R graph gallery

Compilation by Eric Lecoutre

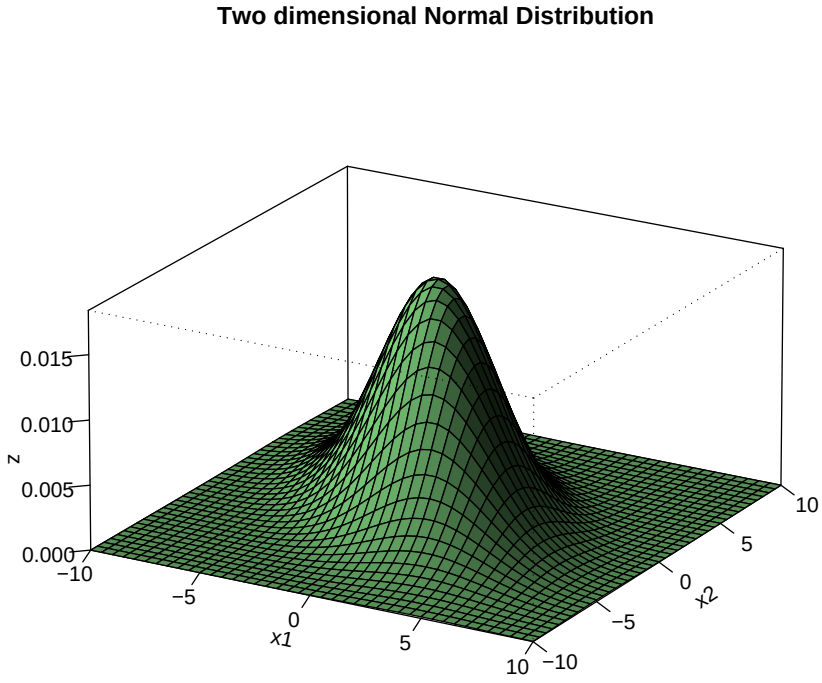
December 12, 2003

Contents

3D Bivariate Normal Density

variables: 2 QT
library: -
function: persp
submitted by: Bernhard Pfaff <Bernhard.Pfaff@drkw.com>

Sample graph



$$f(\mathbf{x}) = \frac{1}{2\,\pi\sqrt{\sigma_{11}\,\sigma_{22}\,(1-\rho^2)}} \cdot \exp\left\{-\frac{1}{2(1-\rho^2)}\left[\frac{(x_1-\mu_1)^2}{\sigma_{11}} - 2\,\rho\,\frac{x_1-\mu_1}{\sqrt{\sigma_{11}}}\,\frac{x_2-\mu_2}{\sqrt{\sigma_{22}}} + \frac{(x_2-\mu_2)^2}{\sigma_{22}}\right]\right\}$$

Sample code

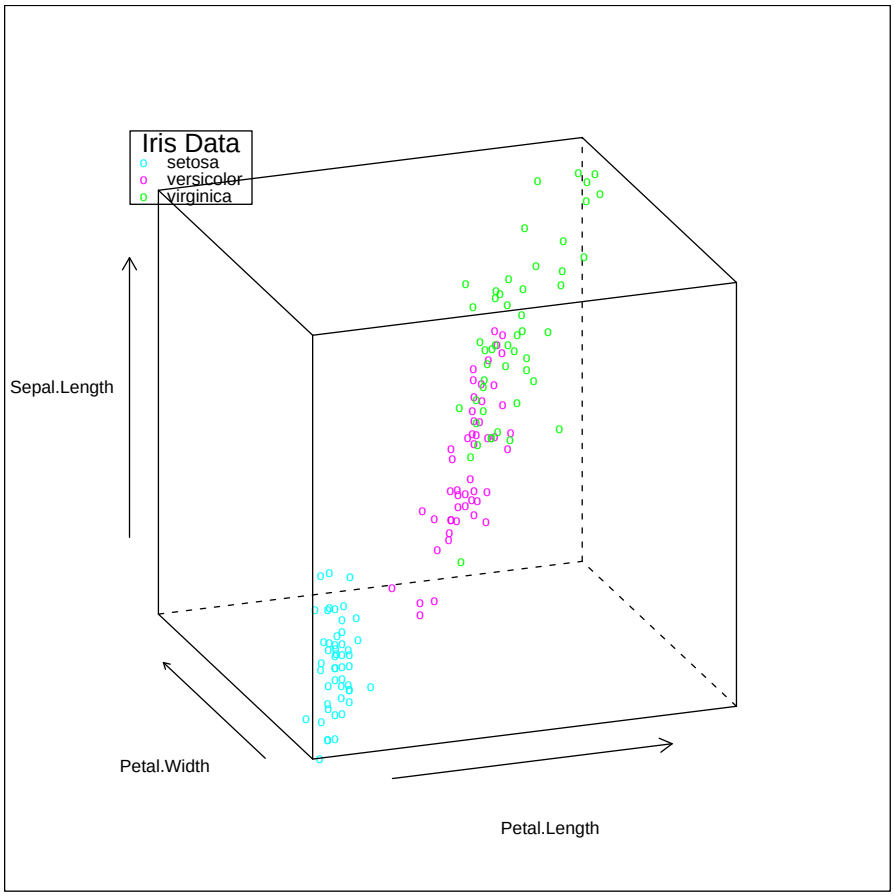
```
# 3-D plots
#
#
mu1<-0 # setting the expected value of x1
mu2<-0 # setting the expected value of x2
s11<-10 # setting the variance of x1
s12<-15 # setting the covariance between x1 and x2
s22<-10 # setting the variance of x2
rho<-0.5 # setting the correlation coefficient between x1 and x2
x1<-seq(-10,10,length=41) # generating the vector series x1
x2<-x1 # copying x1 to x2
#
f<-function(x1,x2)
{
term1<-1/(2*pi*sqrt(s11*s22*(1-rho^2)))
term2<--1/(2*(1-rho^2))
term3<-(x1-mu1)^2/s11
term4<-(x2-mu2)^2/s22
term5<--2*rho*((x1-mu1)*(x2-mu2))/(sqrt(s11)*sqrt(s22))
```

```
term1*exp(term2*(term3+term4-term5))
} # setting up the function of the multivariate normal density
#
z<-outer(x1,x2,f) # calculating the density values
#
persp(x1, x2, z,
main="Two dimensional Normal Distribution",
sub=expression(italic(f)~(bold(x))==frac(1,2~pi~sqrt(sigma[11]~
sigma[22]~(1-rho^2)))~phantom(0)^bold(.)~exp~bgroup("{",
list(-frac(1,2(1-rho^2)),
bgroup("[", frac((x[1]~-~mu[1])^2, sigma[11])~-~2~rho~frac(x[1]~-~mu[1],
sqrt(sigma[11]))~ frac(x[2]~-~mu[2],sqrt(sigma[22]))~+~
frac((x[2]~-~mu[2])^2, sigma[22]),"]")),"}")),
col="lightgreen",
theta=30, phi=20,
r=50,
d=0.1,
expand=0.5,
ltheta=90, lphi=180,
shade=0.75,
ticktype="detailed",
nticks=5) # produces the 3-D plot
#
mtext(expression(list(mu[1]==0,mu[2]==0,sigma[11]==10,sigma[22]==10,sigma[12
]==15,rho==0.5)), side=3) # adding a text line to the graph
```

