

Package ‘cwhmisc’

January 19, 2012

Version 3.0

Date 2012-01-12

Title Miscellaneous Functions for maths, plotting, printing, statistics, strings, and tools

Author Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Maintainer Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Depends R (>= 2.0), lattice, grid

Description Miscellaneous useful functions collected over time

License GPL (>= 2)

Repository CRAN

Date/Publication 2012-01-19 08:12:36

R topics documented:

adaplob	3
cap	4
cbind.colnames	5
clean.na	6
cpos	7
cwhmisc-internal	8
datetime	9
dc	9
delayt	11
delstr	11
div.prot	12
dt2str	13
ellipse	13
eq1	15
f.log	15
FinneyCorr	16

formatFix	17
frac	18
heading	19
interpol	20
invgauss	23
is.constant	23
jitterNA	24
lengths.angle	25
libs	26
like	26
lowess.bygroup	27
lpr	28
ls.functions	28
mult.fig.p	29
my.table	30
n22dig	31
n2c	32
napply	33
negbingof	34
normalize	35
num.ident	35
num2Latex	36
numberof	37
numericString	37
p.screepplot.princomp	38
padding	39
panel	40
pasteInfix	40
pasteRound	41
persp2	42
plotSymbols	43
plt	44
pointfit	46
poisgam	47
printP	48
progress.meter	50
qnorm.appr	51
qres	52
remove.dup.rows	53
replacechar	53
rotm	54
rtf	55
scode	57
select.range	58
seqm	59
setPowerPointStyle	60
shapiro.wilk.test	60
signp	61

smoothed.df	62
solveQeq	63
str2formula	63
strmatch	65
summaryFs	66
T3plot	66
tex.table	67
tri	68
waitReturn	69
weighted.mean1	70
weighted.median	71
whole.number	72

Index 73

adaptlob *Numerically evaluate integral using adaptive rule.*

Description

adaptsim and adaptlob approximate the integral of the function f using *adaptive* Simpson and Lobatto rule. Both methods can deal with discontinuous functions.

adaptlob is more efficient than adaptsim when the accuracy requirement is high. For lower tolerances, adaptsim is generally (but not always) more efficient than adaptlob, but less reliable.

Both routines show excellent response to changes in the tolerance.

The function f must return a vector of output values if given a vector of input values.

adapt...(f, a, b) approximates the integral of $f(x)$ from a to b to *machine* precision.

adapt...(f, a, b, tol) integrates to a *relative* error of tol .

adapt...($f, a, b, tol, trace=TRUE$) displays the stepwise left end point of the current interval, the interval length, and the partial integral. adapt...($f, a, b, tol, trace, P1, P2, \dots$) allows coefficients $P1, \dots$ to be passed directly to the function f : $g <- f(x, P1, P2, \dots)$

Usage

```
adaptsim(f, a,b,tol=.Machine$double.eps,trace=FALSE,...)
adaptlob(f, a,b,tol=.Machine$double.eps,trace=FALSE,...)
```

Arguments

- f function to be integrated.
- a starting abscissa of integral.
- b ending abscissa of integral.
- tol error tolerance > 0 of the final result.
- trace if TRUE then trace will be displayed.
- ... additional coefficients for function f if necessary.

Value

List (Q, term) with Q = the approximate value of the integral and term = the information, whether the tolerance given was too small.

Author(s)

Christian W. Hoffmann, <c-w.hoffmann@sunrise.ch>

Source

Walter Gautschi, 08/03/98. Reference: Gander, Computermathematik, Birkhaeuser, 1992.

References

Gander, W., Gautschi, W., 2000. Adaptive Quadrature - Revisited. ETH Zurich, DI IWR technical report 306. BIT 40, 1, 84–101.

Examples

```
options(digits=7)
FexGander <- function(xx) ifelse(xx < 1,xx+1,ifelse(xx <= 3, 3 - xx, 2 ))
adaptsim(sin,0,pi,2.0e-3,TRUE)$Q - 2.0 # -1.686905e-05
adaptsim(sin,0,pi,2.0e-23)$Q - 2.0 # 0
adaptsim(FexGander,0,5)$Q - 7.5 # -7.993606e-15 instead of 0
adaptlob(FexGander,0,5,2.0e-6,TRUE) # 7.500002 instead of 7.5
adaptlob(FexGander,0,5,2.0e-6)$Q - 7.5 # 1.781274e-06 instead of 0
adaptlob(FexGander,0,5)$Q-7.5 # instead of -8.881784e-16, with warnings
# that required tolerance is too small.
adaptlob(FexGander,0,5,5.0*.Machine$double.eps)$Q-7.5 # -5.329071e-15
```

 cap

Change case of strings

Description

capply apply function to elements in character vector (utility function) cap and capitalize change to capital letters. lower and lowerize change to lower case letters. CapLeading Capitalizes the first character of each element of a character vector

Usage

```
capply(str,ff,...)
cap(char)
capitalize(str)
lower(char)
lowerize(str)
CapLeading(str)
```

Arguments

str	a character vector.
ff	a function.
char	a single letter.
...	a function .

Value

The same as the argument.

Note

capply has been reverse engineered from the help page on strsplit: `strReverse <- function(x) sapply(lapply(strsplit(x, NULL), rev), paste, collapse="")` can be written as `capply(x, rev)`

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>
http://www.wsl.ch/personal_homepages/hoffmann/index_EN

Examples

```
# capitalize shows the use of capply
cap("f")      # "F"
capitalize(c("TruE", "faLse")) # "TRUE" "FALSE"
lower("R")    # "r"
lowerize("TruE") # "true"
CapLeading(c("all you ", "need")) # "All you " "Need"
capply(c("abc", "elephant"), rev) # "cba" "tnahpele"
```

cbind.colnames	<i>Add columns to a data frame, using variable names.</i>
----------------	---

Description

cbind.colnames adds columns to a given data frame. The names of the variables to be added are given in character vector add.

Usage

```
cbind.colnames(add, to=NULL, deparse.level = 1)
```

Arguments

to	Matrix to be augmented.
add	Vector of names of variables to be added to to as columns.
deparse.level	Usually 1, See cbind

Value

Combined matrix. The column names are the concatenation of those of to and add. Row names are taken from to.

Note

Uses get to access the variables listed in add. If the new variables have wrong lengths, an error will result.

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
d <- data.frame(E = c("D", "at", "a"), w = c(11, 22, 33))
x <- c(2, 5, 1)
ch <- c("I", "have", "fun")
F <- factor(c("A", "B", "C"))
cbind.colnames(c("F", "ch", "x", "F"), d)
#   E w F   ch x F
# 1 D 11 A   I 2 A
# 2 at 22 B have 5 B
# 3 a 33 C fun 1 C
cbind.colnames(c("x", "F", "ch", "x"))
data.frame(x, F, ch, x) # the same
cbind.colnames(NULL) # NULL
```

clean.na

Clean a matrix or data frame of rows or columns of containing NA.

Description

clean.na Eliminate rows or columns containing NA.

Usage

```
clean.na(x, margin, drop=FALSE)
```

Arguments

x	A matrix.
margin	= 1 for rows, = 2 for columns
drop	= FALSE (default) if result should be a matrix even if it contains only one row or column.

Value

The matrix without the offending rows or columns.

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

See Also

[drop](#).

Examples

```
x <- matrix(c(1,NA,2,5),2,2)
clean.na(x,1)
#      [,1] [,2]
#[1,]    1    2
clean.na(x,2,TRUE)
# [1] 2 5
```

cpos

Find the position of a substring

Description

cpos finds the first position of a substring `substring.location` returns a list with starting and ending positions, works only with a single string.

Usage

```
cpos(str,sub,start=1)
substring.location(str, sub, restrict)
```

Arguments

str	string (1-dim)
sub	string (1-dim)
start	integer
restrict	vector of lower and upper index the search should be restricted to

Value

number, if found, NA otherwise. `list(first,last)`

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
cpos(" Baldrian", "a", 5) # 8
cpos("Baldrian", "B", 15) # NA
substring.location("In,these,housees,there,are,rats", "ees")
#$first [1] 6 15
#$last  [1] 8 17
```

cwhmisc-internal *Internal cwhmisc functions*

Description

Internal cwhmisc functions

Usage

```
adaptsimstp(f, term, a, b, fa, fm, fb, is, trace, ...)
adaptlobstp(f, term, a, b, fa, fb, is, trace, ...)
```

Arguments

f	function to be integrated.
term	starting abscissa of integral.
a	left abscissa of integral.
b	reight abscissa of integral, $b > a$.
fa	=f(a).
fb	=f(b).
fm	=f((a+b)/2).
is	used for stopping at a suitable multiple of the machine precision.
trace	if TRUE then trace will be displayed.
...	additional coefficients for function f if necessary.

Details

These functions are not to be called by the user.

Value

The approximate value of the integral.

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>
http://www.wsl.ch/personal_homepages/hoffmann/index_EN

datetime	<i>Show date and time in ISO format</i>
----------	---

Description

datetime() outputs date and time in ISO format

Usage

```
datetime(); mydate(); mytime()
```

Arguments

none

Value

character string

Note

These functions are implemented using POSIX

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
datetime() #[1] "2001-02-07, 15:24:51"  
mydate()   #[1] "2001-02-07"  
mytime()   #[1] "15:24:12"
```

dc	<i>Convert number for table columns, for equations</i>
----	--

Description

Convert number

- for use in decimal dot centered table columns: Replace "." in a number by "&" for LaTeX tables using column specification r\@{.}l.

- mpf(r,n) returns "+ r" or "- r", depending on the sign of r, with n decimal digits. Useful in [Sweave](#) files *.Rnw for composing text for linear combinations with coefficients shown in \Sexp.

Usage

```
dc (x,d,ch="&")
dcn(x,d,ch="&")
mpf(r,after)
```

Arguments

x	Numerical vector.
d	Number of decimals after ".". $d \geq 1$, will be forced internally.
ch	Substitute "." by ch
after	See formatFix, the number of decimals after ".".
r	real value.

Value

string representation of x suitable for table column centered on "."

Note

dc = dcn, except for $x = \text{integer}$.
 dc uses frac, dcn uses `sprintf`.

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
nn <- c(0, 1, 0.1, pi, 2*pi, -30*pi)
dc(nn,3) # "0&0" "1&0" "0&100" "3&142" "6&283" "-94&248"
dcn(nn,3) # "0&000" "1&000" "0&100" "3&142" "6&283" "-94&248"
```

```
#### In file T.Rnw:
## <<echo=TRUE>>=
a <- -2; b <- -4; c <- 7
## @
##
## The coefficients are: $a = \Sexpr{a}$, $b = \Sexpr{b}$, $c = \Sexpr{c}$.
##
## For the linear combination $$z = a + bx + cy$$ we have
## $$z = \Sexpr{sprintf("%.4f,a)} \Sexpr{mpf(b,3)} x \Sexpr{mpf(c,5)} y$$
#### end T.Rnw
### Sweave: T.Rnw .. T.tex .. T.dvi
```

```
mpf(pi,5); mpf(-pi,5) # "+ 3.14159" "- 3.14159" Note the space after the sign.
```

delayt	<i>Waiting loop for program execution</i>
--------	---

Description

Wait for approximately sec seconds during program execution

Usage

```
delayt(sec) # wait for sec seconds
```

Arguments

sec	Number of seconds to wait
-----	---------------------------

Details

calls Sys.time()

Value

the number of internal calls of Sys.time()

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
Sys.time(); nrof <- delayt(5); Sys.time()
print(nrof) # 2620 on my machine
```

delstr	<i>Delete a substring from a string</i>
--------	---

Description

delstr deletes a substring from a string

Usage

```
delstr(str,del)
```

Arguments

str	a string, may be empty, string to be edited
del	a string, may be empty, string to be taken out.

Value

A string

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
delstr("Ich gehen heute in den Garten hinein","en")
# "Ich geh heut in d Gart hinein"
```

div.prot

Protected division

Description

1/x, but 1/0 -> 0

Usage

```
div.prot(x, eps = .Machine$double.eps)
```

Arguments

x	array to invert.
eps	limit: if abs(x) < eps then 1/x -> 0.

