Computational Statistics

An Introduction to 

Günther Sawitzki

CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

Computational Statistics
G. Sawitzki

Examples from

Computational Statistics

An Introduction to 

Günther Sawitzki
StatLab Heidelberg

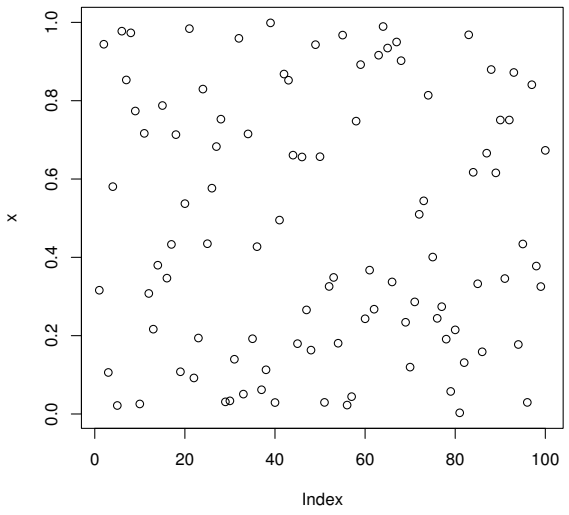
Chapman & Hall/CRC Press, Boca Raton (FL), 2009.
ISBN: 978-1-4200-8678-2.

`http://sintro.r-forge.r-project.org/`

***Example* 1.1: A Simple Plot**

Input

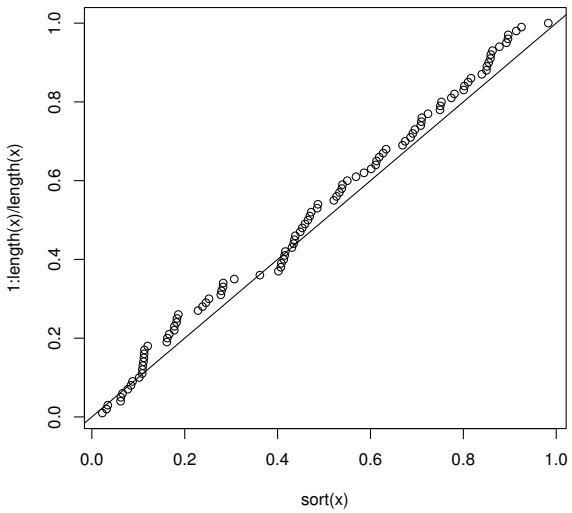
```
x <- runif(100)
plot(x)
```



***Example* 1.5: Empirical Distribution Function**

Input

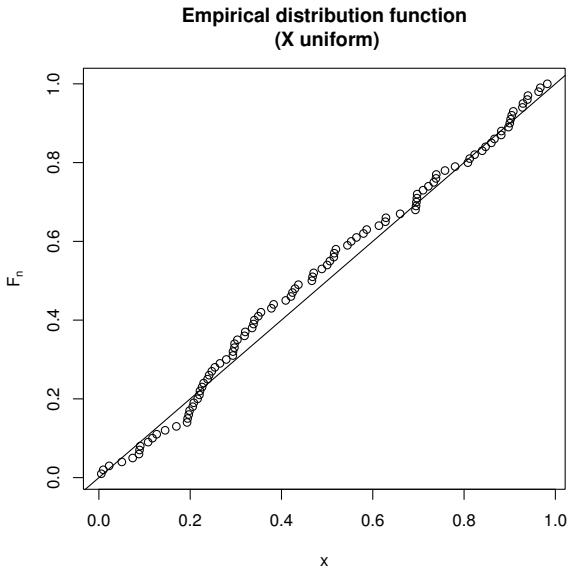
```
x <- runif(100)
plot(sort(x), 1:length(x)/length(x))
abline(0, 1)
```



***Example* 1.6: Empirical Distribution Function (Annotated Plot)**

Input

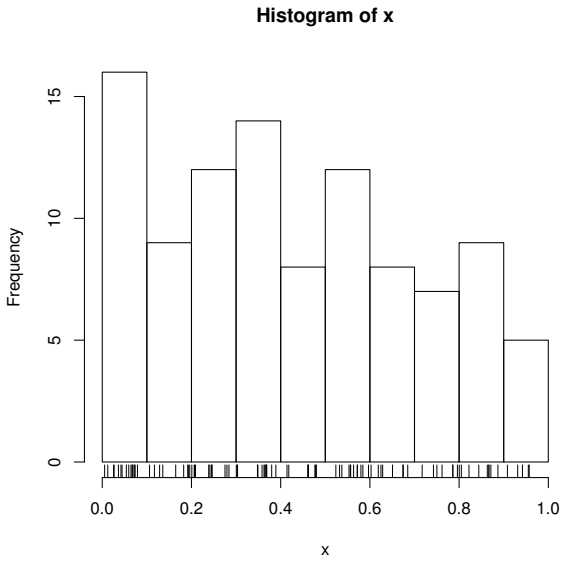
```
x <- runif(100)
plot(sort(x), (1:length(x))/length(x),
    xlab = "x", ylab = expression(F[n]),
    main = "Empirical distribution function\n (X uniform)"
)
abline(0, 1)
```