3D Scatterplot - Cloud, regression plan

variables: 3 QT
library: lattice or scatterplot3d
function: `cloud` `scatterplot3d`,

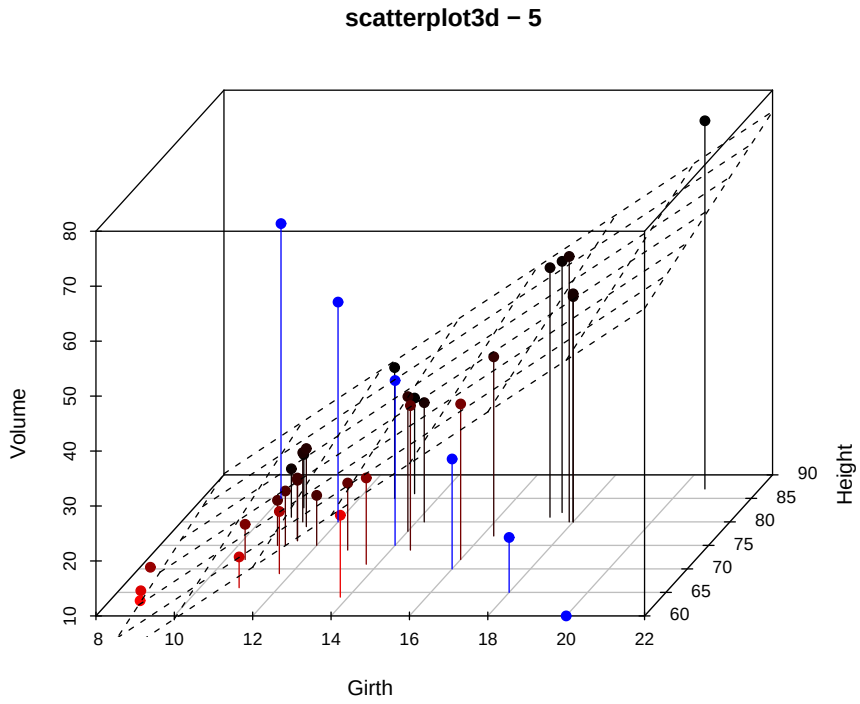
Sample graph



Sample code

```
data(iris)
cloud(Sepal.Length ~ Petal.Length * Petal.Width, data = iris,
  groups = Species, screen = list(z = 20, x = -70),
  perspective = FALSE,
  key = list(title = "Iris Data", x = .15, y=.85, corner = c(0,1),
  border = TRUE,
  points = Rows(trellis.par.get("superpose.symbol"), 1:3),
  text = list(levels(iris$Species))))
  %$%
```