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
div.prot(c(0, .Machine$double.eps, 1.0e-10, 1, 1.0e100))
div.prot(c(0, .Machine$double.eps, 1.0e-10, 1, 1.0e100), 100*.Machine$double.eps)
```

dt2str *Convert time difference to string.*

Description

Convert time difference in seconds to string depending on switch.

Usage

```
dt2str(dt,dec=0,verbose=FALSE)
```

Arguments

dt	Time difference in seconds
dec	Places in decimal fraction of seconds
verbose	If TRUE, then delimited by "hours minutes seconds", else by ":"

Value

String representing the time difference.

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
time1 <- Sys.time()
x <- 0
for (i in 1:100000) x <- x+1
time2 <- Sys.time()
dt2str(unclass(time2)-unclass(time1))
dt2str(unclass(time2)-unclass(time1),,TRUE)
```

ellipse *Generate ellipses*

Description

Given a positive definite symmetric matrix A of dimension 2 by 2 and a constant cn , or the axes a , b and an angle ϕ (in radian, counter clockwise), and the midpoint coordinates m , points on the ellipse $(y - m)' * A^{(-1)} * (y - m) = cn^2$ or `rotm(2,1,2,phi) %*% matrix(c(a,0,0,b), 2, 2)` will be generated.

Usage

```
ellipse(k, m, A = NULL, cn = NULL, a = NULL, b = NULL, phi = NULL)
```

Usage

```
ellipse(k, m, A, cn)
ellipse(k, m, a=, b=, phi=)
conf.ellipse(k, m, A, df, level = 0.95)
```

Arguments

k	the number of generated points on the ellipse.
m	vector of length 2 containing the midpoint coordinates of the ellipse.
A	positive definite symmetric matrix of dimension 2 by 2
cn	positive constant.
a	major axis
b	minor axis
phi	angle in radian describing the counter clockwise rotation from the x-axis to the major axis.
df	degree of freedom of F-distribution.
level	probability level of F-distribution F(2,df).

Value

The matrix with columns consisting of the x and y coordinates of the ellipse.

Author(s)

Of ellipse: originally Bernhard Flury and Marco Bee for Flury's book "A First Course in Multivariate Statistics" (Springer 1997). Of conf.ellipse: Roger Koenker <roger@ysidro.econ.uiuc.edu>, <http://www.econ.uiuc.edu> May 19, 1999. Refined by: Christian W. Hoffmann <christian.hoffmann@wsl.ch>

See Also

[rotm](#)

Examples

```
A <- matrix(c(1,1,1,2), ncol = 2) # define matrix A
m <- c(3, 4) # define vector m
plot(ellipse(1000,m,A,1))

plot(pe <- ellipse(800,m,A,1),pch=".",type="n")
lines(rbind(pe,pe[1,]))
lines({pe <- ellipse(600,m,A,0.5); rbind(pe,pe[1,])})
```

```
lines({pe <- ellipse(400,m,A,0.25); rbind(pe,pe[1,])})
lines(conf.ellipse(51,m,A,20,0.9),lty=4,col="red")
lines(conf.ellipse(51,m,A,20,0.8),lty=4,col="green")
```

eql *Check on equality, including NA==NA and NaN==NaN.*

Description

my.table checks two vectors on equality, two NA's and two NaN's compare as equal.

Usage

```
eql(x, y)
```

Arguments

x, y vectors of equal length.

Value

A vector of logicals indicating the result of the element by element comparison. The elements of shorter vectors are recycled as necessary.

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>,
idea by Peter Dalgaard, <p.dalgaard@biostat.ku.dk>

Examples

```
eql(c(1,2,3),c(1,3)) #> TRUE FALSE FALSE
eql(c(1,2,3),c(1,2)) #> TRUE TRUE FALSE
eql(c(NA,NaN,2,3),c(NA,NaN,1,2)) #> TRUE TRUE FALSE FALSE
```

f.log *Determine an optimized offset s and return log10(data+s).*

Description

f.log determines a positive offset s for zero values to be used in a subsequent log transformation.

Usage

```
f.log(x)
```

Arguments

`x` vector of data.

Value

The transformed values $\log_{10}(\text{data} + s)$.

Note

The value for the offset `s` is optimized to render the transformed values of `x` log-normal

Author(s)

W.Stahel, ETH Zuerich, <werner.stahel@stat.math.ethz.ch> adapted by: Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
x <- c(rep(0,20), exp(rnorm(1000)))
fx <- f.log(x)
## Not run: oldpar <- par(mfrow = c(2, 3))
plot(x)
qqnorm(x)
T3plot(x)
plot(fx)
qqnorm(fx)
T3plot(fx)
par(oldpar)

## End(Not run)
```

FinneyCorr

Finney's correction to log normally distributed data, r-squared and standard deviation of a linear model.

Description

FinneyCorr: Finney's correction factor K in $x = e^{\ln x} * K$ (see Note), to be used if $\ln x$ is normally distributed with standard deviation $s_{\ln x}$. FC.lm, R2.lm, s.lm: Finney's correction, R^2 and standard deviation extracted from an object of class "lm".

Usage

```
FinneyCorr(s,n)
FC.lm(lmobj)
R2.lm(lmobj)
s.lm(lmobj)
```

Arguments

s	Standard deviation of log data.
n	Number of data points.
lmoobj	An object of class "lm".

Note

$$K := e^{s_{\ln}^2/2} \left\{ 1 - \frac{s_{\ln}^2}{4n} (s_{\ln}^2 + 2) + \frac{s_{\ln}^4}{96n^2} (3s_{\ln}^4 + 44s_{\ln}^2 + 84) \right\}$$

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

References

Finney D.J., 1941. On the distribution of a variable whose logarithm is normally distributed. J. R. Stat. Soc., B 7: 155-161

Examples

```
FinneyCorr(0.346274,24+3) # 1.059306936

ok <- RNGkind()
RNGkind(kind = "default", normal.kind = "default")
set.seed(2009, kind = "default")
x <- rnorm(1000); y <- 0.1*rnorm(1000)
## Reset:
RNGkind(ok[1])

lmo <- lm(y ~ x)
FC.lm(lmo) # 1.00472
R2.lm(lmo) # 6.1926e-05
s.lm(lmo) # 0.0970954
```

formatFix

Format to a fixed format representation

Description

formatFix formats to fixed point number format. It 'writes' x with sign (" " or "-") and 'before' decimals before the "." and with 'after' decimals after the ".". If 'after'==0 then the "." will be omitted.

There will always be at least one decimal digit before the "."

If 'before' is too small to represent x: if extend==TRUE, the string will be extended, else a string consisting of "*" of length before+after will be given.

If \$abs(x) >= 10^8\$ values very near \$10^k\$ cannot be represented exactly, so the normal format will be used.

Names are retained.

Usage

```
formatFix(x, after, before=2, extend=TRUE)
```

Arguments

x	The number to be represented.
after	The number of decimals after ".".
before	The minimum number of decimals before ".".
extend	Extend string if necessary.

Value

The string representing the fixed point format of x.

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
xxbig <- c(1.2e9, 3.51e23, 6.72e120, NaN)
xx <- c(0.001, 92, exp(1), 1000*pi)
formatFix(c(-rev(xxbig), -rev(xx), 0, NA, xx, xxbig), 0, 3)
#> [1] " NaN" "-7e+120" "-4e+23" "-1e+09" "-3142" " -3" " -92"
#> [8] " -0" " 0" " NA" " 0" " 92" " 3" " 3142"
#> [15] " 1e+09" " 4e+23" " 7e+120" " NaN"
formatFix(c(-rev(xxbig), -rev(xx), 0, NA, xx, xxbig), 0, 3, FALSE)
#> [1] "NaN" "xxx" "xxx" "xxx" "xxx" "-3" "-92" "-0" " 0" " NA" " 0" " 92"
#> [13] " 3" "xxx" "xxx" "xxx" "xxx" "NaN"
formatFix(c(-rev(xxbig), -rev(xx), 0, NA, xx, xxbig), 6, 3)
#> [1] " NaN" "-6.72e+120" "-3.51e+23" "-1.2e+09" "-3141.592654"
#> [6] " -2.718282" "-92.000000" "-0.001000" " 0.000000" " NA"
#> [11] " 0.001000" " 92.000000" " 2.718282" " 3141.592654" " 1.2e+09"
#> [16] " 3.51e+23" " 6.72e+120" " NaN"
formatFix(c(-rev(xxbig), -rev(xx), 0, NA, xx, xxbig), 6, 3, FALSE)
#> [1] " NaN" "-6.72e+120" "-3.51e+23" "-1.2e+09" "*****"
#> [6] " -2.718282" "-92.000000" "-0.001000" " 0.000000" " NA"
#> [11] " 0.001000" " 92.000000" " 2.718282" "*****" " 1.2e+09"
#> [16] " 3.51e+23" " 6.72e+120" " NaN"
```

frac

Fractional part of number

Description

Split off fractional part of a number

Usage

```
frac(x,d)
```

Arguments

x	Numerical vector.
d	If not missing, determines number of decimals after "."

Value

fractional part, if d is missing; else $round(10^d * fractionalpart)$, i.e. the digits without the leading "0".

Note

d not missing is practical for use in [dc](#)

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
frac(c(0,pi,2*pi,30*pi)) # 0.000000 0.141593 0.283185 0.247780
frac(c(0,pi,2*pi,30*pi),3) # 0 142 283 248
```

heading

Write a line of text with underlining and blank lines

Description

heading writes m blank lines, then text, underline with length(text) copies of ch then write n blank lines comm can be used as comment character

Usage

```
heading(text, comm = "", ch = "-", m = 1, n = 0)
```

Arguments

text	(one line) text string, if "": only m+n empty lines
comm	optional comment character written at start of text and of underline
ch	character used to underline text, if "": no underline
m	number of empty lines written before text
n	number of empty lines written after underlined text

Value

NULL

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
## "|" means the left margin
heading("Very compact", "", "=", 0, 1)
##|
heading("", "=") # one blank line only
##|
heading("standard")
##|
##|standard
##|-----
heading("Just a line, not really a heading", "", "", 0, 0)
##|Just a line, not really a heading
```

interpol

*Polynomial and rational interpolation***Description**

Determine the argument of the minimum by polynomial or rational interpolation of given points

Usage

```
setupInterp(x, y, doPoly = TRUE)
evalInterp(xi, ss)
minInterp(x, y, add = FALSE, doPoly = TRUE)
quadmin(x, y)
```

Arguments

x	vector of x-coordinates
y	vector of y-coordinates
xi	argument x of interpolation
ss	setup given by setupInterp
add	if TRUE one more point is used than for FALSE (default)
doPoly	if TRUE polynomial interpolation is used, if FALSE rational interpolation is used, with three points and four points respectively (for add=FALSE)

Value

xmin x-value of the minimum. NA if too few points are given or no minimum exists in
in

Note

There is a test program "InterpPR.pdf" available

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

References

Stoer, J., 1989. Numerische Mathematik 1ed. 5. Springer, Berlin. Applied and Computational Complex Analysis, Vol.2. Wiley,