***Example* 1.7: Histogram with Rug**

Input

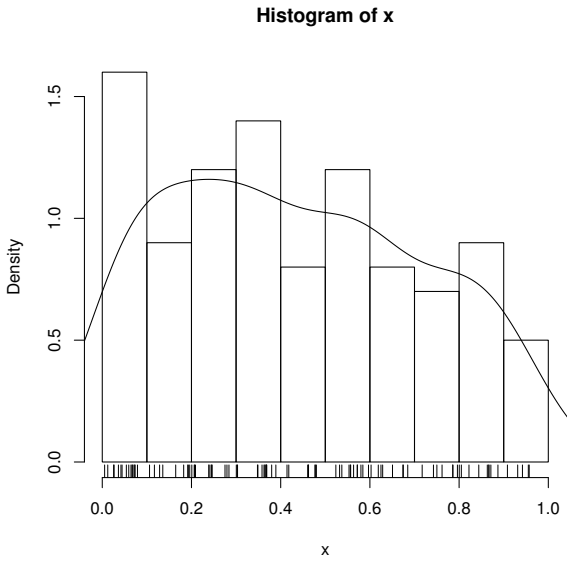
```
x <- runif(100)
hist(x)
rug(x)
```



***Example* 1.8: Augmented Histogram**

Input

```
hist(x, probability = TRUE)
rug(x)
lines(density(x))
```



***Example* 1.9: Histogram Data Structure**

Input

```
x <- runif(100)
whist <- hist(x)
whist
```

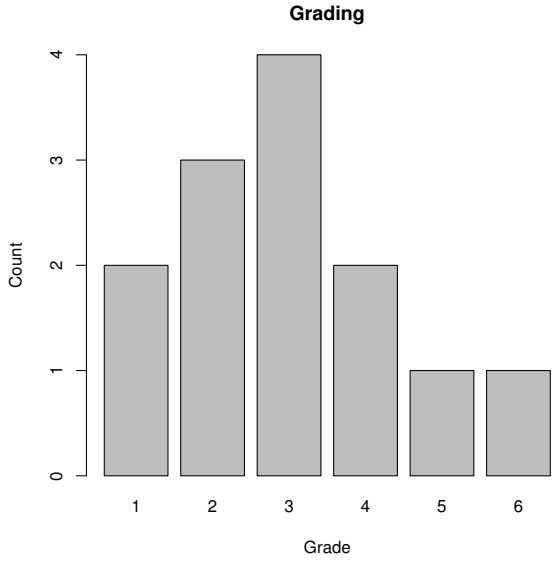
Output

```
$breaks
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

$counts
 [1] 11 13 13 12  8 14  6  6  9  8

$intensities
 [1] 1.100000 1.300000 1.300000 1.200000 0.800000 1.400000 0.600000 0.600000
 [9] 0.900000 0.800000

$density
 [1] 1.100000 1.300000 1.300000 1.200000 0.800000 1.400000 0.600000 0.600000
 [9] 0.900000 0.800000

$mids
 [1] 0.05 0.15 0.25 0.35 0.45 0.55 0.65 0.75 0.85 0.95

$xname
[1] "x"

$equidist
[1] TRUE

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

***Example* 1.10: Barchart**

Input

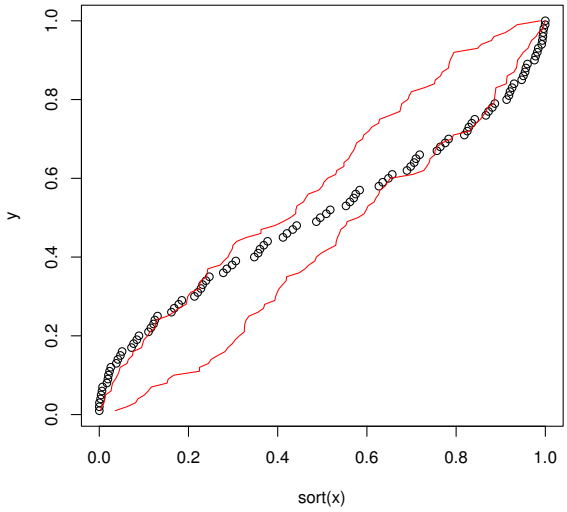
```
grades <- c(2, 1, 3, 4, 2, 2, 3, 5, 1, 3, 4, 3, 6)
barplot(table(grades), xlab = 'Grade', ylab = 'Count',
   ylim = c(0, max(table(grades))),
   main = 'Grading')
```



***Example* 1.11: Monte Carlo Confidence Bands**

Input

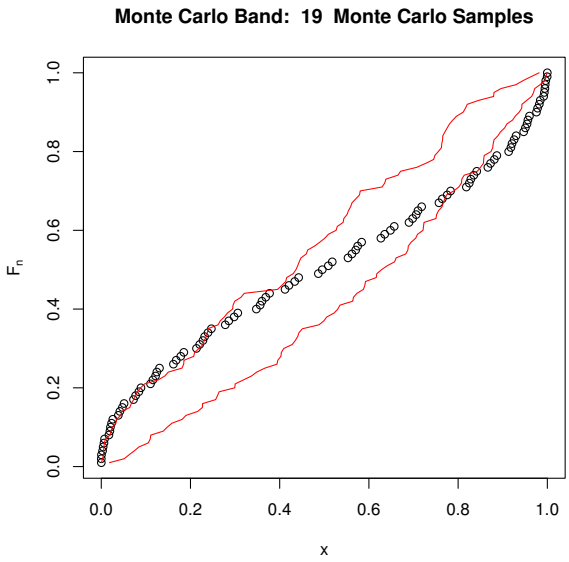
```
x <- (sin(1:100)+1)/2                    # demo example only
y <- (1:length(x))/length(x)
plot(sort(x), y)
nrsamples <- 19                          # no of simulations
samples <- matrix(data = runif(length(x)* nrsamples),
    nrow = length(x), ncol = nrsamples)
samples <- apply(samples, 2, sort)
envelope <- t(apply(samples, 1, range))
lines(envelope[, 1], y, col = "red")
lines(envelope[, 2], y, col = "red")
```



***Example* 1.12: Monte Carlo Confidence Bands (Augmented)**

Input

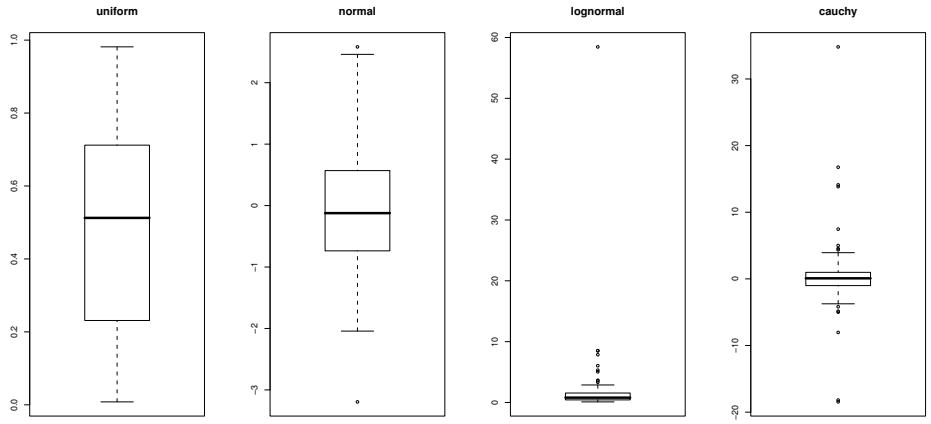
```
plot(sort(x), y,
    main = paste("Monte Carlo Band: ",
        bquote( .(nrsamples)), " Monte Carlo Samples"),
        xlab = 'x', ylab = expression(F[n]))
samples <- matrix(data = runif(length(x) * nrsamples),
    nrow = length(x), ncol = nrsamples)
samples <- apply(samples, 2, sort)
envelope <- t(apply(samples, 1, range))
lines(envelope[, 1], y, col = "red")
lines(envelope[, 2], y, col = "red")
```



***Example* 1.13: Box-and-Whisker Plot**

Input

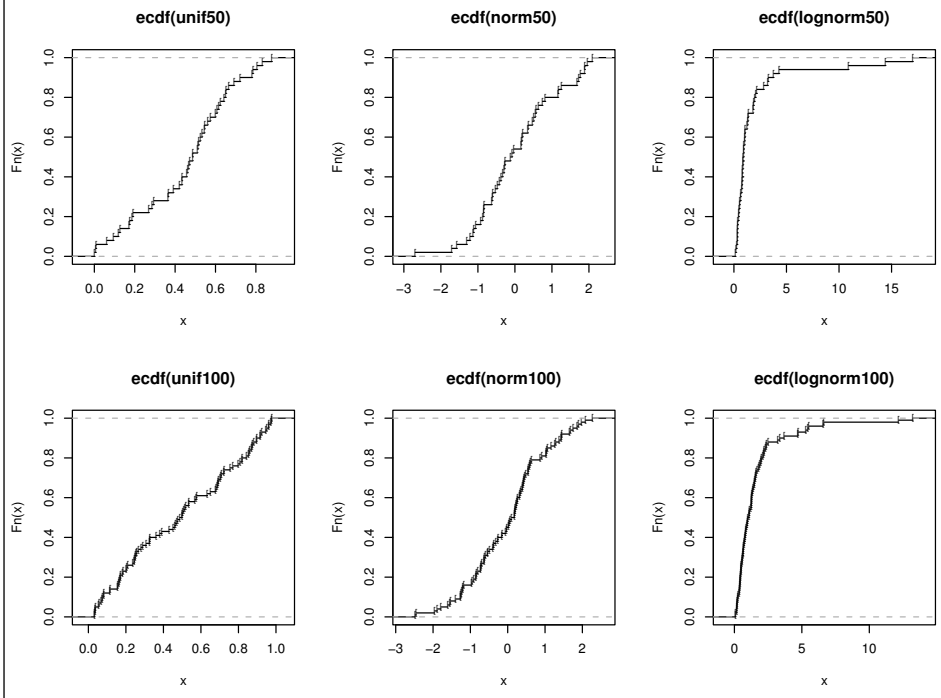
```
oldpar <- par(mfrow = c(1, 4))
boxplot(runif(100), main = "uniform")
boxplot(rnorm(100), main = "normal")
boxplot(exp(rnorm(100)), main = "lognormal")
boxplot(rcauchy(100), main = "cauchy")
par(oldpar)
```



***Example* 1.14: Distribution Functions (Various Distributions)**

Input

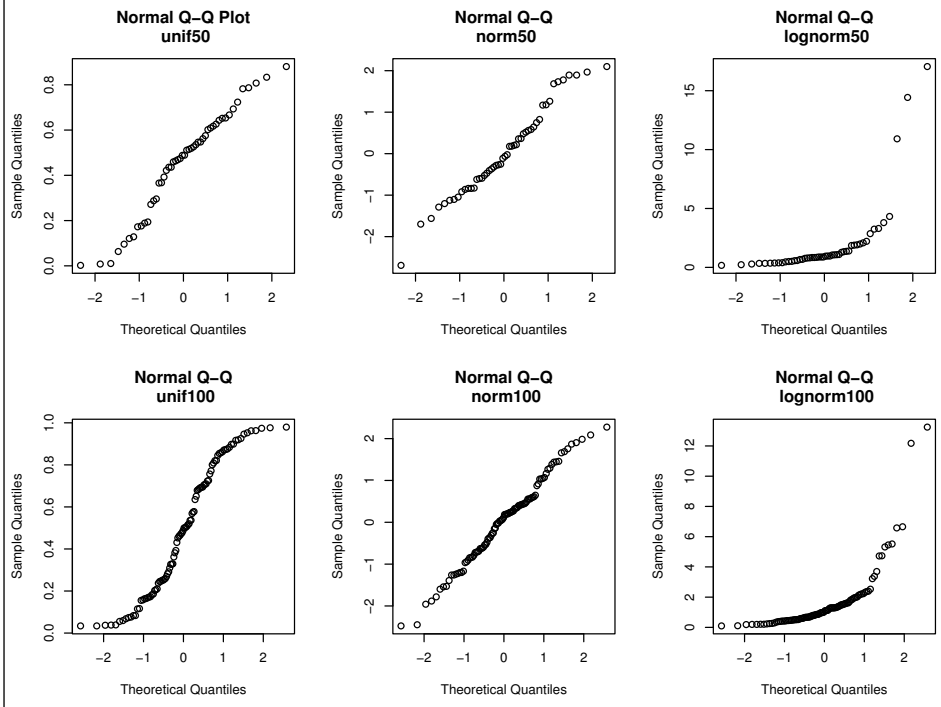
```
oldpar <- par(mfrow = c(2, 3))