Sample graph



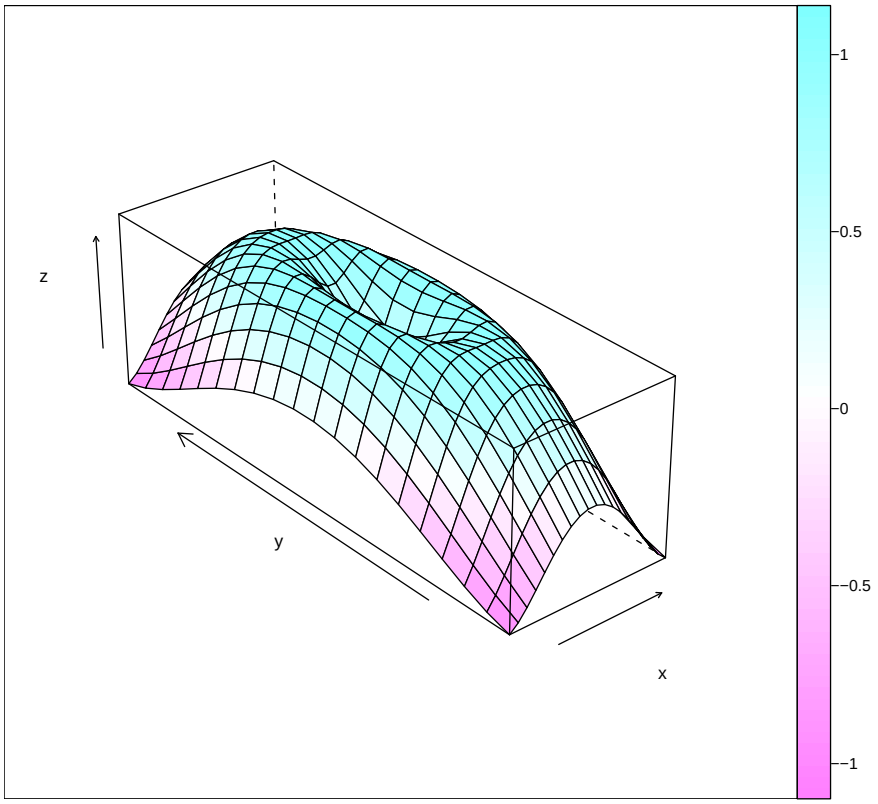
Sample code

```
data(trees)
        s3d <- scatterplot3d(trees, type="h", highlight.3d=TRUE,
            angle=55, scale.y=0.7, pch=16, main="scatterplot3d - 5")
        # Now adding some points to the "scatterplot3d"
        s3d$points3d(seq(10,20,2), seq(85,60,-5), seq(60,10,-10),
            col="blue", type="h", pch=16)
        # Now adding a regression plane to the "scatterplot3d"
        attach(trees)
        my.lm <- lm(Volume ~ Girth + Height)
        s3d$plane3d(my.lm)
```

3D Wireframe - For surfaces

variables: 3 QT
library: lattice
function: `wireframe`

Sample graph



Sample code

```
 x <- seq(-pi, pi, len = 20)
     y <- seq(-pi, pi, len = 20)
     g <- expand.grid(x = x, y = y)
     g$z <- sin(sqrt(g$x^2 + g$y^2))
     wireframe(z ~ x * y, g, drape = TRUE,
               aspect = c(3,1), colorkey = TRUE)
%$%
```

Agreement plot

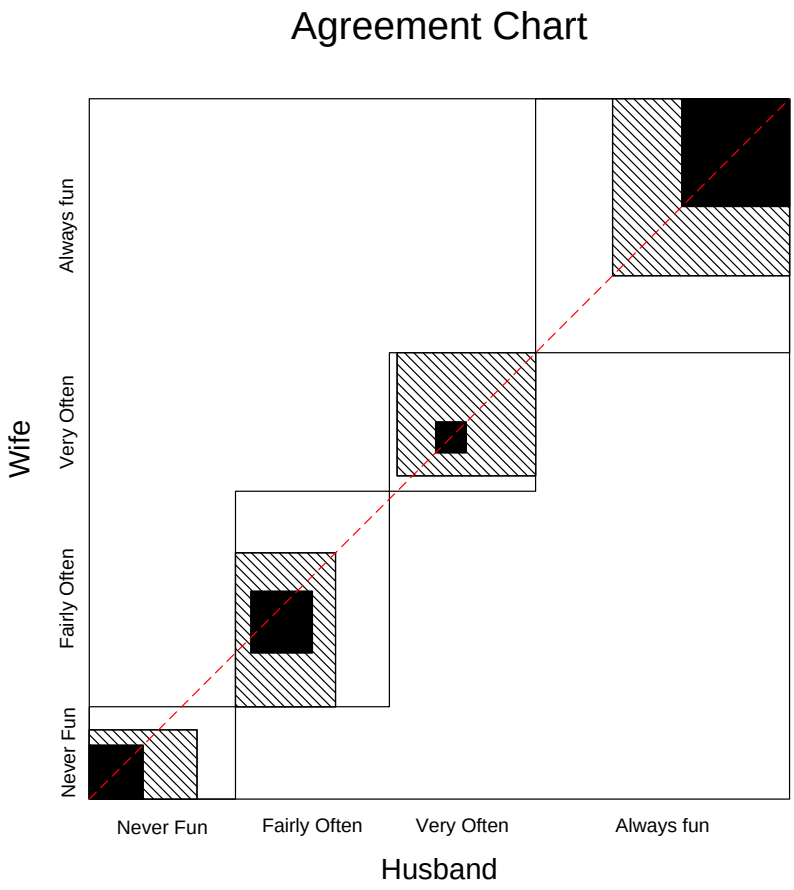
variables: one confusion matrix
library: vcd
function: `agreementplot`

Description

Representation of a $k * k$ confusion matrix, where the observed and expected diagonal elements are represented by superposed black and white rectangles, respectively.

Agreement chart allows to quickly see where two judges do disagree.