Examples

```
x <- c(1,2,4,6)
minInterp(x, (x-3)^2, add=FALSE, doPoly=TRUE)
minInterp(x, (x+1.0/x), add=FALSE, doPoly=FALSE)
minInterp(x, (x+1.0/x), add=TRUE, doPoly=TRUE)

##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.

## The function is currently defined as
function( x, f, add = FALSE, PolyInt=TRUE ) {
  # Interpolate (PolyInt=TRUE) three points (Newton)
  # or four points (x,f) (Thiele),
  # For add=TRUE one more point is used.
  # Returns the numerical minimum and the explicit interpolant,
  # or NA if too few points are given.
  ks <- sort.list(x)
  x <- x[ks]; f <- f[ks]
  nn <- length(x)
  km <- which.min(f)
  bm <- 4+add-PolyInt
  if (nn < bm) {
    if (add && nn == bm-1) { # add == FALSE would also work
      add <- FALSE
      bm <- bm-1
    } else {
      return( list(xmin=NA, int=list(fct=NA, coef=NA, xx=NA)) )
    }
  }
  if (nn >= bm) {
    k <- max(1, min(km-round(bm/2), nn-bm+1))
    x <- x[k:(k+bm-1)]; f <- f[k:(k+bm-1)]
  }
}
```

```

s <- SetupIntp( x, f, PolyInt )
if ( PolyInt ) { # Newton
  if (add==0) sq[4] <- 0.0 # not needed: s$x[3] <- 0.0
  a <- s$x[1] + s$x[2]
  b <- a + s$x[3]
  c <- (s$x[2] + s$x[3])*s$x[1] + s$x[2]*s$x[3]
      # coefficients for quadratic equation for minimum
  A <- 3*sq[4] # coeff of x^2
  B <- (s$q[3] - s$q[4]*b)*2 # coeff of x
  C <- s$q[2] - a*s$q[3] + c*s$q[4] # absolute term
  D <- ((-s$q[4]*s$x[3] + s$q[3])*s$x[2] - s$q[2])*s$x[1] + s$q[1]
  coef <- c(A/3, B/2, C, D) # coefficients for interpolant
  fct <- function(co,x) (co[1]*x^3 + co[2]*x^2 + co[3]*x + co[4])
} else { # Thiele
  z0 <- s$q[bm]; z1 <- z2 <- n1 <- n2 <- 0; n0 <- 1;
  for (ii in ((bm-1):1)) {
    sq <- s$q[ii]; sx <- s$x[ii]
    if (abs(z0) < cMaxRealBy38) {
      Z0 <- sq*z0 - sx*n0
      Z1 <- sq*z1 - sx*n1 + n0
      Z2 <- sq*z2 - sx*n2 + n1
      # not needed z3 <- n2
      n0 <- z0; n1 <- z1; n2 <- z2
      z0 <- Z0; z1 <- Z1; z2 <- Z2
    } else { # "restart"
      z0 <- sq; z1 <- z2 <- n1 <- n2 <- 0; n0 <- 1;
    }
  }
}
# coefficients for quadratic equation for minimum
A <- z2*n1 - z1*n2 # coeff of x^2
B <- 2*(z2*n0 - z0*n2) # coeff of x
C <- z1*n0 - z0*n1 # absolute term
coef <- c(z2,z1,z0,n2,n1,n0)
cc <- coef[1]
if (abs(cc) > 1000.0) coef <- coef/cc
fct <- function(co,x)(co[1]*x^2+co[2]*x+co[3])/(co[4]*x^2+co[5]*x+co[6])
}

xmin <- sort(solveQeq( A, B, C ), na.last=TRUE)
if (!inrange(Re(xmin[1]),x)) {
  xa <- xmin[1]; xmin[1] <- xmin[2]; xmin[2] <- xa
}
if (!PolyInt && abs(n2)/16+1.0 == 1.0) { # high chance of unreachable points
  if ( abs(n1)/16+1.0 == 1.0) { # constant denominator
    xmin <- sort(solveQeq( z2, z1, z0 ), na.last=TRUE)
    coef <- c(z2,z1,z0)
    fct <- function(co,x)(co[1]*x^2+co[2]*x+co[3])
  } else {
    xn <- sort(solveQeq( 0.0, n1, n0 ), na.last=TRUE)[1]
    if (any(abs(xmin-xn)/16+1.0 == 1.0)) { # common zeros
      xmin <- NA # of numerator and denominator
      coef <- c(z2/n1,z0/n0)
    }
  }
}

```

```
      fct <- function(co,x)(co[1]*x+co[2])
    }
  }
}
return( list(xmin=xmin, int=list(fct=fct, coef=coef, xx=s$x)) )
} ## minIntp
```

invgauss

Inverse Gaussian Distribution

Description

Density, cumulative probability, quantiles and random generation for the inverse Gaussian distribution.

See full document: <http://www.maths.uq.edu.au/~gks/s/invgauss.html>

Author(s)

Gordon Smyth, <gks@maths.uq.edu.au>

See Also

<http://www.maths.uq.edu.au/~gks/s/qres.html>

is.constant

is.constant

Description

A numerical vector consists only of identical values

Usage

```
is.constant(x)
```

Arguments

x a vector

Value

TRUE if x is numerical and $\max(x) = \min(x)$

Author(s)

Kjetil Brinchmann Halvorsen, <kjetil@accelerate.com>, expanded by Christian W. Hoffmann <christian.hoffmann@wsl.ch>

See Also

[identical](#), [all.equal](#)

Examples

```
is.constant(rep(c(sin(pi/2),1),10)) # TRUE
x <- factor(c(1,1,NA))
is.constant(x) # FALSE because of NA
is.constant(x[1:2]) # TRUE
is.constant(c(1,1,NA)) # FALSE because of NA
is.constant(c(1,1,2)) # FALSE
is.constant(c(1,1,1)) # TRUE
```

jitterNA

Jitter entities with NA

Description

Extension of [jitter](#) to deal with NA entries

Usage

```
jitterNA(x,...)
```

Arguments

x Data to be jittered, may be vector, matrix, or numerical data frame.
... Other parameters for [jitter](#).

Value

jitterNA(x, ...) return a numeric vector with jittered entries, NA entries are allowed and not changed

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
d <- data.frame(cbind(x=1, y=1:10))
d[5,1] <- d[3,2] <- NA
jitterNA(d)
```

lengths.angle *Lengths of two vectors and angle between them.*

Description

Compute lengths of two vectors and the angle between them.

Usage

```
lengths.angle(x, y)
```

Arguments

x the first vector. Alternatively, a single matrix with both vectors as *columns* can be provided.

y the second vector, *optional* if **x** is an appropriate structure.

Value

A list with elements

lx the length of **x**.

ly the length of **y**.

angle the angle (in radian) between **x** and **y**.

angleDeg the angle (in degrees) between **x** and **y**.

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
x <- c(1,2,2,-1)
y <- c(0,0.1,3,2)
(l <- lengths.angle(x, y)) # == lengths.angle(cbind(x, y))
#> $lx    #> [1] 3.16228
#> $ly    #> [1] 3.60694
#> $angle #> [1] 1.1937
l$angle/pi*180 # in degrees > 68.394
```

libs	<i>List all installed packages, or all functions in a package</i>
------	---

Description

Lists all packages (called without an argument) or the functions in a package (called with the package name - quotes not needed).

Usage

```
libs(Lib)
```

Arguments

Lib package name, if missing see above

Author(s)

??

Examples

```
libs()
libs(base)
```

like	<i>Prepare new data for prediction</i>
------	--

Description

Prepare data to conform to structure of data used in a model, for use as new data

Usage

```
like(X, ...)
```

Arguments

X Data used in linear model like lm
 ... List of other arguments to model to be used for prediction

Value

Data usable for predict which are now compatible, all factors have the same level set, and so forth.

Author(s)

Peter Dalgaard, <p.dalgaard@biostat.ku.dk>

Examples

```
## Not run:
foo <- lm(formula, data=foe)
fie <- like(foe, sex="F", age=20:25, smoke="1-7")
predict(foo, newdata=fie)

## End(Not run)
```

lowess.bygroup

Plot data in groups, each with lowess or loess smoothing

Description

data in groups (shown by variable group) are plotted

Usage

```
lowess.bygroup(x, y, group, lin=FALSE, col = par("col"), bg = NA,
  pch = par("pch"), cex = 1, ...)
loess.bygroup(x, y, group, lin=FALSE, col = par("col"), bg = NA,
  pch = par("pch"), cex = 1, ...)
```

Arguments

x, y	coordinate vectors of equal length
group	grouping variable, must be a vector of same length as x and y
lin	If TRUE, then lines connecting the points (x,y) are drawn
cex	see par
col	colour of lines
bg	background colour
pch	character to plot
...	see pairs

Value

The procedure is called for its side effect of producing a plot

Author(s)

Christian W. Hoffmann, <c-w.hoffmann@sunrise.ch>

lpr *Print an object or plot*

Description

If an object name is given, that object will be printed, otherwise the current plot will be printed (assuming a Postscript printer)

Usage

```
lpr(object, file="Rplotlpr.ps", ...)
```

Arguments

object	The object to be printed.
file	A file name. If missing, the current plot will be printed.
...	Additional parameters for dev.copy .

Author(s)

Ray Brownrigg <ray@mcs.vuw.ac.nz>

ls.functions *List available functions*

Description

Returns a list of all the functions in the current work space.

Usage

```
ls.functions()
```

Author(s)

?

mult.fig.p

Plot Setup for MULTiple FIGures, incl. Main Title

Description

Easy Setup for plotting multiple figures (in a rectangular layout) on one page. It allows to specify a main title, a bottom line, and uses *smart* defaults for several `par` calls.

Usage

```
mult.fig.p(nr.plots, mfrow, mfcoll,
           marP = rep(0, 4), mgp = c(1.5, 0.6, 0),
           mar = marP + 0.1 + c(4, 4, 2, 1),
           main = NULL, sub = NULL, adj.sub = 0.5,
           tit.wid = if (is.null(main)) 0 else 1 + 1.5*cex.main,
           quiet = .Device == "postscript",
           cex.main = par("cex.main"),
           col.main = par("col.main"),
           font.main = par("font.main"), ...)
```

Arguments

<code>nr.plots</code>	integer; the number of plot figures you'll want to draw.
<code>mfrow</code>	<i>instead of nr.plots</i> : integer(2) vector giving the rectangular figure layout for <code>par(mfrow= .)</code>
<code>mfcoll</code>	<i>instead of nr.plots</i> : integer(2) vector giving the rectangular figure layout for <code>par(mfcoll= .)</code>
<code>marP</code>	numeric(4) vector of figure margins to <i>add</i> ("Plus") to default <code>mar</code> , see below.
<code>mgp</code>	argument for <code>par(mgp= .)</code> with a smaller default than usual.
<code>mar</code>	argument for <code>par(mar= .)</code> with a smaller default than usual, using the <code>marP</code> argument, see above.
<code>main</code>	character. The main title to be used for the whole graphic.
<code>sub</code>	character. The bottom line to be used for the whole graphic.
<code>adj.sub</code>	The value of <code>adj</code> determines the way in which <code>sub</code> is justified. A value of 0 produces left-justified text, 0.5 centered text and 1 right-justified text. See <code>par(adj= .)</code>
<code>tit.wid</code>	numeric; the vertical width to be used for the main title.
<code>quiet</code>	Suppress request to restore graphical parameters.
<code>cex.main</code>	numeric; the character size to be used for the main title.
<code>col.main</code>	string; name of the color to be used for the main title.
<code>font.main</code>	numeric; number of the font to be used for the main title.
<code>...</code>	Further arguments to <code>mtext</code> for <code>main</code> and <code>sub</code> .

Value

A `list` with two components that are lists themselves, a subset of `par()`,

`new.par` the current par settings.
`old.par` the par *before* the call.

Author(s)

Martin Maechler, <maechler@stat.math.ethz.ch>,
 modified by Christian W. Hoffmann, <c-w.hoffmann@sunrise.ch>

See Also

[par](#), [layout](#).

Examples

```
AA <- mult.fig.p(5, main= "Sinus Funktionen verschiedener Frequenzen")
x <- seq(0, 1, len = 201)
for (n in 1:5)
  plot(x, sin(n * pi * x), ylab = "", main = paste("n = ",n))
par(AA$old.par)

rr <- mult.fig.p(mfrow=c(4,2), main= "Sinus Funktionen", cex = 1.5,
                marP = - c(0, 1, 2, 0))
for (n in 1:8)
  plot(x, sin(n * pi * x), type = 'l', col="red", ylab = "")
str(rr)
par(rr$old.par)
## Look at the par setting *AFTER* the above:
str(do.call("par", as.list(names(rr$new.par))))
```

my.table

Tabulate data, with extra rows and columns.

Description

`my.table.NA` tabulates a vector of values and lists NA and NaN at the beginning, if they occur.
`my.table.margin` generates contingency table together with both margins of two factors, or of a matrix, if only one parameter is given.

Usage

```
my.table.NA(x, relative=FALSE)
my.table.margin(v,w)
```

Arguments

x	A vector, will be converted to factors.
relative	= TRUE if relative values should be returned.
v	factor or matrix.
w	factor.

Value

A contingency table.

Note

Uses [table](#).