plot(ecdf(unif50), pch = "[")
plot(ecdf(norm50), pch = "[")
plot(ecdf(lognorm50), pch = "[")

plot(ecdf(unif100), pch = "[")
plot(ecdf(norm100), pch = "[")
plot(ecdf(lognorm100), pch = "[")

par(oldpar)
```



***Example* 1.15: Normal QQ Plots (Various Distributions)**

Input

```
oldpar <- par(mfrow = c(2, 3))

qqnorm(unif50, main ="Normal Q-Q Plot\n unif50")
qqnorm(norm50, main = "Normal Q-Q\n norm50")
qqnorm(lognorm50, main = "Normal Q-Q\n lognorm50")

qqnorm(unif100, main = "Normal Q-Q\n unif100")
qqnorm(norm100, main = "Normal Q-Q\n norm100")
qqnorm(lognorm100, main = "Normal Q-Q\n lognorm100")

par(oldpar)
```



Exercise 1.26	
	Use PP plots instead of distribution functions to illustrate the χ^2- and Kolmogorov-Smirnov approximations.

Exercise 1.27	
	Use QQ plots instead of distribution functions. Can you add confidence regions to these plots with the help of the χ^2- resp. Kolmogorov-Smirnov statistics?

To get an impression of the fluctuation, we have to compare the empirical plots with typical plots of a model distribution. A plot matrix is a simple way to do this. Here is an example for the normal QQ plot, implemented as a function:

Input