Sample graph



Sample code

```
library(vcd)
```

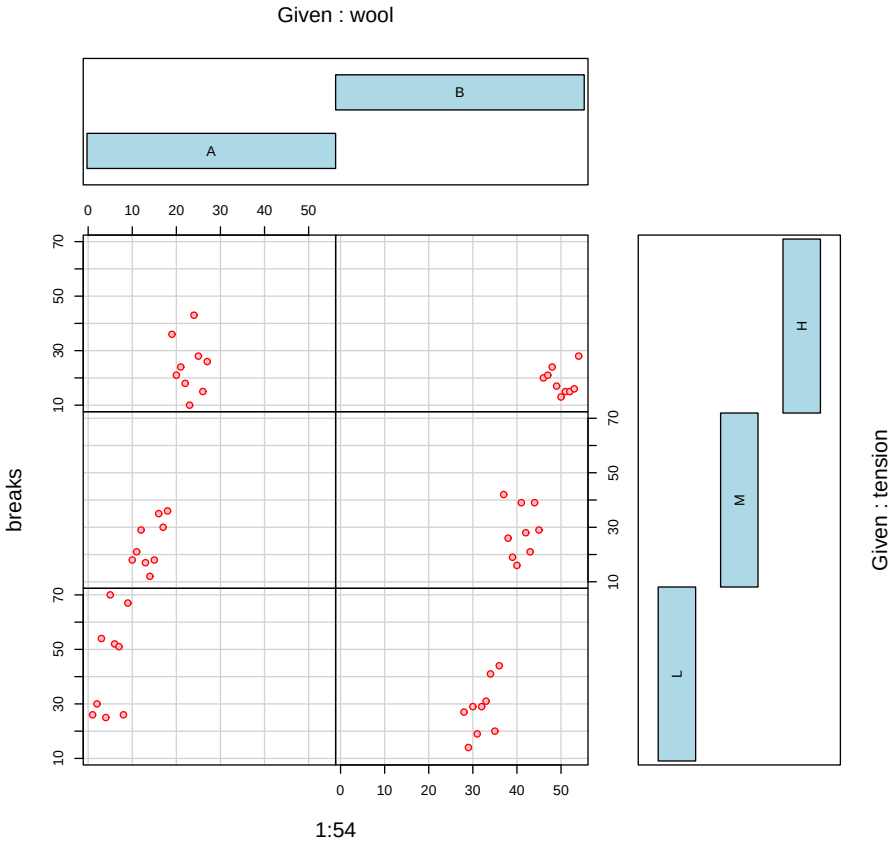
```
data(SexualFun)
agreementplot(t(SexualFun))
```

Conditional Plot 1 - coplot

variables: ≥ 3 QL or QT
library: -
function: `coplot`

Description

Sample graph



Sample code

```
data(warpbreaks)
     ## given two factors
coplot(breaks ~ 1:54 | wool * tension, data = warpbreaks,
            col = "red", bg = "pink", pch = 21,
            bar.bg = c(fac = "light blue"))
```

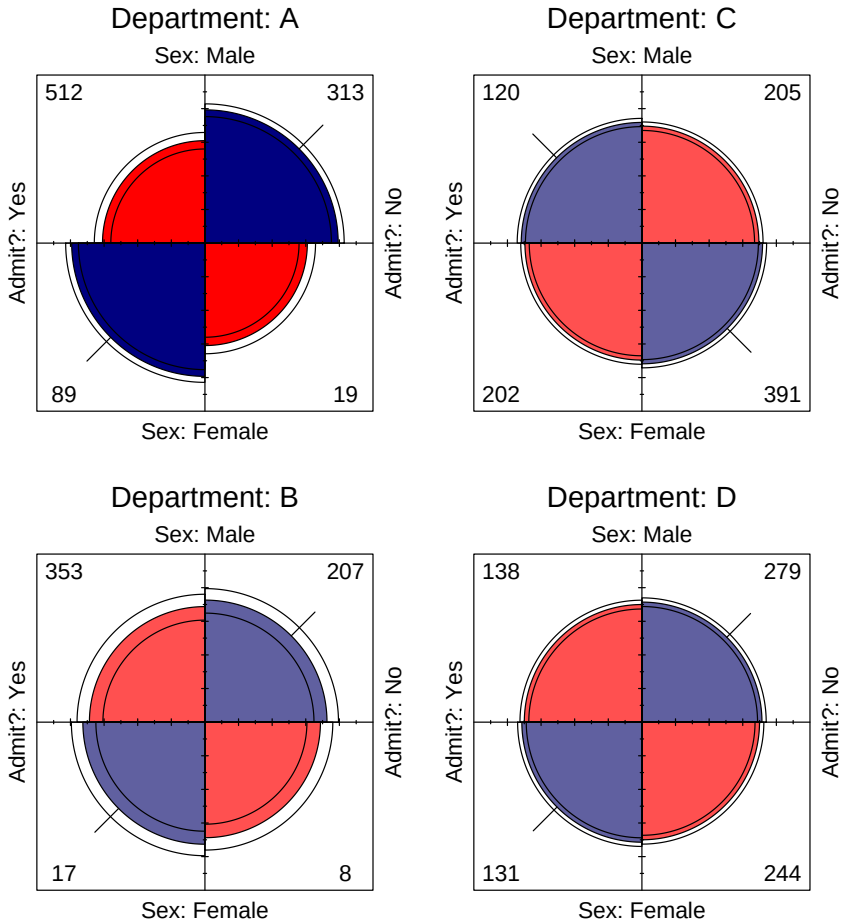
Fourfold Display

data: 2x2xk contingency tables
library: vcd
function: `fourfoldplot`

Description

The fourfold display depicts frequencies by quarter circles, whose radius is proportional to $\sqrt{n_{ij}}$, so the area is proportional to the cell count . The cell frequencies are usually scaled to equate the marginal totals, and so that the ratio of diagonally opposite segments depicts the odds ratio. Confidence rings for the observed odd ratio allow a visual test of the hypothesis $H_0 : \theta = 1$ corresponding to no association. They have the property that the rings for adjacent quadrants overlap iff the observed counts are consistent with the null hypothesis.