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>
and John Fox <jfox@mcmaster.ca> (my.table.margin)

Examples

```
x <- c(1,NA,2,5,-1:7)
my.table.NA(x)
f1 <- sample(1:5,100,replace=TRUE)
f2 <- sample(1:5,100,replace=TRUE)
my.table.margin(f1,f2)
my.table.margin(matrix(1:24,4))
```

n22dig

Show vector or matrix (of $0 \leq x \leq 1$) in a compact way

Description

n22dig shows "0.ab" as "ab", "1.00" as "I", "0" as "0".

Usage

```
n22dig(x, symm = TRUE)
```

Arguments

x	A numerical vector or matrix.
symm	If symm = TRUE then upper triangle will be suppressed.

Value

Representation of x as two-digit vector or matrix.

Note

A violation of the condition on `abs(x)` will not be signalled. Empty places due to `symm = TRUE` are filled with " ".

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

See Also

[n2c](#).

Examples

```
n22dig(cor(matrix(rnorm(100),10)),TRUE)
#      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
# [1,] " I" " " " " " " " " " " " " " " " " " " " " "
# [2,] "10" " I" " " " " " " " " " " " " " " " " " " "
# [3,] " 8" "26" " I" " " " " " " " " " " " " " " " "
# [4,] " 8" "49" " 2" " I" " " " " " " " " " " " " " "
# [5,] " 8" "22" " 9" "46" " I" " " " " " " " " " " " "
# [6,] "40" "26" " 5" "27" "14" " I" " " " " " " " " " "
# [7,] " 8" "15" "21" "58" "13" "26" " I" " " " " " " " "
# [8,] "13" "30" " 2" "58" "21" "41" "61" " I" " " " " " "
# [9,] "46" "22" " 7" "63" "15" "25" "43" "36" " I" " " "
# [10,] "66" "51" "48" "16" "20" "27" "28" "20" "16" " I"
```

n2c

Show absolute values as characters, prepare for plotting

Description

`n2c` takes a numerical vector or matrix and represents it as single characters, with legend. `indexLine` generates a string with dots, ";", and digits, usable as x-label in `n2cCompact`: `.....;....1.....;....2..`. `n2cCompact` combines `n2c` and `indexLine` to generate a vector of strings good for printing numerical matrices. `charMat` processes the output from `n2cCompact` and returns vectors `x`, `y`, `tx` of equal lengths for input to [pltCharMat](#)

Usage

```
n2c(x, symm = FALSE)
indexLine(n)
n2cCompact(x, symm=FALSE)
charMat(cc)
```

Arguments

x	A numerical vector or matrix.
symm	If symm = TRUE then upper triangle will be suppressed.
n	integer, length of string wanted
cc	output from n2cCompact, input to charMat

Value

n2c: Representation of x as a single-character vector or matrix, as explained in *attribute* legend.
charMat: list(x,y,txt)

Note

Empty places due to symm = TRUE are filled with " ".

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

See Also

[n22dig](#).

Examples

```
n2c(c(10e20, -10e5, 20, 10, 0.9, -0.7, 0.6, 0, -0.5, 0.1))
n2c(matrix(c(10e20, 10e5, 20, 10, 0.7, 0.6, 0, 0.5, 0.1), 3, 3), FALSE)
#      [,1] [,2] [,3]
# [1,] "x"  "1"  " "
# [2,] "5"  "#"  "="
# [3,] "1"  "*"  ", "
# attr(,"legend")
# [1] ">=1: log, >=0. 9& 8% 7# 6* 5= 4+ 3- 2: 1, 05. ' ' "
m <- matrix(rnorm(500), nrow=50, ncol=10)
n2c(m, symm=TRUE)
indexLine(ncol(m))
(n2 <- n2cCompact(m, symm=FALSE))
charMat(n2)
```

napply

Apply a function to the corresponding elements of two lists (?)

Description

lapply extended for two lists ??

Usage

```
napply(...,FUN)
```

Arguments

```
...          list of vectors ??  
FUN         function to be used
```

Value

list of values

Author(s)

Peter Dalgaard, <p.dalgaard@biostat.ku.dk>

Examples

```
l1 <- 2  
l2 <- 4  
napply(l1,l2,FUN=union)
```

negbingof

Approximate a Negative binomial distribution.

Description

neg.bin.gof determines negative binomial distribution parameter 'k' by successive approximation

Usage

```
neg.bin.gof(data)
```

Arguments

```
data          vector of data.
```

Value

Prints statistics of approximated distribution.

Author(s)

?

normalize	<i>Normalize vectors</i>
-----------	--------------------------

Description

Normalize the columns of a matrix.

Usage

```
normalize(x)
```

Arguments

x the matrix containing the vectors to be normalized as *columns*.

Value

The matrix containing the normalized vectors as *columns*. If you want to normalize the *rows* of a matrix, use `t(normalize(t(x)))`

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
x <- matrix(rnorm(30),nrow=6)
normalize(x)
t(normalize(t(x))) # normalize the *rows* of x
```

num.ident	<i>Check numerical values for identity</i>
-----------	--

Description

Check two variables on numerical identity or whether both are either NaN or NA.

Usage

```
num.ident(x,y)
```

Arguments

x, y Variables to check for identity, may be arrays.

Value

TRUE or FALSE

Note

No check is made whether x or y are numeric

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
xxxx <- c(100,-1e-13,Inf,-Inf, NaN, pi, NA)
names(xxxx) <- formatC(xxxx, dig=3)
(aaaa <- outer(xxxx,xxxx,function(x,y) num.ident(x,y)))
all((aaaa & !is.na(aaaa)) == (row(aaaa) == col(aaaa)))
# aaaa has TRUE only on the diagonal, i.e. identity works correctly
```

num2Latex

Convert numeric containing e+-power

Description

Latex string with power notation

Usage

```
num2Latex(x, digits = 0)
```

Arguments

x numerical vector
 digits digits to show, see also [options](#) scipen

Value

Vector of strings representing the given numbers, $x \cdot 10^{-y}$ -> $x \cdot 10^{-y}$

Author(s)

<dimitris.rizopoulos@med.kuleuven.be>

Examples

```
z <- c(1.5, 5e-12, 2.33e-03, 8.12e+10, 2)
num2Latex(z)     # 1.5 # 5 \cdot 10^{-12} # 0.00233 # 8.12 \cdot 10^{10} # 2
num2Latex(z, 2) # 1.5 # 5 \cdot 10^{-12} # 2.33 \cdot 10^{-3} # 8.12 \cdot 10^{10} # 2
num2Latex(z, -3) # 1.5 # 5 \cdot 10^{-12} # 0.00233 # 81200000000 # 2
```

numberof	<i>Count the number elements that satisfy a condition.</i>
----------	--

Description

numberof counts the number elements that satisfy a condition.

Usage

```
numberof(x, f)
```

Arguments

x	Numerical array.
f	Logical function emulating the condition to be satisfied.

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
numberof(c(1:100,NA,NA,NaN),function(x) !is.na(x))
```

numericString	<i>Test string for being a number or made up of digits only, convert to bases.</i>
---------------	--

Description

Functions for testing strings and for conversion of integers to bases other than decimal as string representations.

Usage

```
numericString(str)
allDigits(str)
intToASCII(i)
intToBase(i,Base=2)
intToOct(i)
intToHex(i)
```

Arguments

str	A character vector.
i	Integer number to be converted.
Base	Number base to be converted to.

Details

numericString Test whether the elements of a character vector represent legal numbers only.
 allDigits Test whether the elements of a character vector consist of digits only. intToASCII
 Show character or octal representation at a place in the ASCII sequence. intToBase Convert an
 integer number to string representation in a base between 2 and 16 inclusive. intToOct Convert
 an integer number to string representation in octal. intToHex Convert an integer number to string
 representation in hexadecimal.

Value

TRUE, FALSE, string representations

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
allDigits(c("1231", "89a8742")) # TRUE FALSE
numericString(c("1231", "8.9e-2", ".7d2")) # [1] TRUE TRUE FALSE
intToASCII(1:255)
sapply(1:50, intToBase, 2)
sapply(1:50, intToBase, 7)
sapply(1:50, intToOct)
sapply(1:50, intToHex)
```

p.screepLOT.princomp *Plot screepLOT*

Description

p.screepLOT.princomp plots a screepLOT with labels of cumulative variances

Usage

```
p.screepLOT.princomp(object, variables=seq(length(object\$sdev)), cumulative=TRUE, main=deparse(subs
```

Arguments

object	an object of class "princomp" created by princomp
variables	a vector of integers selecting the elements of <code>\$object\\$sdev^2\$</code> .
cumulative	a logical value. If FALSE, the labels on bars show variances. If TRUE, they show cumulative variances.
main	overall title for the plot.
ylim	limits for the y axis.
...	optional arguments for barplot

Value

A numeric vector (or matrix, when `beside = TRUE`), say `mp`, giving the coordinates of all the bar midpoints drawn, useful for adding to the graph.

Author(s)

Christian W. Hoffmann, <c-w.hoffmann@sunrise.ch>

Examples

```
library(stats)
m <- cbind(rnorm(100),rlnorm(100))
p.screepplot.princomp(princomp(m))
## Not run: p.screepplot.princomp(princomp(m),main="Screepplot",ylab="Variances",cex=0.6)
```

padding

Padding a string with justification

Description

`padding` Pads a string.

Usage

```
padding(str, space, with, to=c("left","right","center"))
```

Arguments

<code>str</code>	String to be padded.
<code>space</code>	Resulting length of padded string.
<code>with</code>	String to pad with. Will be repeated as often as necessary.
<code>to</code>	Mode of padding, one of "left", "right", "center".

Value

A string

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
padding("My string",25,"XoX","center")
# [1] "XoXXoXXoMy stringXXoXXoXX"
```

panel *Alternative panel functions for lattice plots*

Description

Functions which can be used instead of the default functions in panel plots.

Usage

```
panel.hist(x, ...)
panel.cor(x, y, digits=2, prefix="", cex.cor)
```

Arguments

x, y	variables defining the contents of the panel.
digits	Number of decimals after dot with which correlations will be printed.
prefix	Prefix text for numbers.
cex.cor	Determines height of printed digits, may be missing.
...	graphical parameters can be supplied. see function definition for details.

Author(s)

?? <>

Examples

```
n <- 1000; a <- rnorm(n,mean=1)
x <- matrix(c(a,a+2*log(runif(n)),a^2+0.2*rnorm(n,mean=1)),nrow = n)
pairs(x,lower.panel=panel.smooth, diag.panel=panel.hist,
upper.panel=panel.cor, labels = c("rnorm","rnorm+log(runif)","rnorm^2"))
```

pasteInfix *Paste(infix)*

Description

Paste as infix

Usage

```
a %&& b
```

Arguments

a	an R objects, to be coerced to character vectors.
b	an R objects, to be coerced to character vectors.

Value

The concatenation of a and b, same as `paste(a, b, sep="")`

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

See Also

String manipulation with [paste](#), [as.character](#), [substr](#), [nchar](#), [strsplit](#); further, [cat](#) which concatenates and writes to a file, and [sprintf](#) for C like string construction.

Examples

```
"I am" %% " hungry" # [1] "I am hungry"
```

pasteRound

Paste rounded values

Description

Paste rounded values

Usage

```
pasteRound(..., digits=16, sep=" ", collapse=NULL)
```

Arguments

<code>...</code>	list of arguments to be pasted.
<code>digits</code>	argument to round .
<code>sep</code>	argument to paste .
<code>collapse</code>	argument to paste .

Value

The concatenation of formatted values

Author(s)

Dimitris Rizopoulos <dimitris.rizopoulos@med.kuleuven.ac.be>

Examples

```
x <- rnorm(3)
x
pasteRound("x1=", x[1], ", x2=", x[2], ", x3=", x[3], sep="", collapse="")
pasteRound("x1=", x[1], ", x2=", x[2], ", x3=", x[3], digits=3, sep="", collapse="")
```

persp2

*Central perspective from 3 to 2 dimensions along specified axis***Description**

Transform 3-dimensional data by central perspective to plane coordinates.

Usage

```
persp2(d,axis,r)
```

Arguments

d	Data matrix, one row is one point (x,y,z).
axis	One of 1, 2, 3 to name axis of perspective.
r	The eye point lies at (axis = r, 0, 0).

Value

The matrix d with column axis unchanged, the other two columns replaced by their perspective transformation.

Note

To avoid ambiguity the column d[, axis] is retained.

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
par(mfrow=c(1,1))
x <- c(0,5,NA,0,0,NA,0,0)
y <- c(0,0,NA,0,6,NA,0,0)
z <- c(0,0,NA,0,0,NA,0,4)
d <- cbind(x,y,z)
dr <- d %*% rotm(3,1,3,pi/5) %*% rotm(3,1,2,-pi/4)
D <- persp2(dr,1,2)
dx <- D[1:2,]; dy <- D[4:5,]; dz <- D[7:8,]
plot(0,xlim=range(D[!is.na(D[,2]),2])*1.3,ylim=range(D[!is.na(D[,3]),3])
     *1.3,type="n",xlab="",ylab="")
lines(dx[,2],dx[,3],col="red")
lines(dy[,2],dy[,3],col="green")
lines(dz[,2],dz[,3],col="blue")
```

`plotSymbols`*Plot symbols, colours, and allow to choose*

Description

A plot of symbols is generated. By clicking the mouse on a symbol the numeric codes are given in ASCII, octal, hex. Plot symbols depending on font.