```
qqnormx <- function(x, nrow = 5, ncol = 5, main = deparse(substitute(x))){
    oldpar <- par(mfrow = c(nrow, ncol))
    qqnorm(x, main = main)
    for (i in 1:(nrow*ncol-1))
     qqnorm(rnorm(length(x)), main = "N(0, 1)", xlab="", ylab="")
    par(oldpar)
}
```

In this example we used a *for* loop. Like all programming languages, R has control structures such as loops or conditional statements. In R, however, loops should be avoided if possible in favor of more efficient language constructs (see [**?**]). A summary of control structures in R can be found in Appendix **??** (page **??**).

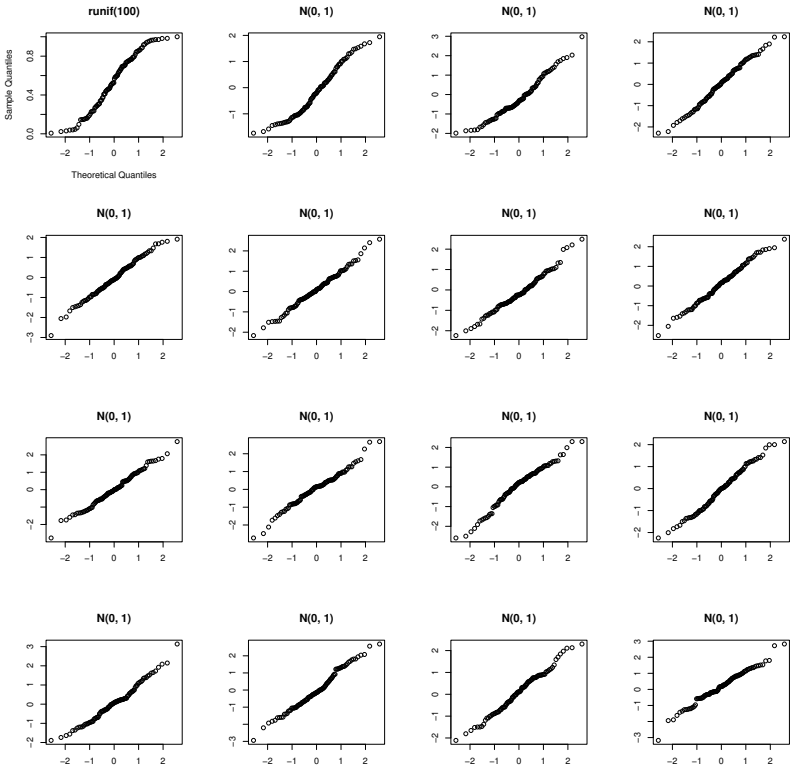
Exercise 1.28	
	Generate a matrix of dimensions $(nrow * ncol - 1), length(x)$ with random numbers and use *apply()* to avoid the loop. *Hint:* See Example 1.11 (page 32).

Deviations from a linear structure should be considered as fluctuations if they stay within the frame of the simulated examples. If the data set under investigation is too extreme in comparison with the simulated examples, this indicates a contradiction with the model assumptions. Because of space limitations of the printing area, we use only a small number of comparison plots here.

***Example* 1.16: Graphical Monte Carlo Test**

Input

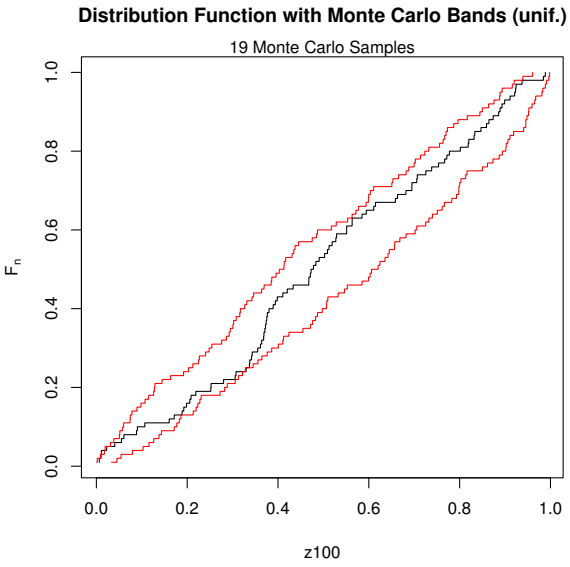
```
qqnormx(runif(100), nrow=4, ncol=4)
```



***Example* 1.17:** R **Function Call**

Input

```
z100 <- runif(100)
ppdemo(z100)
```



***Example* 1.18: R Function Listing**

Input

```
ppdemo
```

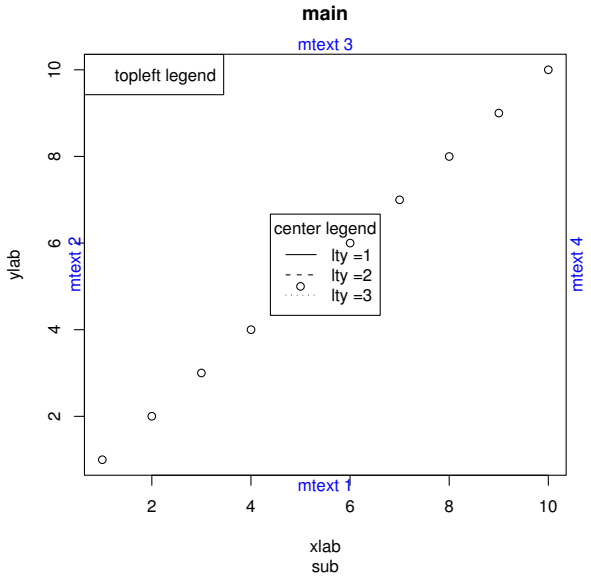
Output

```
function (x, samps  =  19) {     # samps: nr of simulations
    y <- (1:length(x))/length(x)
    plot(sort(x), y, xlab = substitute(x), ylab = expression(F[n]),
        main = "Distribution Function with Monte Carlo Bands (unif.)",
        type = "s")
    mtext(paste(samps, "Monte Carlo Samples"), side = 3)
    samples <- matrix(runif(length(x)* samps),
        nrow = length(x), ncol = samps)
    samples <- apply(samples, 2, sort)
    envelope <- t(apply(samples, 1, range))
    lines(envelope[, 1], y, type = "s", col = "red");
    lines(envelope[, 2], y, type = "s", col = "red")
}
```

***Example* 1.19: Margin Text**

Input

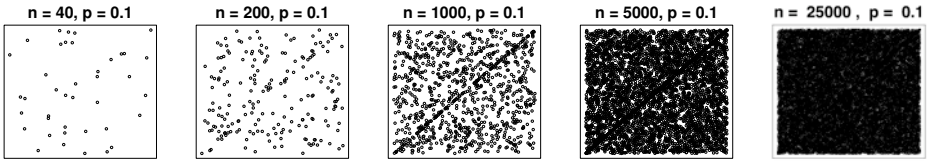
```
plot(1:10, xlab = "xlab", ylab = "ylab", main = "main", sub = "sub")
mtext("mtext 1", side = 1, col = "blue")
mtext("mtext 2", side  = 2, col = "blue")
mtext("mtext 3", side  = 3, col = "blue")
mtext("mtext 4", side  = 4, col = "blue")
legend("topleft", legend = "topleft legend")
legend("center", legend = c("lty =1", "lty =2", "lty =3"),
    lty =  1:3, title = "center legend")
```



***Example* 1.20: Needle in the Haystack**

Input

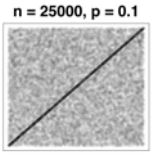
```
NeedleInTheHayStack <- function(nn, p=0.1, col="black", ... ) {
    oldpar <- par(mfrow=c(1,length(nn)))
    on.exit(par(oldpar))
    for (n in nn){
        nhay <- n-round(p*n); xhay <- runif(nhay);  yhay <- runif(nhay)
        needle <- runif(round(p*n))
        plot( x = c(xhay, needle), y = c(yhay,needle),
            main = paste("n = ", n, ", p = ", p, sep=""),
            cex.main=3.0,
            axes=FALSE, frame.plot=TRUE,
            xlab="", ylab="",
            col= col, ...)
        }
}
NeedleInTheHayStack( c(40, 200,1000, 5000, 25000) )
```



***Example* 1.21: Needle in the Haystack**

Input

```
NeedleInTheHayStack( 25000, col = rgb(red=0, blue=0, green=0, alpha=0.1))
```



***Example* 2.3: Least Squares Estimator**

Input

```
lm(y ~ x)
```

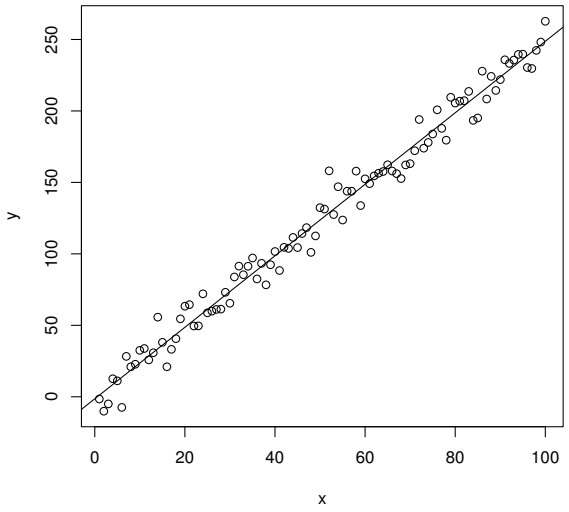
Output

```
Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
     -1.416        2.502
```

***Example* 2.4: Linear Model Plot**

Input