Sample graph



Sample code

```
library("vcd")
# Load data
 data(UCBAdmissions)
 x <- aperm(UCBAdmissions, c(2, 1, 3))
 dimnames(x)[[2]] <- c("Yes", "No")
 names(dimnames(x)) <- c("Sex", "Admit?", "Department")
# Shows for 4 departments
fourfoldplot(x[,,1:4])
```

Hexagon Binning Matrix

variables: 2 QL x 2 QT
sample size: large datasets (fast also for $n \geq 10^6$)
library: hexbin (bioconductor)
function: `plot.hexbin`, `hmatplot`

Description

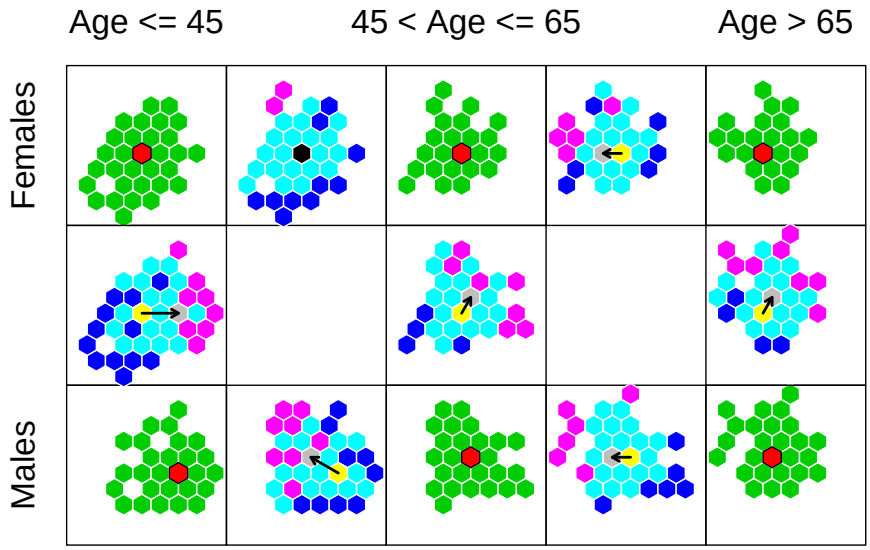
Hexagon binning is a form of bivariate histogram useful for visualizing the structure in datasets with large n. The underlying concept of hexagon binning is extremely simple;

1. the xy plane over the set (range(x), range(y)) is tessellated by a regular grid of hexagons.
2. the counts of points falling in each hexagon are counted and stored in a data structure
3. the hexagons with count > 0 are plotted using a color ramp or varying the radius of the hexagon in proportion to the counts.

The algorithm is extremely fast and effective for displaying the structure of datasets with $n \geq 10^6$. If the size of the grid and the cuts in the color ramp are chosen in a clever fashion than the structure inherent in the data should emerge in the binned plots. The same caveats apply to hexagon binning as apply to histograms and care should be exercised in choosing the binning parameters.

The hexbin library is a set of function for creating and plotting hexagon bins. The library extends the basic hexagon binning ideas with several functions for doing bivariate smoothing, finding an approximate bivariate median, and looking at the difference between two sets of bins on the same scale. The basic functions can be incorporated into many types of plots.

Sample graph



Sample code

```
library("hexbin")

data(NHANES)# pretty large data set!
good <- !(is.na(NHANES$Albumin) | is.na(NHANES$Transferin))
NH.vars <- NHANES[good, c("Age","Sex","Albumin","Transferin")]

# extract dependent variables and find  ranges for global binning
x <- NH.vars[,"Albumin"]
rx <- range(x)
y <- NH.vars[,"Transferin"]
```

```
ry <- range(y)

age <- cut(NH.vars$Age,c(1,45,65,200))
sex <- NH.vars$Sex
subs <- tapply(age,list(age,sex))
bivariate bins for each factor combination

for (i in 1:length(unique(subs))) {
   good <- subs==i
   assign(paste("nam",i,sep=""),
   erode.hexbin(hexbin(x[good],y[good],xbins=23,xbnds=rx,ybnds=ry)))
     }

nam <- matrix(paste("nam",1:6,sep=""),ncol=3,byrow=TRUE)
rlabels <-c("Females","Males")
clabels  <- c("Age <= 45","45 < Age <= 65","Age > 65")
zoom <- hmatplot(nam,rlabels,clabels,border=list(hbox=c("black","white"),
  hdiff=rep("white",6)))
```