Usage

```
plotSymbols(interactive=FALSE)
availColors(indx = 0:6)
plotSymbolsFonts(fn=1)
```

Arguments

<code>interactive</code>	allow choice of symbols
<code>indx</code>	indices of panels showing 100 colours each
<code>fn</code>	a font number 1 ... 5

Value

list of	
<code>ch</code>	character value of symbol
<code>dec</code>	decimal value of symbol
<code>hex</code>	hex value of symbol
<code>oct</code>	octal value of symbol

Note

To turn off the click-bell do `'options(locatorBell=FALSE)'` (see `?locator`).

Author(s)

Henrik Bengtsson <hb@maths.lth.se>, adapted by Christian W. Hoffmann, <c-w.hoffmann@sunrise.ch>

Examples

```
plotSymbols(3)
```

plt *Plot depending on switch, Create multiple plots with title and time stamp*

Description

- pltCharMat uses output from [charMat](#) to plot numerical matrices as characters. - pltRCT executes a (series of) plotting function(s) under the control of some useful switches, may be useful in [source](#).
 - histRCT creates a (series of) histogram(s), uses pltRCT. - SplomT creates a scatterplot matrix,

Usage

```
pltCharMat(m,tit)
pltRCT(rows, cols, tit="", f = function(x) 0, cex = 1.5,
       reset = TRUE, outer = TRUE, oma = c(2, 2, 4, 2), mar = c(4, 4, 2, 1))
histRCT(data, rows = round(sqrt(ncol(data))),
        cols = ceiling(ncol(data)/rows), breaks = "Sturges", mainL = deparse(substitute(data)), mainC = coln
SplomT(data, mainL = deparse(substitute(data)), xlabL = "",
       hist = "h", adjust = 1, hist.col = trellis.par.get("strip.background")$col[5], cex.diag = 1, h.diag=
```

Arguments

m	Numerical matrix
tit	Overall title for plot. A vector of one or two elements. If an element is an expression , plotmath will be used.
rows	Number of rows of panels
cols	Number of columns of panels
f	A function to plot the individual plot panels. It can also be a statement sequence {...}.
cex	Font size used for tit
reset	Should previous rows, cols be restored after execution. See note
outer	Passed on to mtext.
oma	Outer margin used in initial par(...).
mar	Lines of margin used in initial par(...).
data	Matrix or dataframe containing data, variables in columns
breaks	Breaks for histogram
mainL	Label on top of scatterplot matrix or matrix of histograms
mainC	Labels on top of each of the histograms, should be character vector of length = number of columns of data
xlabL	Label for x axis
hist	"h" = histogram, "d" = density curve, "b" = both
adjust	factor to adjust smoothing window for density curve

hist.col	colour for the bars of the histograms
cex.diag	correction factor for font height of correlations and names in the diagonal
h.diag	placement of the variable name in the diagonal panel, =0 means on the lower border, = 0.5 in the middle between lower and upper border
colYonX, colXonY	colour of smoothing lines, y on x and x on y
...	Parameters passed on to upper.panel,lower.panel,diag.panel

Value

These functions are called for their side effect to produce a plot.

WARNING

The sequence of functions contained in `f` MUST NOT contain any call to `postscript`, because this would try to open another ps device without closing the old one!

Note

`oldpar <- par(mfrow = c(rows, cols), oma=oma, mar=mar)` is called at the beginning of `pltRCT`. Uses `spIom`, `[lattice:extend.limits]extend.limits`, and `datetime` .

If you have `n` panels you want to plot in a nearly quadratic arrangement, use `rows = round(sqrt(n))`, `cols=ceiling(n/rows)` (tending to slightly "landscape"). This is very similar to `n2mfrow`. `histRCT` drops columns with less than 2 legal (non-NA) values. For empty matrices no plot will be generated.

Author(s)

Christian W. Hoffmann, <c-w.hoffmann@sunrise.ch>, with the assistance of Deepayan Sarkar <deepayan@cs.wisc.edu>.

Examples

```
## Not run:
x <- rnorm(100); y <- rnorm(100)+1; z <- y+rlnorm(100)
pltRCT(1,1,f={plot(x,y,xlab="this is my x");abline(reg=lm(y~x),lty=2);plot(x,z,pch=3)})
nr <- 100; nc <- 8;
data <- as.data.frame(matrix(rnorm(nr*nc),nrow=nr,ncol=nc))
data[,nc] <- data[,nc-2] + 0.3*data[,nc-1] #generate higher correlations
data[,nc-1] <- data[,nc-1] + 0.9*data[,nc]
colnames(data)<-paste("vw", letters[1:nc], sep="")
SpIomT(data,mainL="",hist="d",cex.diag=0.6,hist.col="green")
SpIomT(data,mainL="",hist="b",adjust=0.4,cex.diag = 0.5)
pltHist(data)
pltRCT(1, 1, tit="1 by 2 plot", f=plot(y,x-3*y) )
pltRCT(1, 2, f={plot(x,y,xlab="my x"); abline(reg=lm(y~x),lty=2);plot(x,z,pch=3)})
m <- matrix(rnorm(500),nrow=50,ncol=10)
pltCharMat(m,"Random example")

## End(Not run)
```

pointfit

*Least squares fit of point clouds, or the Procrustes problem.***Description**

Find a transformation which consists of a translation tr and a rotation Q multiplied by a positive scalar f which maps a set of points x into the set of points $xi : xi = fQx + tr$. The resulting error is minimized by least squares.

Usage

```
pointfit(xi,x)
```

Arguments

x	Matrix of points to be mapped. Each row corresponds to one point.
xi	Matrix of target points. Each row corresponds to one point.

Details

The optimisation is least squares for the problem $xi : xi = fQx + tr$. The expansion factor f is computed as the geometric mean of the quotients of corresponding coordinate pairs. See the program code.

Value

A list containing the following components:

Q	The rotation.
f	The expansion factor.
tr	The translation vector.
res	The residuals $xi - f * Q \%*\% x + tr$.

Author(s)

Walter Gander, <gander@inf.ethz.ch>,
<http://www.inf.ethz.ch/personal/gander/> adapted by Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

References

"Least squares fit of point clouds" in: W. Gander and J. Hrebicek, ed., Solving Problems in Scientific Computing using Maple and Matlab, Springer Berlin Heidelberg New York, 1993, third edition 1997.

See Also

`rotm` to generate rotation matrices (mathlib), `rotangle` to determine them from orthogonal matrix.

Examples

```

# nodes of a pyramid
A <- matrix(c(1,0,0,0,2,0,0,0,3,0,0,0),4,3,byrow=TRUE)
nr <- nrow(A)
v <- c(1,2,3,4,1,3,4,2) # edges to be plotted
# plot
# points on the pyramid
x <-
matrix(c(0,0,0,0.5,0,1.5,0.5,1,0,0,1.5,0.75,0,0.5,2.25,0,0,2,1,0,0),
      7,3,byrow=TRUE)
# simulate measured points
# thetar <- runif(3)
thetar <- c(pi/4, pi/15, -pi/6)
# orthogonal rotations to construct Qr
Qr <- rotm(3,1,2,thetar[3]) %*% rotm(3,1,3,thetar[2]) %*% rotm(3,2,3,thetar[1])
# translation vector
# tr <- runif(3)*3
tr <- c(1,3,2)
# compute the transformed pyramid
fr <- 1.3
B <- fr * A %*% Qr + outer(rep(1,nr),tr)
# distorted points
# xi <- fr * x + outer(rep(1,nr),tr) + rnorm(length(x))/10
xi <- matrix(c(0.8314,3.0358,1.9328,0.9821,4.5232,2.8703,1.0211,3.8075,1.0573,
0.1425,4.4826,1.5803,0.2572,5.0120,3.1471,0.5229,4.5364,3.5394,1.7713,
3.3987,1.9054),7,3,byrow=TRUE)
(pf <- pointfit(xi,x))
# the fitted pyramid
(C <- A %*% pf$Q + outer(rep(1,nrow(A)),pf$str))
(theta <- rotangle(pf$Q))
thetar
# as a final check we generate the orthogonal matrix S from the computed angles
# theta and compare it with the result pf$Q
Ss <- rotm(3,1,2,theta[3]) %*% rotm(3,1,3,theta[2]) %*% rotm(3,2,3,theta[1])
max(svd(Ss*pf$factor-pf$Q)$d) #> 4.84082e-16 but why pf$factor ??

```

poisgam

*Poisson Gamma Distribution***Description**

Density, cumulative probability function and random variates for the Poisson-gamma or compound Poisson distribution. The Poisson-gamma distribution is a Tweedie distribution with index p between 1 and 2.

See: <http://www.statsci.org/smyth/index.html>

Author(s)

Gordon Smyth, <gks@maths.uq.edu.au>

printP *Print without square brackets, expression values together with their call strings*

Description

These functions may be helpful for documenting ongoing work using `sink()`.

- `catn(...)` is shorthand for `cat(...); cat("\n")`
- `pn()` is shorthand for `cat("\n")` which is awkward for me to type.
- `prinV` prints a vector without `\[`, in fix format.
- `prinM` prints a matrix without `\[`, in fix format.
- `prinT` prints an array, TAB delimited.
- `prinE(xsv, ...)` prints a string expression and its evaluation in the form "`xsv = evaluation`", in vector form. The string may contain "'commenting text'; expression(s)", but only the last expression will be evaluated.
- `prinL(xs, ...)` prints a string expression and its evaluation in the form "`xs \newline evaluation`".
- `prinP(xs)` prints a string argument and evaluates it i.e. the body of the function evaluated should contain `print` and `cat` statements.
- The variants `N...` prepend a linefeed.

Usage

```
catn(...)
pn()
prinV(x,after=2,before)
prinM(x,after=2,before)
prinT(x,rownam=FALSE,colnam=FALSE)
prinE(xsv,...)
prinL(xs,...)
prinP(xs)
```

Arguments

<code>x</code>	A numerical vector or matrix.
<code>before</code>	See <code>formatFix</code> , the number of decimals before "."
<code>after</code>	See <code>formatFix</code> , the number of decimals after "."
<code>rownam</code>	Should row names be printed.
<code>colnam</code>	Should column names be printed.
<code>xsv</code>	A string representing a vector expression.
<code>xs</code>	A string representing an expression.
<code>...</code>	Additional parameters for <code>print</code> .