```
lmres <- lm(y ~ x)
plot(x, y)
abline(lmres)
```



***Example* 2.5: Linear Model Summary**

Input

```
summary(lm( y ~ x))
```

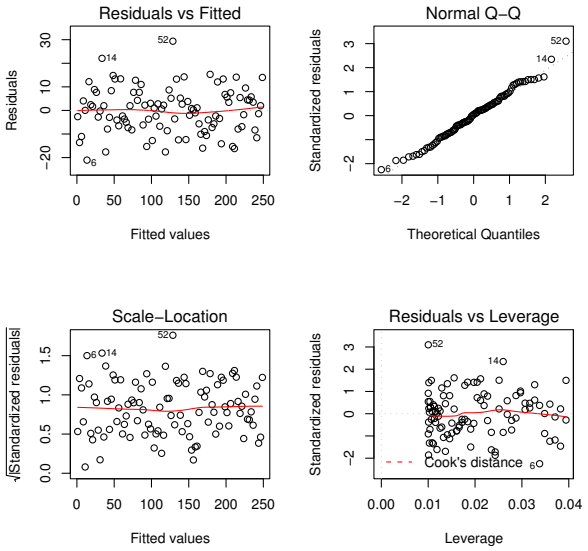
Output

```
Call:
lm(formula = y ~ x)

Residuals:
     Min       1Q   Median       3Q      Max
-21.0700  -6.7568   0.4417   5.6749  29.3925

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.41598    1.92062  -0.737    0.463
x            2.50154    0.03302  75.762   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 9.531 on 98 degrees of freedom
Multiple R-squared: 0.9832,        Adjusted R-squared: 0.983
F-statistic:  5740 on 1 and 98 DF,  p-value: < 2.2e-16
```

***Example* 2.6: Linear Model Plot**

Input

```
plot(lm(y ~ x))
```



***Example* 2.7: Linear Model ANOVA Summary**

Input

```
summary(aov(lmres))
```

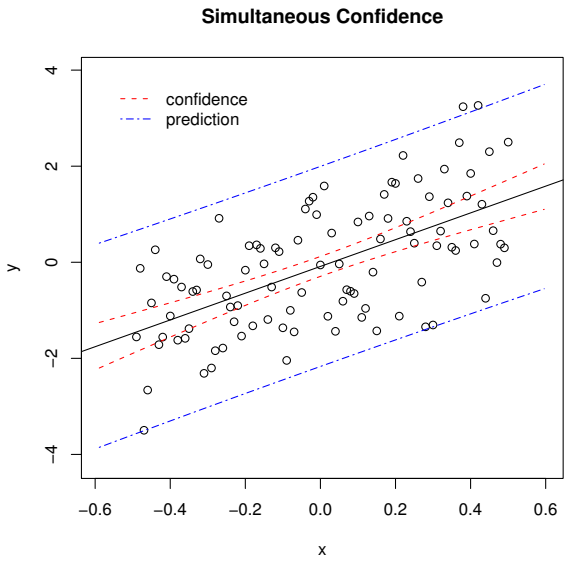
Output

```
            Df Sum Sq Mean Sq F value    Pr(>F)
x            1 521421  521421  5739.8 < 2.2e-16 ***
Residuals   98   8903      91
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Example **2.8: Linear Model Confidence Bands**

Input

```
plot(x, y, xlim = xlim, ylim = ylim)
abline(lmxy)
matplot(newx$x,
    cbind(pred.w.clim[, -1], pred.w.plim[, -1]),
    lty = c(2, 2, 6, 6),
    col = c(2, 2, 4, 4),
    type = "l", add = TRUE)
title(main = "Simultaneous Confidence")
legend("topleft",
    lty = c(2, 6),
    legend = c("confidence", "prediction"),
    col = c(2, 4),
    inset = 0.05, bty = "n")
```



***Example* 2.9: Tukey's Multiple Comparison**

Input

```
library(multcomp)
lhtres <- glht(lmres, linfct=mcp(Tmt="Tukey"))
summary(lhtres)   # multiple tests
```

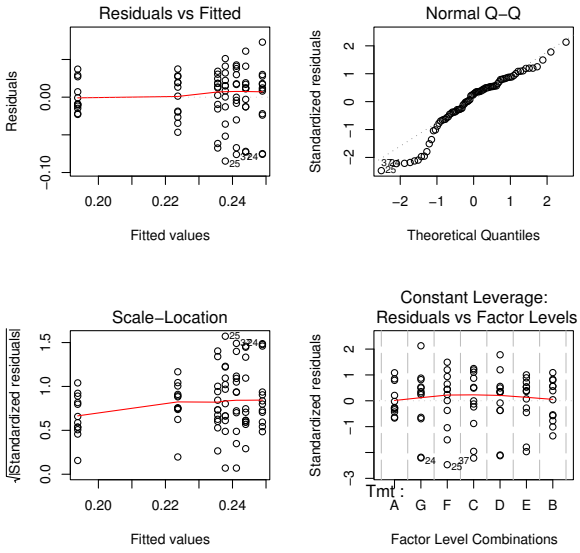
Output

```
        Simultaneous Tests for General Linear Hypotheses

Multiple Comparisons of Means: Tukey Contrasts


Fit: lm(formula = y ~ 0 + Tmt, data = s35)

Linear Hypotheses:
            Estimate Std. Error t value p value
B - A == 0  0.055083   0.014632   3.765 0.00597 **
C - A == 0  0.044000   0.014632   3.007 0.05256 .
D - A == 0  0.047333   0.014632   3.235 0.02838 *
E - A == 0  0.050083   0.014632   3.423 0.01642 *
F - A == 0  0.041750   0.014632   2.853 0.07792 .
G - A == 0  0.029833   0.014632   2.039 0.39924
C - B == 0 -0.011083   0.014632  -0.757 0.98819
D - B == 0 -0.007750   0.014632  -0.530 0.99832
E - B == 0 -0.005000   0.014632  -0.342 0.99986
F - B == 0 -0.013333   0.014632  -0.911 0.96971
G - B == 0 -0.025250   0.014632  -1.726 0.60097
D - C == 0  0.003333   0.014632   0.228 0.99999
E - C == 0  0.006083   0.014632   0.416 0.99958
F - C == 0 -0.002250   0.014632  -0.154 1.00000
G - C == 0 -0.014167   0.014632  -0.968 0.95931
E - D == 0  0.002750   0.014632   0.188 1.00000
F - D == 0 -0.005583   0.014632  -0.382 0.99974
G - D == 0 -0.017500   0.014632  -1.196 0.89359
F - E == 0 -0.008333   0.014632  -0.570 0.99748
G - E == 0 -0.020250   0.014632  -1.384 0.80870
G - F == 0 -0.011917   0.014632  -0.814 0.98280
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
(Adjusted p values reported -- single-step method)
```