Mosaicplot - Associations in a contingency table

variables: QL (≥ 2)
library: -
function: `mosaicplot`

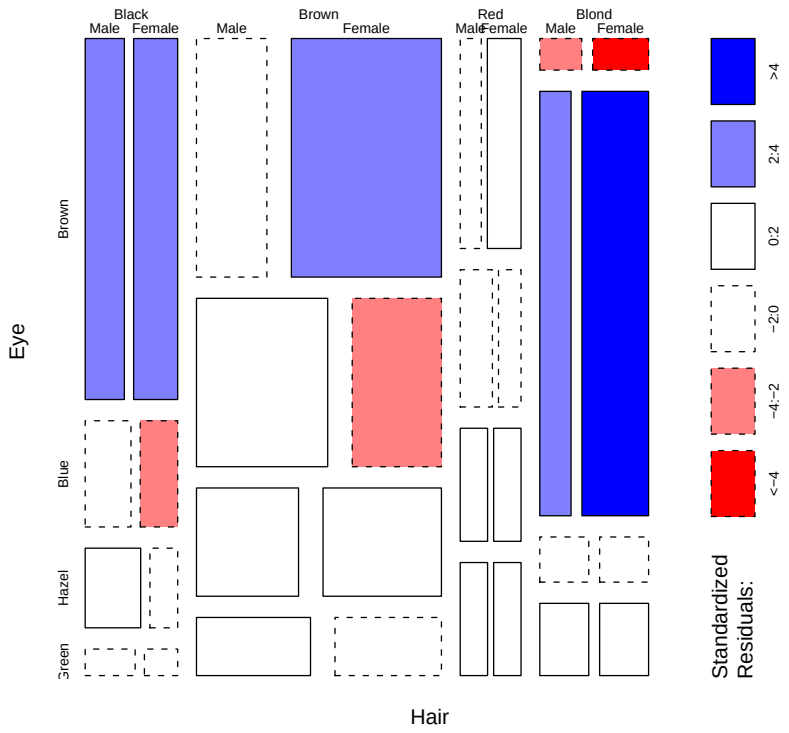
Description

Mosaicplot graph represents a contingency table, each cell corresponding to a piece of the plot, which size is proportional to cell entry.

Extended mosaic displays show the standardized residuals of a loglinear model of the counts from by the color and outline of the mosaic's tiles. (Standardized residuals are often referred to a standard normal distribution.) Negative residuals are drawn in shaded of red and with broken outlines; positive ones are drawn in blue with solid outlines.

Thus, mosaicplot are perfect to visualize associations within a table and to detect cells which create dependancies.

Sample graph



Sample code

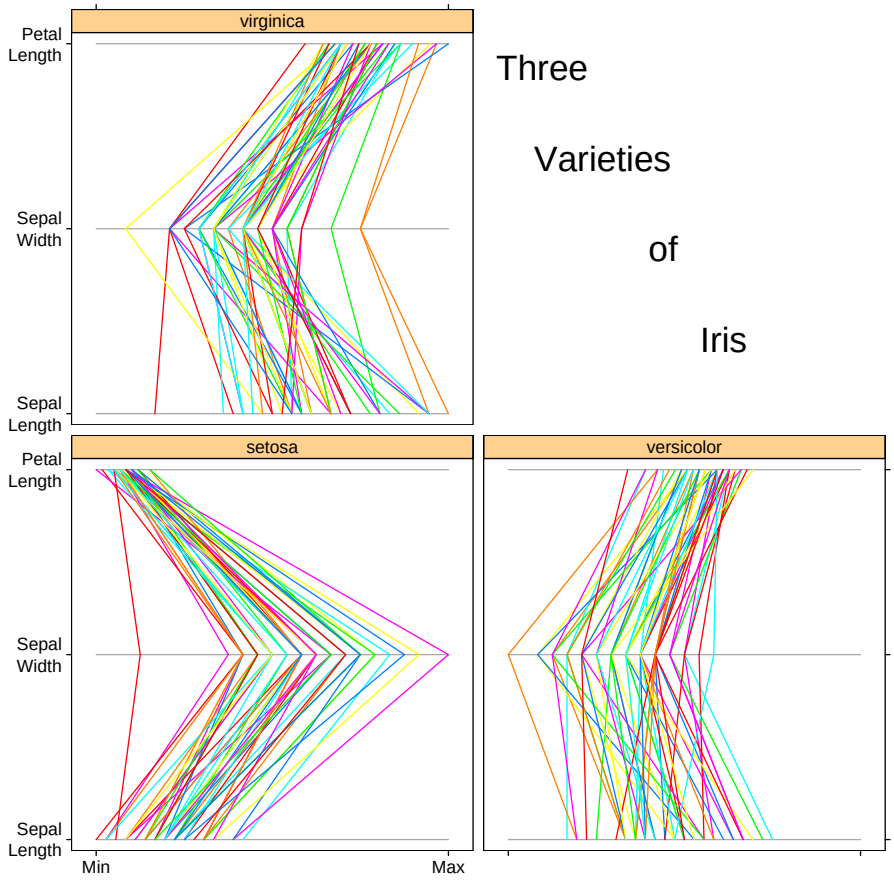
```
data(HairEyeColor)
mosaicplot(HairEyeColor, shade = TRUE)
```

Parallel Plot - Comparing groups with few subjects

variables: 1 QL and at least 2 other variables
library: lattice or MASS
function: `parallel`