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```

xx <- options(digits=7)
x <- matrix(c(5,3,2,7,8.235,exp(1),pi,0,99),3,3)
m <- matrix(c("a","b c","d","ff"," x","","7","8","99"),3,3)
dimnames(x) <- list(c("r1","r2","r3"),c("c1","c2","c3"))

prinV(as.vector(x))
# 5.00 3.00 2.00 7.00 8.24 2.72 3.14 0.00 99.00

prinM(x,3)
# 5.00 7.00 3.14
# 3.00 8.24 0.00
# 2.00 2.72 99.00

prinT(x,TRUE,TRUE)
# c1 c2 c3
# r1 5 7 3.14159265358979
# r2 3 8.235 0
# r3 2 2.71828182845905 99

prinT(c(c1="a",c2="b c",c3="d",c4="ff",c5=" x"),TRUE)
# c1 c2 c3 c4 c5
# a b c d ff x # the tabs are not visible here

prinT(c(c1=5,c2=7,c3=1,c4=3),TRUE)
# 5 7 1 3 # the tabs are not visible here

####prinE("x")
# x =[1] 5.000000 3.000000 2.000000 7.000000 8.235000 2.718282 3.141593
# [8] 0.000000 99.000000

####prinE("'This is a comment: ';3+5;pi-3",digits=4)
# 'This is a comment: ';3+5;pi-3 =[1] 0.1416

prinL("x")
# x
# c1 c2 c3
# r1 5 7.000000 3.141593
# r2 3 8.235000 0.000000
# r3 2 2.718282 99.000000

catt <- function(x) {cat("This function will write '",x,'" on one line\n") }
y <- prinP("catt(32)");
# catt(32)
# This function will write ' 32 ' on one line

####prinE("y ")
# y =[1] "catt(32)"

prinL("y ")
# y

```

```
# [1] "catt(32)"

prinP("y ")
# y

options(digits=xx$digits)
```

progress.meter

Monitor the progress of a repetitive calculation.

Description

progress.meter writes a symbol to the output at each invocation. The symbol is usually a ".", a "+" if $i \% \% == 0$, and $(ii \% \vee \% 10) \% \% 10$ if $ii \% \% 10 == 0$. If $ii \% \% 50 == 0$, a line break will be written and i printed.

Usage

```
progress.meter(i)
```

Arguments

i An integer.

Value

invisible(NULL).

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
cat("\n 0")
for (ii in seq(500)) {
  # do something time consuming
  progress.meter(ii)
}
cat("\n")
```

qnorm.appr *Approximation to the inverse normal distribution function.*

Description

qnorm.apxx approximate the normal quantile function. They compute x such that $P(x) = \text{Prob}(X \leq x) = p$.

Usage

```
qnorm.app3(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
qnorm.app4(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
qnorm.ap16(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
```

Arguments

p	vector of probabilities.
mean	vector of means.
sd	vector of standard deviations.
log.p	logical; if TRUE, probabilities p are given as log(p).
lower.tail	logical; if TRUE (default), probabilities are $P(X \leq x)$, otherwise, $P(X > x)$.

Value

qnorm.apxx gives the quantile function.

Warning

If $p \leq 0$ or $p \geq 1$, then NA will be returned.

If p is very close to 1, a serious loss of significance may be incurred in forming $c := 1 - p$, resulting in $p = 0$. In this case c should be derived, if possible, directly (i.e. not by subtracting p from 1) and evaluate `qnorm(p, ..., lower.tail=B)` as `qnorm(c, ..., lower.tail = (B==FALSE))`.

Note

qnorm.ap16 is the approximation used in [qnorm](#). The others have an absolute error $< 10^{-3}$ and 10^{-4} .

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Source

qnorm.app3 and qnorm.app4: Abramowitz and Segun, Dover, 1968, formulae 26.2.22 and 26.2.23,
qnorm.ap16: Wichura, M. J., 1988. Algorithm AS 241: The Percentage Points of the Normal Distribution. Applied Statistics 37, 477-484.

Examples

```

prec <- function(x,y,z=y) max(abs((x-y)/z))
x2 <- -0.6744897501960817; p2 <- 0.25
x0 <- -3.090232306167814; p0 <- 0.001
xm <- -9.262340089798408; pm <- 1.0e-20
x <- c((-100:0)/10,x2,x0,xm)
p <- pnorm(x)
x3 <- qnorm.app3(p)
x4 <- qnorm.app4(p)
x1 <- qnorm.ap16(p)
prec(x,x3,1) # 0.002817442
prec(x,x4,1) # 0.0004435874
prec(x,x1,1) # 0.1571311
prec(x,qnorm(p),1) # 1.776357e-15
# Special values
prec(qnorm.app3(p2),x2) # 0.004089976
prec(qnorm.app3(p0),x0) # 0.0007736497
prec(qnorm.app3(pm),xm) # 7.29796e-06
prec(qnorm.app4(p2),x2) # 0.0004456853
prec(qnorm.app4(p0),x0) # 9.381806e-05
prec(qnorm.app4(pm),xm) # 4.151165e-05
prec(qnorm.ap16(p2),x2) # 0
prec(qnorm.ap16(p0),x0) # 2.874148e-16
prec(qnorm.ap16(pm),xm) # 0.01211545

```

qres

*Randomized quantile residuals***Description**

Computes randomized quantile residuals for binomial, Poisson, negative binomial, gamma and inverse Gaussian generalized linear models.

See full document: <http://www.maths.uq.edu.au/~gks/s/qres.html>

Author(s)

Gordon Smyth, <gks@maths.uq.edu.au>

See Also

- <http://www.maths.uq.edu.au/~gks/s/tweedie.html> Tweedie Distributions,
- <http://www.maths.uq.edu.au/~gks/s/tweedief.html> Tweedie family,
- <http://www.maths.uq.edu.au/~gks/s/poisgam.html> Poison-gamma Distribution,
- <http://www.maths.uq.edu.au/~gks/s/invgauss.html> Inverse Gaussian Distribution.

remove.dup.rows	<i>Remove duplicate rows</i>
-----------------	------------------------------

Description

Removes duplicate rows from a dataframe.

Usage

```
remove.dup.rows(dfr)
```

Arguments

dfr A dataframe

Details

Uses the function `eq1`.

Value

The dataframe with only one copy of identical rows.

Author(s)

Peter Dalgaard, <p.dalgaard@biostat.ku.dk>

Examples

```
dfr <- data.frame(matrix(c(1:3,2:4,1:3,1:3,2:4,3:5),6,byrow=TRUE))
remove.dup.rows(dfr)
```

replacechar	<i>Replace a character in a string by another</i>
-------------	---

Description

replacechar replaces a character in a string by another

Usage

```
replacechar(str, char = "_", newchar = ".")
```

Arguments

str	The string to be altered.
char	The character to be replaced.
newchar	The character to replace with.

Value

The altered string.

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
replacechar("my_queer_file,name") # "my.queer.file,name"
replacechar("my_queer_file,name","m","M") # "My.queer.file,naMe"
```

rotm	<i>Rotation matrices and compute rotation angles from orthogonal matrix</i>
------	---

Description

Generate a (square) rotation matrix. Decompose the orthogonal matrix Q into the product $R3 * R2 * R1$ where the Ri are the elementary rotations around the i -th axis.

Usage

```
rotm(n,x,y,phi)
rotangle(Q)
```

Arguments

n	Order of the square matrix.
x,y	Integers describing the plane of rotation.
phi	Angle (counted counter clockwise) of rotation of coordinate system.
Q	Orthogonal matrix.

Value

rotm: Matrix to use for pre-multiplying. rotangle: The vector containing the rotation angles [radian]

Author(s)

rotangle: Walter Gander, <gander@inf.ethz.ch>
<http://www.inf.ethz.ch/personal/gander/>
 adapted by: Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

References

"Least squares fit of point clouds" in: W. Gander and J. Hrebicek, ed., Solving Problems in Scientific Computing using Maple and Matlab, Springer Berlin Heidelberg New York, 1993, third edition 1997.

See Also

[pointfit](#), [rotm](#) for synthesizing such a matrix.

Examples

```
par (mfrow=c(2,2))
Data <- rbind(rnorm(100)*100,rnorm(100)*40,rnorm(100)*5,rnorm(100)*1,rnorm(100)*0.01)
Rotdata <- rotm(dim(Data)[1],1,2,pi/3)
Rotdata2 <- rotm(dim(Data)[1],2,4,pi/5)
plot(Data[1,],Data[2,])
plot(Rotdata[1,],Rotdata[2,],col="red")
points(Rotdata2[1,],Rotdata2[2,],col="blue")
plot(Rotdata2[2,],Rotdata2[3,],col="red")
plot(Rotdata2[4,],Rotdata2[3,],col="red")
rot <- matrix(c(0.847101, -0.480873, -0.226234, 0.489074, 0.538865,
0.685880, -0.207912, -0.691655, 0.691655),3,3,byrow=TRUE)
rotangle(rot) #> 0.785398 0.209440 -0.523599
rotangle(rot)/pi*180 #> degrees: 45 12 -30
```

 rtf

Rational Transfer Function objects for R

Description

no detqils given.

Usage

```
rtf(A = 1, B = 1, delay = 0, unit.sg = TRUE, stability.check = TRUE)
rtf.filter(x, rtfobj, init)
rtf.impulse(rtfobj, lag.max, plot.it=TRUE,
           nzero=2, type="h",
           xlab="Lag", ylab="Impulse Response",...)
rtf.step(rtfobj, lag.max, plot.it=TRUE,
        nzero=2, type="h",
        xlab="Lag", ylab="Step Response", ylim,...)
```

```
printrtf(x, digits,...)
plotrtf(x, lag.max,...)
```

Arguments

A	See details
B	See details
delay	delay
unit.sg	See details
stability.check	See details
x	Filter (print.rtf), or (plot.rtf) rational transfer function object as created by rtf()
rtfobj	rational transfer function object as created by rtf()
init	If initialization is needed 'init' is supplied to the recursive filter (the first)
lag.max	maximum lag
plot.it	if TRUE generate the plot
nzero	number of zeros
type	see plot.default
xlab	label at x-axis
ylab	label at y-axis
ylim	limit in y-direction
digits	for print
...	passed to rtf.impulse and rtf.step

Details

rtf: Creates and checks a rational transfer-function object

$y_t = H(q) x_t$, where $H(q) = q^{-\text{delay}} B(q^{-1}) / A(q^{-1})$

If unit.sg is TRUE (default) the coefs. of $B(q^{-1})$ is multiplied with a factor making the stationary gain one.

$A(q^{-1}) = 1 - a_1 q^{-1} - \dots - a_{na} q^{-na}$ $B(q^{-1}) = b_0 + b_1 q^{-1} + \dots + b_{nb} q^{-nb}$

Note that '-' is used in A() and '+' in B(), these are specified as:

$A = c(1, a_1, \dots, a_{na})$ $B = c(b_0, b_1, \dots, b_{nb})$

If stability.check is TRUE (default) the function will stop if any poles of A() is outside the unit circle.

rtf.filter: Filter x using a rational transfer function object (rtfobj) as created by rtf(). If initialization is needed 'init' is supplied to the recursive filter (the first).

Note that:

* 'init' is multiplied with the stationary gain of the recursive filter before it is applied, i.e. replaced by $\text{init}/A(1)$.

* First the series is filtered through $1/A(q^{-1})$, and the initialization is in terms of the output of this filter. Furthermore, 'init' is used to calculate the first value of the filtered series, i.e. 'init' corresponds to times 0, -1, -2, ...

* The causal convolution filter cannot return values for time $\leq \text{length}(B) - 1$, since it does not use initialization.

* Since the recursive filter is run first (and no missing is allowed in x) the bug in filter() when the series starts with NA will not become active.

rtf.impulse: Impulse response of rtfobj (one like the one created by rtf()), i.e. the response on a unit impulse corresponding to index 1 of the output.

rtf.step: Step response of rtfobj (one like the one created by rtf()) i.e. the response on a unit step corresponding to index 1 of the output.

Value

rtf:

call	Image of the call
stable	Logical indicating if the transfer-function is stable
sg	Stationary gain (only if stable is TRUE)
n.init	Number of initial values needed
A	A
B	B
delay	delay

Author(s)

Henrik Aalborg Nielsen, IMM, DTU, <han@imm.dtu.dk> adapted by Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

scode

Generate the significance codes as in summary.lm

Description

Generate the significance codes as in summary.lm

Usage

scode(p)

Arguments

p Probability

Value

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Note

lifted from stats::printCoefmat

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
for (ii in c(0.005, 0.02,0.0501,0.2)) { scode(ii) }
```

select.range

Select values from a vector depending on a range in a second vector.

Description

select.range accepts two vectors of paired observations and returns a vector of observations from data. The observations returned are those for which the paired values in groupvec are within the range specified by min and max. NOTE: The in-range condition is *greater than or equal to* min and *less than* max. This allows contiguous ranges to be specified without returning the same value in two sets.

Usage

```
select.range(groupvec, min, max, data)
```

Arguments

groupvec	a vector of observations to be used for grouping.
min	the minimum value of the range.
max	the maximum value of the range.
data	a numeric vector of observations.