***Example* 2.10: Multiple Comparison Model Check**

Input

```
oldpar <- par(mfrow=c(2,2))
plot(lmres)
par(oldpar)
```



***Example* 2.11: Diagnostic Using Studentized Residuals**

Input

```
library(MASS)
s35$studres <- studres(lmres)
s35[s35$studres < -1,]
```

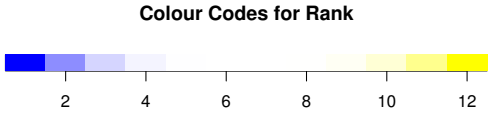
Output

```
       y Tmt Lane   studres
13 0.174   B    1 -2.239390
24 0.173   B   12 -2.271296
25 0.153   C    1 -2.559766
33 0.202   C    9 -1.044865
36 0.186   C   12 -1.523409
37 0.165   D    1 -2.279286
48 0.174   D   12 -1.994858
49 0.171   E    1 -2.175828
60 0.172   E   12 -2.144169
61 0.168   F    1 -2.007887
72 0.174   F   12 -1.821449
73 0.177   G    1 -1.367608
84 0.189   G   12 -1.010381
```

***Example* 2.12: Custom Colour Palette**

Input

```
oldpar <- par(yaxt="n")
image(x=1:12, z=matrix(1:12,ncol=1),
    col=blueyellow4.colors(12),
    xlab="",  main="Colour Codes for Rank")
par(oldpar)
```

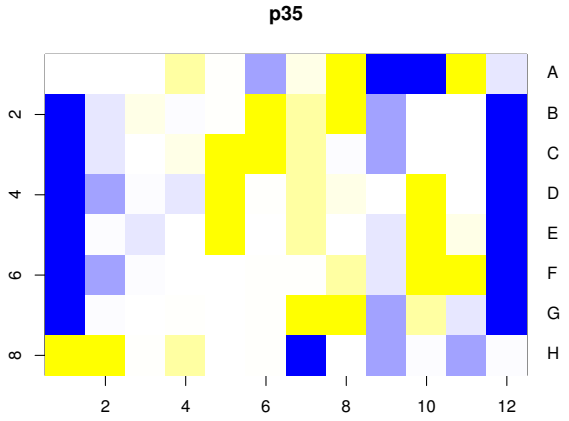


See Colour Figure 2.

***Example* 2.13: Residual Diagnostic**

Input

```
a35 <- as.matrix( p35[3:10] )
a35rk <- apply(a35, 2, rank)
imagem(t(a35rk), col=blueyellow4.colors(10), main="p35")
```

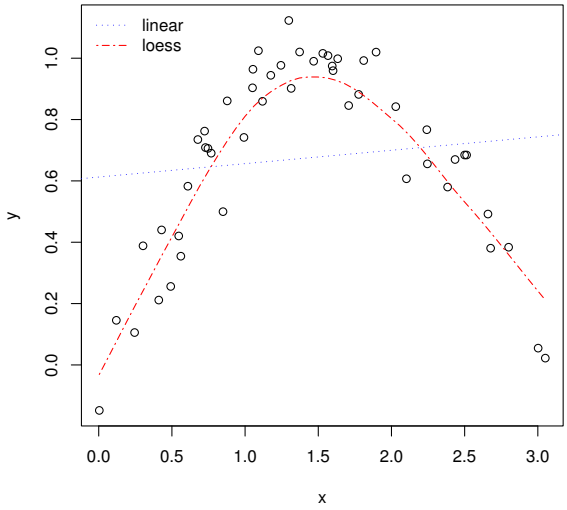


See Colour Figure 2.

***Example* 2.14: Non-Linear Regression**

Input

```
plot(x, y)
abline(lm(y ~ x), lty = 3, col = "blue")
lines(loess.smooth(x, y), lty = 6, col = "red")
legend("topleft",
    legend = c("linear", "loess"),
    lty = c(3, 6), col = c("blue", "red"), bty = "n")
```



***Example* 2.15: Method Dispatch**

Input

```
plot
```

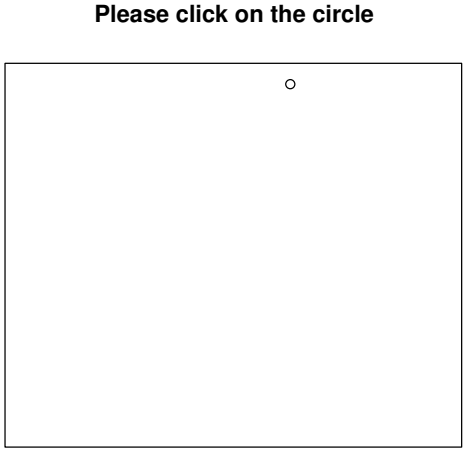
Output

```
function (x, y, ...)
{
    if (is.function(x) && is.null(attr(x, "class"))) {
        if (missing(y))
            y <- NULL
        hasylab <- function(...) !all(is.na(pmatch(names(list(...)),
            "ylab")))
        if (hasylab(...))
            plot.function(x, y, ...)
        else plot.function(x, y, ylab = paste(deparse(substitute(x)),
            "(x)"), ...)
    }
    else UseMethod("plot")
}
<environment: namespace:graphics>
```

Example **3.1: Interactive Location**

Input

```
plot(x = runif(1), y = runif(1),
     xlim = c(0, 1), ylim = c(0, 1),
     main = "Please click on the circle",
     xlab = '', ylab = '',
     axes = FALSE, frame.plot = TRUE)
xclick <- locator(1)
```



***Example* 3.2: Click Timing**

Input

```
click1 <- function(){
    x <- runif(1);y <- runif(1)
    plot(x = x, y = y, xlim = c(0, 1), ylim = c(0, 1),
        main = "Please click on the circle",
        xlab = '', ylab = '',
        axes = FALSE, frame.plot = TRUE)
    clicktime <- system.time(xyclick <- locator(1))
    list(timestamp = Sys.time(),
        x = x, y = y,
        xclick = xyclick$x, yclick = xyclick$y,
        tclick = clicktime[3])
}
```

***Example* 3.3: Sequential Recording**

Input

```
dx <- as.data.frame(click1())
dx <- rbind(dx, data.frame(click1()))
dx
```

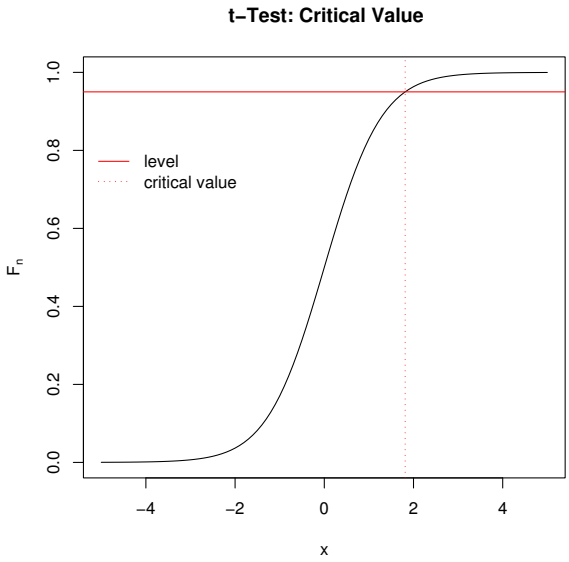
Output

```
                  timestamp      x      y xclick yclick tclick
elapsed  2008-12-12 15:14:26 0.7034 0.6072 0.7044 0.6070  1.899
elapsed1 2008-12-12 15:14:28 0.6948 0.9285 0.7003 0.9352  1.855
```