Description

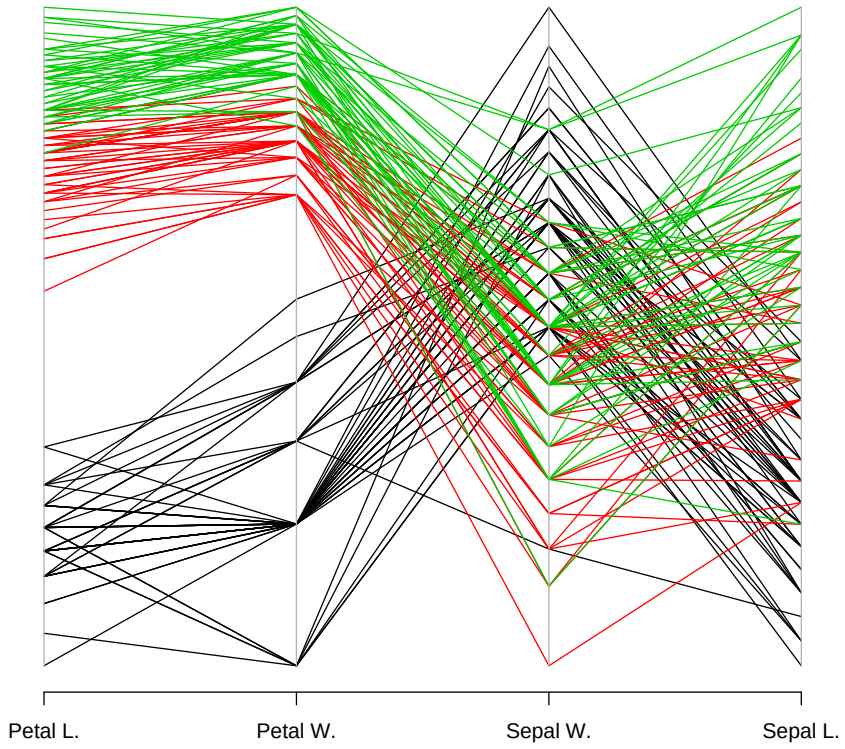
Sample graph



Sample code

```
data(iris)
parallel(~iris[1:3]|Species, data = iris,
          layout=c(2,2), pscales = 0,
          varnames = c("Sepal\nLength", "Sepal\nWidth", "Petal\nLength"),
          page = function(...) {
              grid.text(x = seq(.6, .8, len = 4),
                        y = seq(.9, .6, len = 4),
                        label = c("Three", "Varieties", "of", "Iris"),
                        gp = gpar(fontsize=20))
          })
```

Sample graph



Sample code

```
data(iris3)
   ir <- rbind(iris3[,,1], iris3[,,2], iris3[,,3])
   parcoord(log(ir)[, c(3, 4, 2, 1)], col = 1 + (0:149)%/%50)
```

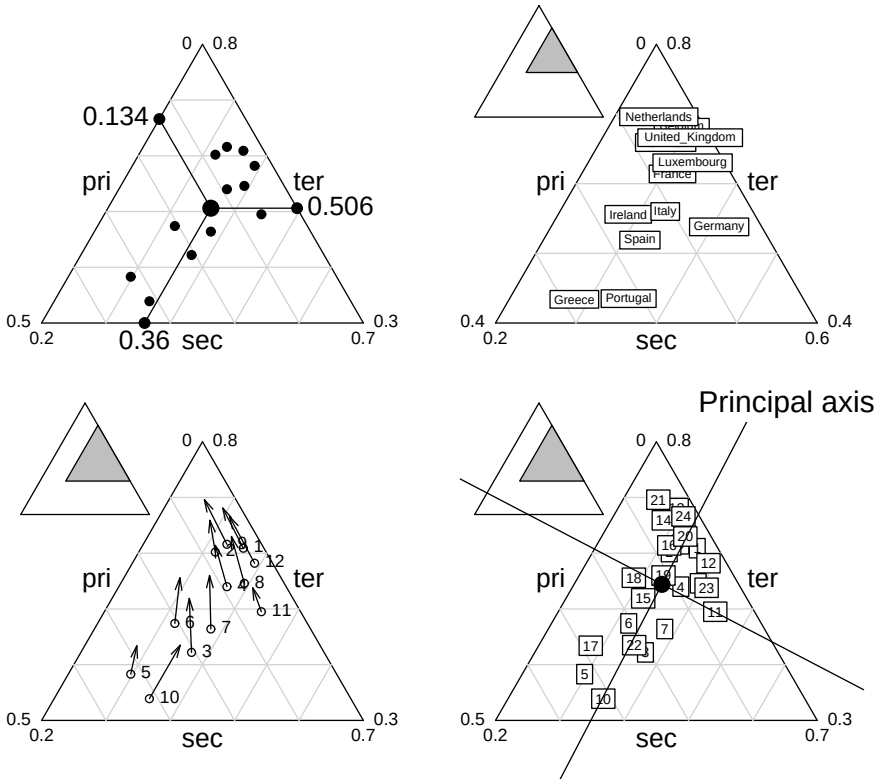
Ternary Plot - Biplot

variables: 3 QT and 1 QL
library: ade4
function: `triangle.plot`

Description

Graphs for a dataframe with 3 columns of positive or null values 'triangle.plot' is a scatterplot 'triangle.biplot' is a paired scatterplots

