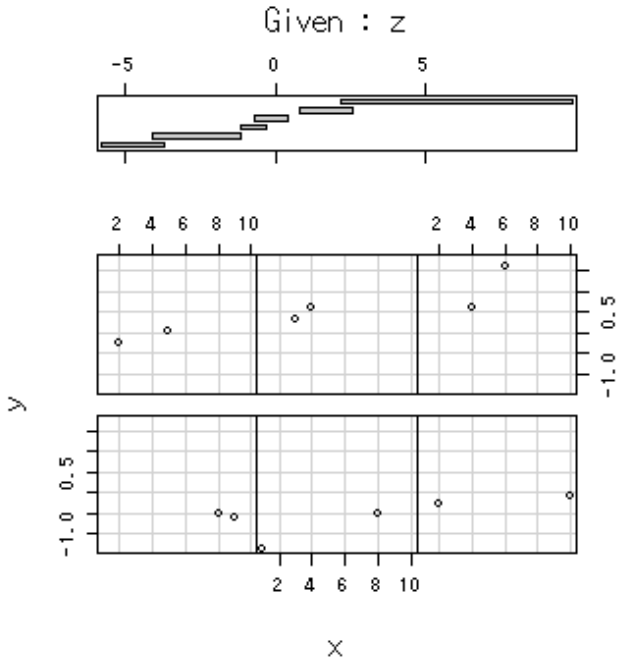
多変量データの図示：coplot()

関数 coplot() で共変量プロットを描く．この関数は数値ベクトル x，y，z (z は因子オブジェクトでも可) を引数としてとり，z の与えられた値における y に対する x の散布図を複数生成する．

coplot(y ~ x | z) ： z が因子ならば，z の水準ごとに y が x に対してプロットされる． z が数値ならば，いくつかの『条件を与える区間』に分割され，その区間に含まれる z の値に対して y が x に対してプロットされる．

coplot(y ~ x | z_1 * z_2) ： 2 変量 z_1，z_2 に条件付けされた，x に対する y の散布図を生成する．

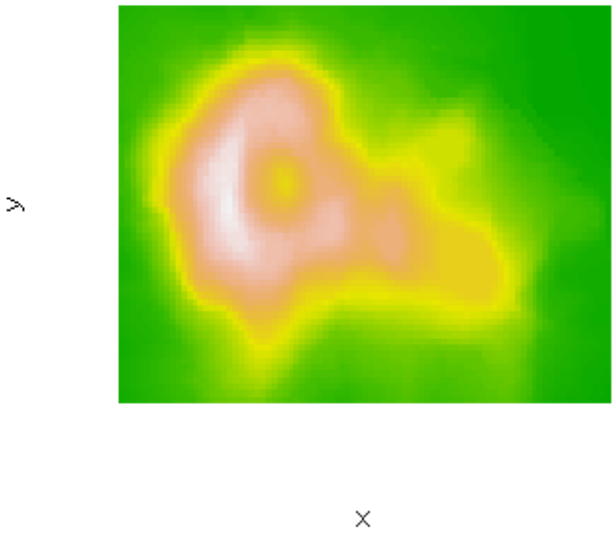
```
x <- 1:10;  y <- rnorm(10);  z <- x*y
coplot(y ~ x | z)
```



3次元データの図示：image()

関数 image() で 3 次元データをカラーイメージで図示する.

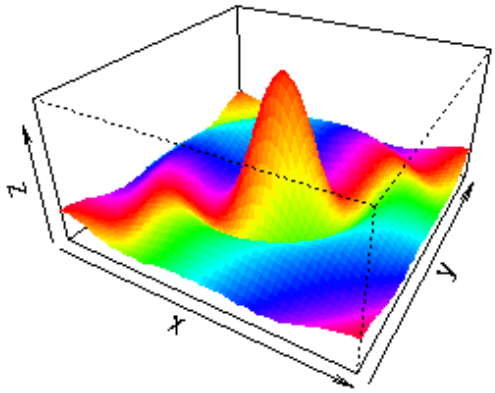
```
x <- 10*(1:nrow(volcano));   y <- 10*(1:ncol(volcano))
image(x, y, volcano, col = terrain.colors(100), axes = FALSE)
```



3次元データの図示：persp()