Value

The subset of observations from data is returned invisibly.

Author(s)

?

Examples

```
testvec <-c(2,4,3,5,4,5,7,6,4,5,6,8,7,9,8)
agevec  <-c(10,13,14,25,29,32,34,45,48,55,62,67,69,70,74)
newvec  <-select.range(agevec,0,30,testvec)
newvec  # [1] 2 4 3 5 4
```

seqm	<i>sequences, empty if "by" not conforming</i>
------	--

Description

Generate sequences, but return NULL, when "seq" would generate an error. This function is useful for `for`-loops, when empty loops are required, if `by` is in the "wrong" direction, see *examples*.

Usage

```
seqm(from, to, by=1)
```

Arguments

<code>from</code>	starting value of sequence.
<code>to</code>	(maximal) end value of the sequence.
<code>by</code>	increment of the sequence.

Value

NULL, if $(to-from)*by < 0$, otherwise usual result of `seq` i.e. `seq.default`.

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
seqm(12,4,-1) # 12 11 10 9 8 7 6 5 4
seqm(12,4,2) # NULL
lo <- 1; up <- 3
for (ii in lo:up) {
  cat(ii, " ")
  for (kk in seqm(lo,ii-1)) {
    cat(" ",kk) # do-in-lower-triangle
  }
  cat(" diag") # do-something-on-the-diagonal
  for (kk in seqm(ii+1,up)) {
    cat(" :",kk) # do-in-upper-triangle
  }
  cat("\n")
}
```

```
# 1      diag : 2 : 3
# 2      1 diag : 3
# 3      1  2 diag
```

```
setPowerPointStyle      Set PowerPoint style
```

Description

Set par() with suitable options for Power point slides. Note that you have to set them every time you open up a new device. Then save the pictures to PNG format with 'width=5' and 'height=5' pointsize='14'.

Usage

```
setPowerPointStyle()
```

Author(s)

Henrik Bengtsson, <hb@maths.lth.se>

```
shapiro.wilk.test      Shapiro-Wilk Normality Test
```

Description

Performs the Shapiro-Wilk test for normality.

Usage

```
shapiro.wilk.test(x)
```

Arguments

x a numeric vector of data values, the number of which must be between 3 and 5000. Missing values are allowed.

Value

A list containing the following components:

W the value of the Shapiro-Wilk statistic.
n length(x)
p the p-value for the test.

Author(s)

??

See Also[shapiro.test](#) of `ctest`**Examples**

```
shapiro.wilk.test(rnorm(100, mean = 5, sd = 3))
shapiro.wilk.test(runif(100, min = 2, max = 4))
```

`signp`*Sign Function*

Description

`sign` returns a vector with the signs of the corresponding elements of `x` (the sign of a real number is 1, 1, or -1 if the number is positive, zero, or negative, respectively).

Note that `sign` does not operate on complex vectors.

Usage

```
signp(x)
```

Arguments

`x` a numeric vector

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

See Also[sign](#)**Examples**

```
signp(-2:3)# -1 -1 1 1 1 1
(m <- matrix(rnorm(9),3,3))
m %*% diag(signp(diag(m)))
```

`smoothed.df`*Fit cumulative distribution from kernel estimate.*

Description

Given a kernel density estimate, this function carries out a (very quick and dirty) numerical integration, and then fits a spline to get a function which can be used to look up cumulative probabilities.

Usage

```
smoothed.df(d)
```

Arguments

`d` kernel density estimate

Value

The spline function approximating the df.

Author(s)

Ross Ihaka, <ihaka@stat.auckland.ac.nz>

Examples

```
x <- rnorm(1000) + ifelse(runif(1000) > .5, -3, 3)
d <- density(x)
F <- smoothed.df(d) # F returns cumulative probs

# Plot the true (red) and estimated (blue) density functions
par(mfrow=c(1,2))
curve(0.5 * dnorm(x, -3) + 0.5 * dnorm(x, 3), -7, 7, col="red")
lines(d, col="blue")

# Plot the true (red) and estimated (blue) distribution functions
curve(0.5 * pnorm(x, -3) + 0.5 * pnorm(x, 3), -7, 7, col="red")
curve(F(x), add=TRUE, col="blue")
```

 solveQeq

Solve linear and quadratic equations.

Description

Compute the solution(s) avoiding cancellation in the real case.

Usage

```
solveQeq(A, B, C)
```

Arguments

A, B, C coefficients in $Ax^2 + Bx + C = 0$.

Value

The solution(s)

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
solveQeq(0,-1,-1) # -1
solveQeq(0,-1,0) # 0
solveQeq(0,0,0) # NaN
solveQeq(1,1,1) # -0.5-0.866025i, -0.5+0.866025i
solveQeq(1,-1,1) # 0.5+0.866025i, 0.5-0.866025i
```

 str2formula

Extract sides of a formula to strings, Convert back to a formula

Description

str2formula is the inverse function to formula2string. term.names2formula combines two vectors of strings into a formula. formula2string Returns the left and the right hand sides of a formula.

formula2term.names Returns one chosen side of a formula.

formula2Rterm.names Returns the right hand side of a formula.

Usage

```
str2formula(s)
term.names2formula(ls,rs)
formula2string(form)
formula2term.names(form,side)
formula2Rterm.names(form)
```

Arguments

s	A list(left,right) containing the string representation of the left and the right hand side of the formula (one string each).
ls	A character vector (usually of length 1) containing the names of the terms on the left hand side of the formula.
rs	A character vector containing the names of the terms on the right hand side of the formula.
form	a formula.
side	one of "left","right".

Value

str2formula: A formula. formula2string: Character vector containing the string representation of the formulas side(s).

Note

The inverse function to str2formula is [formula2string](#).
 The inverse functions to term.names2formula are [formula2term.names](#) and [formula2Rterm.names](#).
 Functions use strsplit

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
fo <- a ~ b + c
abc <- formula2string(fo) # $left: "a" $right: "b+c"
str2formula(abc) # a ~ b + c
term.names2formula(formula2term.names(fo,"left"),c(formula2Rterm.names(fo),"X")) # a ~ b + c + X
formula2string(a ~ b + c) # $left: "a" $right: "b+c"
formula2string(~ b + c) # $left: "" $right: "b+c"
formula2string(a ~ .) # $left: "a" $right: "."
formula2term.names(a ~ b1 + c,"left") # "a"
formula2Rterm.names(a ~ b1 + c) # "b1" "c"
```

strmatch	A "shortest unique identifier" match.
----------	---------------------------------------

Description

If an input string matches no output string, return NA.

If it is an ambiguous match, return 0.

Otherwise return the index of the match.

Exception: if the target string contains "log" and "logistic", and the user type "lo" it is ambiguous, but if he types "log" consider it a perfect match.

Usage

```
strmatch(inputs, target)
```

Arguments

inputs	The string to be found as a match.
target	The string which should contain the match.

Value

The matched string, if it exists. NA, if input matches no output string. 0, if ambiguous matches are found.

Note

Uses .C("strmatch", which is Not Found in R ???)

Author(s)

? from SCCS: \@(\\#)strmatch.s 2.2 1989-09-27

Examples

```
# strmatch(c("he", "we"), c("She will", "in awe", "xxxx"))
```

summaryFs	<i>Print extended summary of lm.</i>
-----------	--------------------------------------

Description

Print summary of lm, std.dev and Finney's correction to log normally distributed data.

Usage

```
summaryFs(lm)
```

Arguments

lm	Result of an <code>lm(log(y) ~ .)</code>
----	--

Note

Uses [FC.lm](#) and [s.lm](#)

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
x <- rnorm(1000); y <- 0.1*rnorm(1000)
summaryFs(lm(y ~ x))
```

T3plot	<i>T3plot</i>
--------	---------------

Description

T3 plot for a graphical check on normality together with 95%- and 99%-acceptance regions. If the black line does not cross either the 5% nor the 1% line, the input data are normal with less than 1% error.

Usage

```
T3plot(x, lab=paste("T3 plot of ", deparse(substitute(x))),
legend.pos="bottom", cex=0.6, ...)
```

Arguments

x Data vector.
 lab String for heading of plot.
 legend.pos, cex, ...
 see [legend](#).

Value

Is called for its side effect to produce a T3 plot.

Author(s)

Sucharita Ghosh, <rita.ghosh@wsl.ch>,
[http://www.wsl.ch/personal_homepages/gghosh/index_EN?redir=1&](http://www.wsl.ch/personal_homepages/gghosh/index_EN?redir=1&with_cosmetics),
 with cosmetics by Christian W. Hoffmann, <c-w.hoffmann@sunrise.ch>

References

Ghosh, S. (1996) A new graphical tool to detect non-normality. *Journal of the Royal Statistical Society B*, 58, 691-702.

Examples

```
T3plot(rnorm(100))
## Not run: T3plot(rnorm(10000))
## Not run: T3plot(rnorm(1000)+runif(1000)*0.1)
## Not run: T3plot(rnorm(1000)+runif(1000)*10)
```

 tex.table

Convert a data matrix into LaTeX code.

Description

These functions convert a data matrix into \LaTeX ; \LaTeX .

Usage

```
tex.table(datmat, bare = FALSE, precision = if (bare) "NA" else 2,
  rnames = if (bare) "-1" else dimnames(datmat)[[1]], cnames = if (bare)
  "-1" else dimnames(datmat)[[2]], caption = NULL, label = NULL,
  tpos = "b", stretch = NULL, adjust = "r", file = NULL)
tex.tab.head(datmat, precision = 2, rnames = NULL, cnames = NULL,
  caption = NULL, label = NULL, tpos = "b", stretch = NULL,
  adjust = "r", file = NULL)
tex.tab.tail(datmat, precision = 2, rnames = NULL, cnames = NULL,
  caption = NULL, label = NULL, tpos = "b", stretch = NULL,
  adjust = "r", file = NULL)
```

```
tex.tabelle(datmat, precision = 2, rnames = NULL, cnames = NULL,
  caption = NULL, label = NULL, tpos = "b", stretch = NULL,
  adjust = "r", file = NULL)
```

Arguments

datmat	data matrix
bare	TRUE: precision, rnames, cnames will get useful defaults, FALSE: set these parameters yourself
precision	precision of rounding within the LATEX table, if NA, then no transformation to numeric is done
rnames	row names
cnames	column names
caption	caption for LATEX table, default: no caption
label	LATEX label for the table, default: no lable
tpos	position of captions: above or below table, "a" for above, "b" for below
stretch	optional vector with two entries, giving the baselinestretch for the caption (stretch[1]) and the columns of the table (stretch[2]); default: no adjustment of baselinestretch
adjust	adjusts the columns of the LATEX table, default: "r" (right), also possible: "l" (left) and "c" (centre) or user defined: "adjust=c("l","c","r",...)" yields {llcr...}
file	output file, default: printout in console

Value

These functions are called for their side effect to write to a file.

Author(s)

Adapted by: Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

tri

Ternary or Triangular Plots.

Description

plotCI plots in a triangle the values of three variables. Useful for mixtures (chemistry etc.).

Usage

```
tri(a, f, m, symb=2, grid=FALSE, ...)
```

Arguments

a	Vector of first variable.
f	Vector of second variable.
m	Vector of third variable.
symb	Symbol to be plotted
grid	Plot the grid: TRUE or FALSE
...	Additional parameters for plot

Value

The function `tri` is called for its side effect to produce a plot. `codetrigrd` is internal to `tri`.

Author(s)

Colin Farrow Computing Service, University of Glasgow, Glasgow G12 8QQ, <c.farrow@comp.serv.gla.ac.uk>

Examples

```
# some random data in three variables
c1 <- runif(5, 10, 20)
c2 <- runif(5, 1, 5)
c3 <- runif(5, 15, 25)
# basic plot
tri(c1,c2,c3)
# plot with different symbols and a grid
tri(c1,c2,c3, symb=7, grid=TRUE)
```

waitReturn	<i>Wait for <Return></i>
------------	--------------------------------

Description

Wait for the user to type <Return>, depending on argument.

Usage

```
waitReturn(ask=TRUE)
```

Arguments

ask	TRUE will generate the interruption, FALSE will not.
-----	--

Details

The interruption will only be generated for the interactive use of R and if the call is not sinked (where it would hang the process).

Value

None.