Sample graph



Sample code

```
  data (euro123)

par(mfrow = c(2,2))
    triangle.plot(euro123$in78, clab = 0, cpoi = 2, addmean = TRUE,
        show = FALSE)
    triangle.plot(euro123$in86, label = row.names(euro123$in78), clab = 0.8)
    triangle.biplot(euro123$in78, euro123$in86)
    triangle.plot(rbind.data.frame(euro123$in78, euro123$in86), clab = 1,
        addaxes = TRUE, sub = "Principal axis", csub = 2, possub = "topright")
    par(mfrow = c(1,1))
```

Tukey's Hanging Rootogram

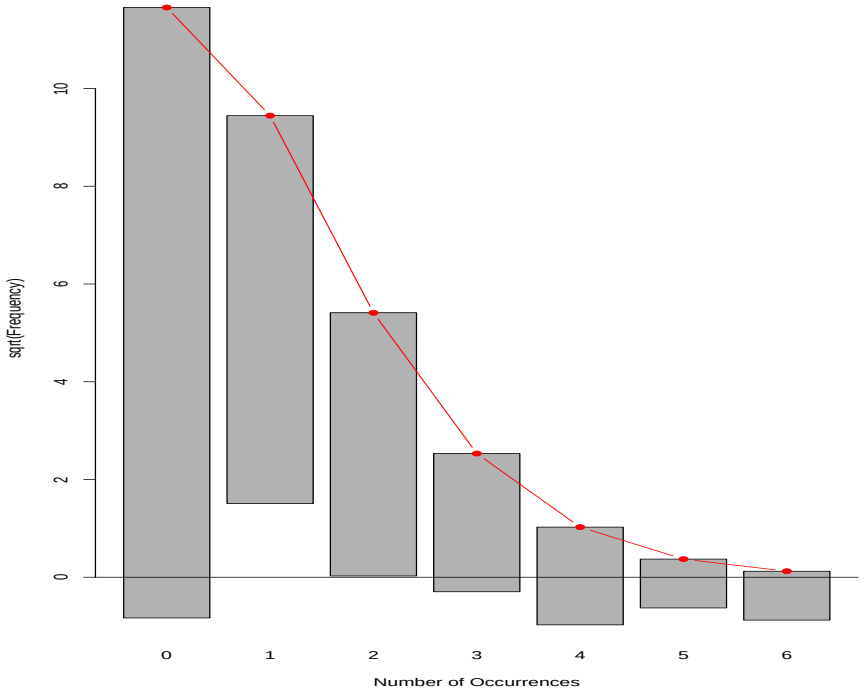
variables: 1 QL
library: vcd
function: `rootogram`

Description

Discrete frequency distributions are often graphed as histograms, with a theoretical fitted distribution superimposed. It is hard to compare the observed and fitted frequencies visually, because (a) we must assess deviations against a curvilinear relation, and (b) the largest frequencies dominate the display.

The hanging rootogram (Tukey, 1977) solves these problems by (a) shifting the histogram bars to coincide with the fitted curve, so that deviations may be judged by deviations from a horizontal line, and (b) plotting on a square-root scale, so that smaller frequencies are emphasized. Featured example shows more clearly that the observed frequencies differ systematically from those predicted under a Poisson model.

Sample graph



Sample code

```
library("vcd")

# create data
madison=table(rep(0:6,c(156,63,29,8,4,1,1)))
# fit a poisson distribution
madisonPoisson=goodfit(madison,"poisson")
rootogram(madisonPoisson)
```

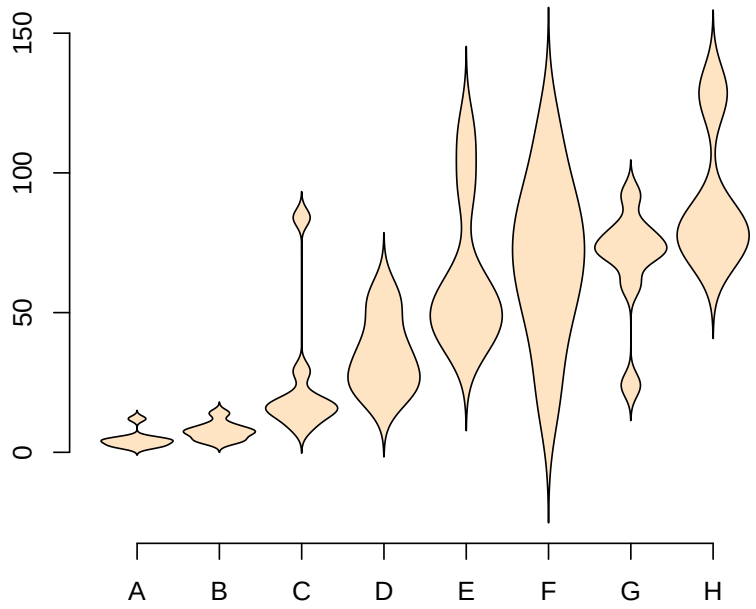
Violin Plot - Boxplot showing density, aka vase boxplot

variables: 1 QT and optional QL for groups
library: simpleR - Using R for Introductory Statistics
author: John Verzani, College of Staten Island - http://www.math.csi.cuny.edu/ verzani/
featured in: Simple R
function: `simple.violinplot`

Description

Violin plot are similar to boxplot except that they show the density of the data, estimated by kernel method.

Sample graph



Sample code

```
# library(Simple)  - required library which comes with Simple R
# http://www.math.csi.cuny.edu/Statistics/R/simpleR
data(OrchardSprays)
simple.violinplot(decrease ~ treatment, data = OrchardSprays,col="bisque",border="black")
```