***Example* 3.4: Test Construction**

Input

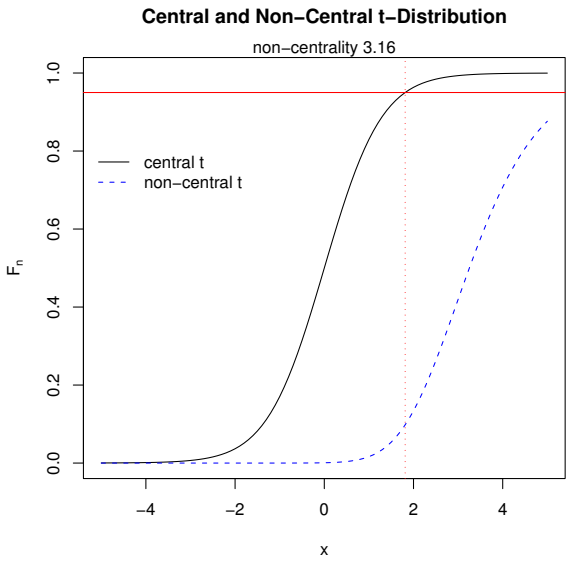
```
n1<- 6; n2 <- 6
df <-  n1 + n2 -2
alpha <- 0.05
curve(pt(x,df=df),from=-5, to=5, ylab= expression(F[n]),
    main="t-Test: Critical Value")
abline(h=1-alpha, col="red")      # cut at upper quantile
abline(v=qt(1-alpha, df=df), lty=3, col="red") # get critical value
legend("topleft", legend=c("level","critical value"),
    lty=c(1,3),col="red",
    bty="n", inset=c(0,0.2))
```



***Example* 3.5: Power (Fixed Non-Centrality)**

Input

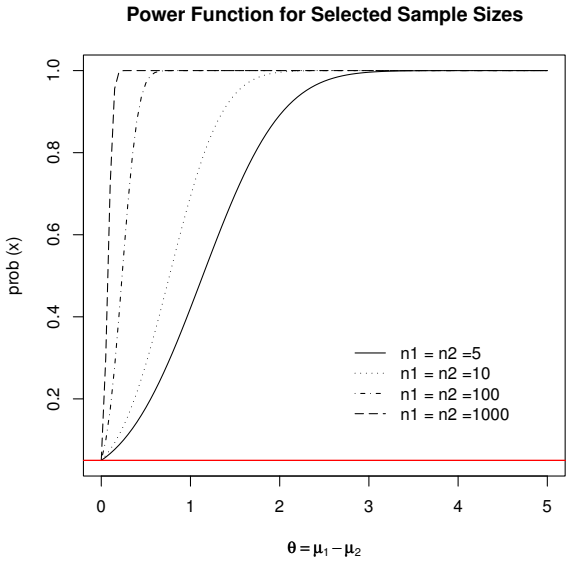
```
n1<- 6; n2 <- 6
df <-  n1 + n2 -2
alpha <- 0.05
curve(pt(x,df=df),from=-5, to=5, ylab= expression(F[n]),
    main="Central and Non-Central t-Distribution")
abline(h=1-alpha, col="red")      # cut at upper quantile
abline(v=qt(1-alpha, df=df), lty=3, col="red") # get critical value
n1 <- 5
n2 <- 5
n <- n1+n2
theta <- 2
ncp <- theta * sqrt(n1 * n2/(n1+n2))
mtext(paste("non-centrality",round(ncp,2)))
curve(pt(x,df=df, ncp=ncp), lty=2, add=TRUE, col="blue")
legend("topleft", legend=c("central t","non-central t"),
    lty=c(1,2), col=c("black","blue"),
    bty="n", inset=c(0,0.2))
```



***Example* 3.6: Power Function**

Input

```
tpower <- function(n1, n2, alpha,...){
        df <-  n1 + n2 -2
        tlim <- qt(1-alpha,df=df)
        prob <- function(theta){
            pt(tlim, df = df,
                ncp = theta * sqrt(n1 * n2/(n1+n2)),
                lower.tail=FALSE)}
        curve(prob, 0, 5, xlab=expression(theta==mu[1]-mu[2]), ...)
        abline(h=alpha, col="red")
}

tpower(5, 5, 0.05, main="Power Function for Selected Sample Sizes")
tpower(10, 10, 0.05, add =TRUE, lty = 3)
tpower(100,100, 0.05, add =TRUE, lty = 4)
tpower(1000, 1000, 0.05, add =TRUE, lty = 5)
legend("bottomright",
    lty=c(1,3,4,5),
    legend=c("n1 = n2 =5", "n1 = n2 =10", "n1 = n2 =100","n1 = n2 =1000"),
    inset=0.1, bty="n")
```



***Example* 3.7: Sample Size Determination**

Input

```
power.t.test(delta=2,
    power=0.8,
    sig.level=0.01,
    type="two.sample",
    alternative="one.sided")
```

Output

```
    Two-sample t test power calculation

              n = 6.553292
          delta = 2
             sd = 1
      sig.level = 0.01
          power = 0.8
    alternative = one.sided

NOTE: n is number in *each* group
```

***Example* 3.8: Violation of Assumptions (log-normal)**

Input

```
nsimul <- 500
n1<- 10; n2 <- 10
alpha <- 0.01 #nominal level
x <- 0
for (i in 1:nsimul) {
        if (t.test(exp(rnorm(n1)),exp(rnorm(n2)),
        alternative="less",
        var.equal = TRUE)$p.value < alpha){
            x <- x+1}
         }
p <- x/nsimul
cat("estimated level p", p)
```

Output

```
estimated level p 0.006
```

***Example* 3.9: Binomial Confidence Set**

Input

```
prop.test(n=nsimul, x=x)
```

Output

```
        1-sample proportions test with continuity correction

data:  x out of nsimul, null probability 0.5
X-squared = 486.098, df = 1, p-value < 2.2e-16
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.001550854 0.018951807
sample estimates:
    p
0.006
```

***Example* 3.10:**

Input

```
nsimul <- 500
n1<- 10; n2 <-10
alpha <- 0.01
x<-0
for (i in 1:nsimul) {
        if (t.test(exp(rnorm(n1)),exp(rnorm(n2, mean = 1)),
            alternative="less",
            var.equal = TRUE)$p.value < alpha){
                x <- x+1}
         }
p <- x/nsimul
cat("estim p", p)
```

Output

```
estim p 0.188
```

Input

```
prop.test(n = nsimul, x = x)
```

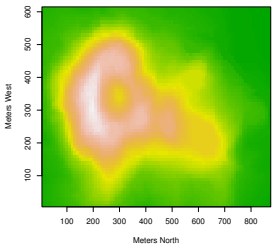
Output

```
        1-sample proportions test with continuity correction

data:  x out of nsimul, null probability 0.5
X-squared = 193.442, df = 1, p-value < 2.2e-16
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.1552568 0.2256388
sample estimates:
    p
0.188
```