関数 persp() で 3 次元立体図を描く．

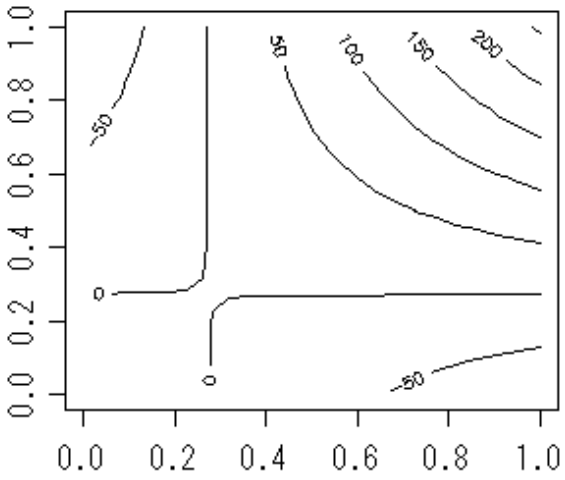
```
x <- seq(-10, 10, length= 50);   y <- x
f <- function(x, y) { r <- sqrt(x^2+y^2); 10 * sin(r)/r }
z <- outer(x, y, f);
persp(x, y, z, theta = 30, phi = 30, expand = 0.5, col = rainbow(50), border=NA)
```



3次元データの図示：contour()

関数 contour() で 3 次元データを等高線図で図示する．関数 filled.contour() も同様の関数である．

```
x  <- -6:16
contour(outer(x, x), method = "edge", vfont = c("sans serif", "plain"))
```

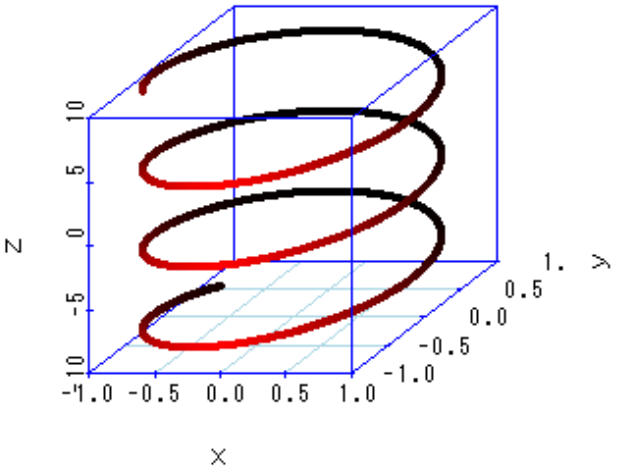


3次元データの図示：scatterplot3d()

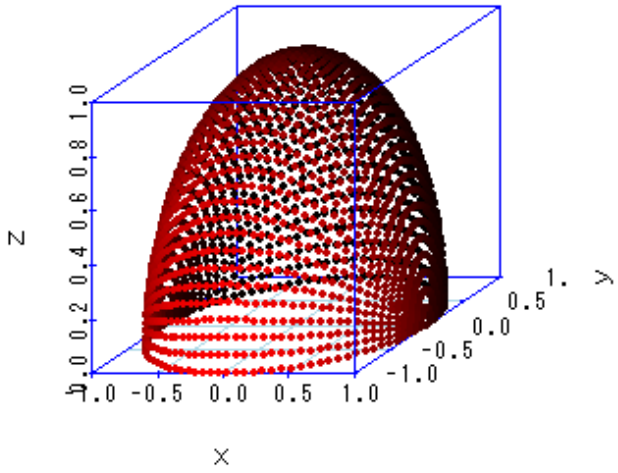
パッケージ scatterplot3d には 3 次元的に表示する散布図を描く関数 scatterplot3d() が入っている．まず，パッケージ scatterplot3d を呼び出す必要がある．

```
library(scatterplot3d)
z <- seq(-10, 10, 0.01);  x <- cos(z);  y <- sin(z)
scatterplot3d(x, y, z, highlight.3d=TRUE, col.axis="blue",
              col.grid="lightblue", main="Plot-1", pch=20)  # 線プロット

temp <- seq(-pi, 0, length = 50)
x <- c(rep(1, 50) %*% t(cos(temp)))
y <- c(cos(temp) %*% t(sin(temp)))
z <- c(sin(temp) %*% t(sin(temp)))
scatterplot3d(x, y, z, highlight.3d=TRUE, col.axis="blue",
              col.grid="lightblue", main="Plot-2", pch=20)  # 点プロット
```

Plot-1


Z
10
5
0
-5
-10
-1.0 -0.5 0.0 0.5 1.0
X
1.
0.5
0.0
-0.5
-1.0
y

Plot-2


Z
1.0
0.8
0.6
0.4
0.2
0.0
-1.0 -0.5 0.0 0.5 1.0
X
1.
0.5
0.0
-0.5
-1.0
y