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
for (ii in 1:5) {  
  cat(ii, "\n")  
  waitReturn(ii %% 2 == 1)  
}
```

weighted.mean1	<i>Weighted mean1</i>
----------------	-----------------------

Description

Compute the (trimmed) arithmetic mean of the vector of values given as its first argument. The weights are given in the second argument.

Usage

```
weighted.mean1(x, w=NULL, trim = 0, na.rm=FALSE)
```

Arguments

<code>x</code>	a numeric vector containing the values whose mean is to be computed.
<code>w</code>	a numeric vector containing the weights.
<code>trim</code>	the fraction (0 to 0.5) of observations to be trimmed from each end of <code>x</code> before the mean is computed.
<code>na.rm</code>	a logical value indicating whether NA values should be stripped before the computation proceeds.

Value

If `trim` is zero (the default), the arithmetic mean of the values in `x` is computed. If `trim` is non-zero, a symmetrically trimmed mean is computed with a fraction of `trim` observations deleted from each end before the mean is computed.

Author(s)

Christian W. Hoffmann, <c-w.hoffmann@sunrise.ch>

See Also

[weighted.mean](#), [mean](#), [quantile](#)

Examples

```

weighted.mean1(c(7,1,2,4,10,15),c(1,1/3,1/3,1/3,1,1)) # 8.583333333
weighted.mean1(c(1,2,4,7,10,15),c(1/3,1/3,1/3,1,1,1)) # ordered differently 8.583333333
weighted.mean1(c(7,7/3,10,15)) # same as previous, but unweighted:
# '1','2','4 of weights='1/3' are replaced by '7/3' (weight=1)

weighted.mean1(c(7,1,2,4,10),c(1,1/3,1/3,1/3,1)) # 6.444444444
weighted.mean1(c(7,1,2,4,10)) # 4.8
weighted.mean1(c(7,1,NA,4,10),c(1,1/3,1/3,1/3,1),na.rm =TRUE) # 7

```

weighted.median	<i>Weighted median</i>
-----------------	------------------------

Description

Compute the sample median of the vector of values given as its first argument. The weights are given in the second argument; if given, must be of the same length as first argument.

Usage

```
weighted.median(x, w=NULL, na.rm=FALSE)
```

Arguments

x	a numeric vector containing the values whose median is to be computed.
w	a numeric vector containing the weights.
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.

Value

Unweighted mean, if weights w are not given. Otherwise, value of x nearest to the position, where the cumulative sum of the weights reaches 50% of its maximum value.

Special cases are considered as closely as possible, see code.

Author(s)

Christian W. Hoffmann, <c-w.hoffmann@sunrise.ch>

See Also

[median](#), [quantile](#)

Examples

```
weighted.median(c(7,1,2,4,10,15),c(1,1/3,1/3,1/3,1,1)) # 8.5 [even number]
weighted.median(c(1,2,4,7,10,15),c(1/3,1/3,1/3,1,1,1)) # ordered differently 8.5
weighted.median(c(7,7/3,10,15)) # same as previous, but unweighted:
# '1','2','4' of weights='1/3' are replaced by '7/3' (weight=1)

weighted.median(c(7,1,2,4,10),c(1,1/3,1/3,1/3,1)) # 7
weighted.median(c(7,1,2,4,10)) # 4
weighted.median(c(7,1,NA,4,10),c(1,1/3,1/3,1/3,1),na.rm =TRUE) # 8.5
```

whole.number

Check an array on whole numbers (x in I).

Description

whole.number checks an array whether it consists of whole numbers only (x in I).

Usage

```
whole.number(x)
```

Arguments

x A numerical array.

Value

TRUE, FALSE

Author(s)

Bill Venables adapted by Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
whole.number(c(pi,2,3)) # FALSE
whole.number(c(1,2,3)) # TRUE
```

Index

- *Topic **NA**
 - jitterNA, 24
- *Topic **algebra**
 - div.prot, 12
 - lengths.angle, 25
 - normalize, 35
 - persp2, 42
 - pointfit, 46
 - rotm, 54
 - solveQeq, 63
- *Topic **arith**
 - dc, 9
 - eql, 15
 - frac, 18
 - num.ident, 35
 - numericString, 37
 - seqm, 59
 - signp, 61
- *Topic **array**
 - rtf, 55
 - select.range, 58
- *Topic **character**
 - numericString, 37
 - padding, 39
 - replacechar, 53
 - str2formula, 63
 - strmatch, 65
- *Topic **chron**
 - datetime, 9
 - delayt, 11
 - dt2str, 13
- *Topic **color**
 - plotSymbols, 43
- *Topic **data**
 - jitterNA, 24
 - whole.number, 72
- *Topic **device**
 - lpr, 28
 - panel, 40
- *Topic **distribution**
 - f.log, 15
 - invgauss, 23
 - my.table, 30
 - negbingof, 34
 - numberof, 37
 - poisgam, 47
 - qnorm.appr, 51
 - qres, 52
 - smoothed.df, 62
 - T3plot, 66
 - weighted.mean1, 70
 - weighted.median, 71
- *Topic **documentation**
 - cpos, 7
 - cwhmisc-internal, 8
 - ls.functions, 28
 - pasteInfix, 40
 - pasteRound, 41
 - setPowerPointStyle, 60
- *Topic **dplot**
 - ellipse, 13
 - persp2, 42
- *Topic **hplot**
 - lowess.bygroup, 27
 - mult.fig.p, 29
 - panel, 40
 - plotSymbols, 43
 - plt, 44
 - tri, 68
- *Topic **htest**
 - shapiro.wilk.test, 60
- *Topic **interface**
 - tex.table, 67
- *Topic **iplot**
 - plotSymbols, 43
- *Topic **logic**
 - is.constant, 23
 - num.ident, 35

- remove.dup.rows, 53
- *Topic **manip**
 - cbind.colnames, 5
 - clean.na, 6
 - libs, 26
 - napply, 33
 - remove.dup.rows, 53
 - str2formula, 63
- *Topic **math**
 - adaptlob, 3
 - interpol, 20
 - rtf, 55
- *Topic **misc**
 - cpos, 7
 - pasteInfix, 40
 - pasteRound, 41
- *Topic **models**
 - FinneyCorr, 16
 - like, 26
 - summaryFs, 66
- *Topic **multivariate**
 - ellipse, 13
 - p.screplot.princomp, 38
- *Topic **print**
 - cap, 4
 - datetime, 9
 - delstr, 11
 - formatFix, 17
 - heading, 19
 - n22dig, 31
 - n2c, 32
 - num2Latex, 36
 - printP, 48
 - progress.meter, 50
- *Topic **programming**
 - waitReturn, 69
- *Topic **regression**
 - FinneyCorr, 16
 - scode, 57
 - summaryFs, 66
- *Topic **symbolmath**
 - interpol, 20
- %%(pasteInfix), 40
- adaptlob, 3
- adaptlobstp (cwhmisc-internal), 8
- adaptsim (adaptlob), 3
- adaptsimstp (cwhmisc-internal), 8
- all.equal, 24
- allDigits (numericString), 37
- as.character, 41
- ASCII (cwhmisc-internal), 8
- availColors (plotSymbols), 43
- barplot, 38
- cap, 4
- capitalize (cap), 4
- CapLeading (cap), 4
- caplow.ff (cwhmisc-internal), 8
- capply (cap), 4
- cat, 41
- catn (printP), 48
- cbind, 5
- cbind.colnames, 5
- charMat, 44
- charMat (n2c), 32
- clean.na, 6
- conf.ellipse (ellipse), 13
- cpos, 7
- cwhmisc-internal, 8
- datetime, 9, 45
- dc, 9, 19
- dcn (dc), 9
- delayt, 11
- delstr, 11
- dev.copy, 28
- dinvgauss (invgauss), 23
- div.prot, 12
- dpoisgam (poisgam), 47
- drop, 7
- dt2str, 13
- ellipse, 13
- eql, 15
- evalInterp (interpol), 20
- expression, 44
- f.log, 15
- FALSE, 72
- FC.lm, 66
- FC.lm (FinneyCorr), 16
- FinneyCorr, 16
- for, 59
- formatFix, 17
- formula2Rterm.names, 64
- formula2Rterm.names (str2formula), 63

- formula2string, 64
- formula2string (str2formula), 63
- formula2term.names, 64
- formula2term.names (str2formula), 63
- frac, 18
- heading, 19
- HexDig (cwhmisc-internal), 8
- histRCT (plt), 44
- identical, 24
- indexLine (n2c), 32
- interpol, 20
- intToASCII (numericString), 37
- intToBase (numericString), 37
- intToHex (numericString), 37
- intToOct (numericString), 37
- invgauss, 23
- is.constant, 23
- jitter, 24
- jitterNA, 24
- layout, 30
- legend, 67
- lengths.angle, 25
- libs, 26
- like, 26
- list, 30
- loess.bygroup (lowess.bygroup), 27
- lower (cap), 4
- lowerize (cap), 4
- lowess.bygroup, 27
- lpr, 28
- ls.functions, 28
- mean, 70
- median, 71
- minInterp (interpol), 20
- mpf (dc), 9
- mtext, 29
- mult.fig.p, 29
- my.table, 30
- mydate (datetime), 9
- mytime (datetime), 9
- n22dig, 31, 33
- n2c, 32, 32
- n2cCompact (n2c), 32
- n2mfrow, 45
- napply, 33
- nchar, 41
- neg.bin.gof (negbingof), 34
- negbingof, 34
- normalize, 35
- NprinE (printP), 48
- NprinL (printP), 48
- NprinM (printP), 48
- NprinP (printP), 48
- NprinT (printP), 48
- NprinV (printP), 48
- num.ident, 35
- num2Latex, 36
- numberof, 37
- numericString, 37
- options, 36
- p.screepLOT.princomp, 38
- padding, 39
- panel, 40
- par, 27, 29, 30
- paste, 41
- pasteInfix, 40
- pasteRound, 41
- persp2, 42
- pinvgauss (invgauss), 23
- plot.default, 56
- plotmath, 44
- plotrtf (rtf), 55
- plotSymbols, 43
- plotSymbolsFonts (plotSymbols), 43
- plt, 44
- pltCharMat, 32
- pltCharMat (plt), 44
- pltRCT (plt), 44
- pn (printP), 48
- pointfit, 46, 55
- poisgam, 47
- postscript, 45
- ppoiscc (poisgam), 47
- ppoisgam (poisgam), 47
- princomp, 38
- prinE (printP), 48
- prinL (printP), 48
- prinM (printP), 48
- prinP (printP), 48
- prinT (printP), 48
- print, 48

printP, 48
printrtf (rtf), 55
prinV (printP), 48
progress.meter, 50

qinvgauss (invgauss), 23
qnorm, 51
qnorm.ap16 (qnorm.appr), 51
qnorm.app3 (qnorm.appr), 51
qnorm.app4 (qnorm.appr), 51
qnorm.appr, 51
qpoisgam (poisgam), 47
gres, 52
quadmin (interpol), 20
quantile, 70, 71

R2.lm (FinneyCorr), 16
remove.dup.rows, 53
replacechar, 53
rinvgauss (invgauss), 23
rotangle, 46
rotangle (rotm), 54
rotm, 14, 46, 54, 55
round, 41
rpoisgam (poisgam), 47
rtf, 55

s.lm, 66
s.lm (FinneyCorr), 16
scode, 57
select.range, 58
seq, 59
seqm, 59
setPowerPointStyle, 60
setupInterp (interpol), 20
shapiro.test, 61
shapiro.wilk.test, 60
sign, 61
signp, 61
smoothed.df, 62
solveQeq, 63
source, 44
spacC (cwhmisc-internal), 8
splom, 45
SplomT (plt), 44
sprintf, 10, 41
str2formula, 63
strmatch, 65
strsplit, 41
substr, 41
substring.location (cpos), 7
summaryFs, 66
Sweave, 9

T3plot, 66
table, 31
term.names2formula (str2formula), 63
tex.tab.head (tex.table), 67
tex.tab.tail (tex.table), 67
tex.tabelle (tex.table), 67
tex.table, 67
tri, 68
trigrd (tri), 68
TRUE, 72

waitReturn, 69
weighted.mean, 70
weighted.mean1, 70
weighted.median, 71
whole.number, 72