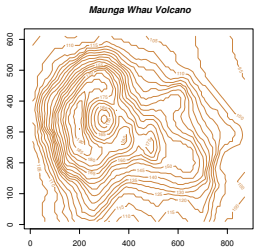
***Example* 4.1: 3d Surface Displays Using Base Graphics**

Input

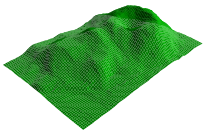
```
x <- 10*(1:nrow(volcano))
y <- 10*(1:ncol(volcano))
image(x, y, volcano, col = terrain.colors(100),
    axes = FALSE, xlab = "Meters North", ylab = "Meters West")
axis(1, at = seq(100, 800, by = 100))
axis(2, at = seq(100, 600, by = 100))
box()
title(main = "Maunga Whau Volcano", font.main = 4)

contour(x, y, volcano, levels = seq(90, 200, by = 5),
         col = "peru", main = "Maunga Whau Volcano", font.main = 4,
         xlab = "Meters North", ylab = "Meters West")
z <- 2 * volcano        # Exaggerate the relief
x <- 10 * (1:nrow(z))   # 10 meter spacing (S to N)
y <- 10 * (1:ncol(z))   # 10 meter spacing (E to W)

## Don't draw the grid lines :  border = NA
persp(x, y, z, theta = 135, phi = 30, col = "green3", scale = FALSE,
      ltheta = -120, shade = 0.75, border = NA, box = FALSE)
```



image()



contour()



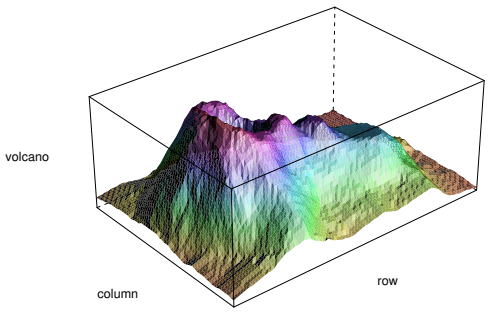
persp()

See Colour Figure 3.

***Example* 4.2: 3d Surface Display Using Lattice Graphics**

Input

```
library(lattice)
print(wireframe(volcano, shade = TRUE,
          aspect = c(61/87, 0.4),    ## volcano  ## 87 x 61 matrix
          par.settings = list(axis.line = list(col = "transparent")),
          light.source = c(10,0,10)))
```



See Colour Figure 4.

***Example* 4.3: Scatterplot Matrix for Diabetes Data**

Input

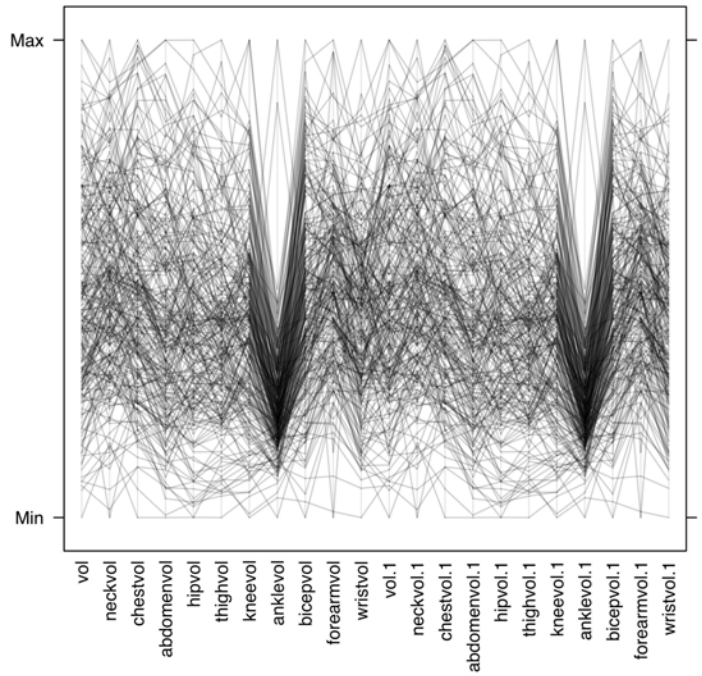
```
pairs(~fpg + ga + ina + sspg, data = chemdiab,     pch = 21,
    main = "Diabetes-data",
     bg = c("magenta", "green3", "cyan")[unclass(chemdiab$cc)],
    oma = c(8, 8, 8, 8))
mtext(c("Colour codes:", levels(chemdiab$cc)),
    col = c("black", "magenta", "green3", "cyan"),
    at = c(0.1, 0.4, 0.6, 0.8), side = 1, line = 2)
```

Output: see Figure 4.1.

***Example* 4.4: Parallel Plots**

Input

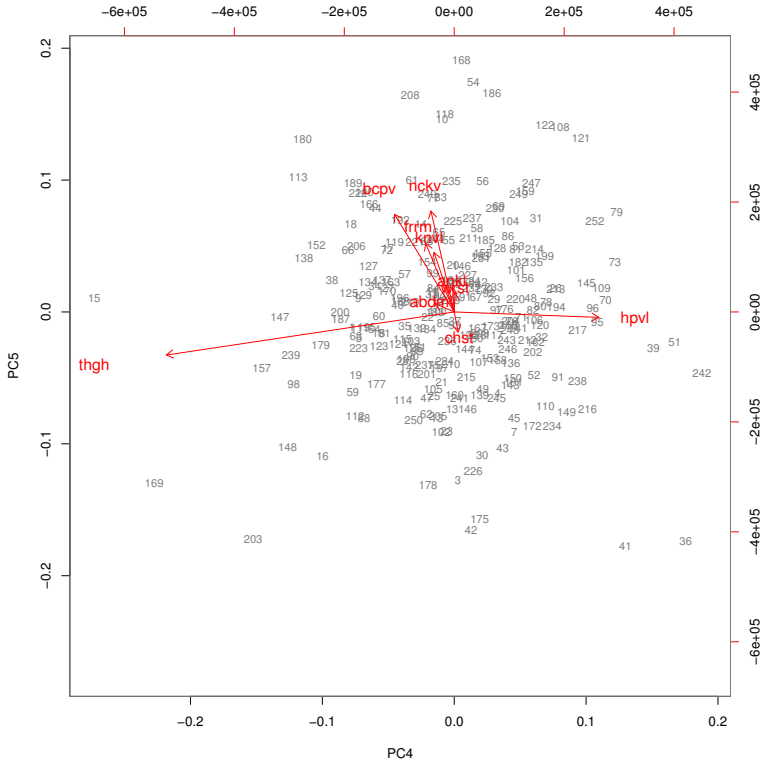
```
print(parallel(fat[ -c(39, 41, 216), c(19:29,19:29)],
    horizontal.axis = FALSE, scales = list(x = list(rot = 90)),
    col =  rgb(red=0, blue=0, green=0, alpha=0.2))
)
```



***Example* 4.5: Biplot**

Input

```
biplot(pcfatvol, choices = 4:5,
     col = c("grey50", "red"),
     ylabs = abbreviate( rownames(pcfatvol$rotation), 4 ),
     cex = c(0.8, 1.2))
```



See Colour Figure 11.