

Package ‘cwhmisc’

August 3, 2015

Version 6.0

Date 2015-07-30

Title Miscellaneous Functions for Math, Plotting, Printing,
Statistics, Strings, and Tools

Author Christian W. Hoffmann <christian@echoffmann.ch> <<http://www.echoffmann.ch>>

Maintainer Christian W. Hoffmann <christian@echoffmann.ch>

Depends R (>= 2.0), lattice, grid

Description Miscellaneous useful or interesting functions. Some parameters of functions may have changed, so beware!

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2015-08-03 12:09:33

R topics documented:

adaplob	3
arcs	4
astroC	5
astroGeo	6
cap	7
clean.na	8
clocksense	9
Const	9
coords	10
cpos	12
cwhmisc	13
cwhmisci	14
datetime	15
dcm	15
Ddim	17
delayt	17

delstr	18
digits	19
div.prot	20
dt2str	20
ellipse	21
eql	22
f.log	23
factor	24
FinneyCorr	25
formatFix	26
frac	27
functions	29
grepno	31
Halton	32
hours	33
int2	34
interpol	35
invgauss	36
is.constant	37
jitterNA	38
Julian date	39
libs	40
lowess.bygroup	41
lpr	42
ls.functions	42
mult.fig.p	43
my.table	44
n22dig	45
n2c	46
normalize	47
num.ident	48
num2Latex	49
numberof	50
numer	50
padding	51
panel	52
pasteInfix	52
pasteRound	53
plotSymbols	54
plt	55
pointfit	57
printP	59
progress.meter	61
qnorm.appr	62
RCA	63
remove.dup.rows	64
replacechar	65
scode	66

select.range	66
seqm	67
shapiro.wilk.test	68
signp	69
smoothed.df	70
T3plot	71
tex.table	72
triplot	73
w.median	74
waitReturn	75
whole.number	75

Index	77
--------------	-----------

adaptlob	<i>Numerically evaluate integral using adaptive rules.</i>
----------	--

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	tolerance for termination
$trace$	should intermediate steps be traced
\dots	additional parameters for function f .

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 <christian@echoffmann.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)$Q - 7.5 # -7.993606e-15 instead of 0
adaplob(FexGander,0,5,2.0e-6,TRUE) # 7.500002 instead of 7.5
adaplob(FexGander,0,5,2.0e-6)$Q - 7.5 # 1.781274e-06 instead of 0
adaplob(FexGander,0,5)$Q-7.5 # instead of -8.881784e-16, with warnings
# that required tolerance is too small.
adaplob(FexGander,0,5,5.0*.Machine$double.eps)$Q-7.5 # -5.329071e-15
```

arcs

Convert and reduce arcs

Description

Functions for reduction of arcs.

Usage

```
deg( radian )
rad( degree )
reda( U, ref )
reda2(U, V, ref )
```

Arguments

U, V, ref, radian, degree
Real

Details

deg Convert radians to degrees.
rad Convert degrees to radians.
reda Add or subtract multiples of ref to make $abs(U) < ref/2$.
reda2 Subtract from U and V the greatest multiple of ref, so that $0 \leq \min U_{new}, V_{new} < ref$.

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
deg(pi/2) # 90
rad(180) # 3.141593
reda(580,360) # -140
reda2(200,120,70) # 130, 50
reda2(100,-200,70) # 310, 10
```

astroC

Astronomical constants

Description

Astronomical constants

Details

```
cJDJ2000 = 2451545.0 , Julian day number of the epoch J2000.0
cDAYPJULCENT = 36525.0 , days per julian century
cDAYPYEARTROP = 365.242198781 , days per tropical year
cDAYPYEARSID = 365.25636042 , days per sidereal year
cDAYPMONSYN = 29.53058868 , days per synodical month
cDAYPMONSID = 27.321655 , days per sidereal month
cK = 0.01720209895 , Gravitational constant, GAUSSian defintion
cC = 299792458.0 , [m/s] defined speed of light
cRE = 6378140.0 , [m] radius of earth.s equator
cMY = 0.01230002 , ratio mass of moon/mass of earth
cPRECESS = 5029.0966 , [arc sec] precession per year at 2000.0
cEPSOBL23.43929111 , [deg] inclination of ecliptic at 2000.0
cAE = 1.49597870E11 , [m] distance Earth to Sun
cSBYE = 332946.0 , ratio mass of Sun/mass of Earth
cSBYEM = 328900.5 , ratio mass of Sun/mass of, Earth+Moon
cSBYME = 6023600.0 , ratio mass of Sun/mass of Mercury
cSBYVE = 408523.5 , ratio mass of Sun/mass of Venus
cSBYMA = 3098710.0 , ratio mass of Sun/mass of Mars
cSBYJU = 1047.355 , ratio mass of Sun/mass of Jupiter
```

cSBYSA = 3498.5 , ratio mass of Sun/mass of Saturn
 cSBYUR = 22869.0 , ratio mass of Sun/mass of Uranus
 cSBYNE = 19314.0 , ratio mass of Sun/mass of Neptun
 cSBYPL = 13000000.0 , ratio mass of Sun/mass of Pluto
 cSOLBYSID = 1.00273790934 , ratio solar/sidereal day
 cSIDBYSOL = 0.99726956634 , ratio sidereal/solar day
 DPY = cDAYPJULCENT/100.0; days/jul.Jahr
 DAYINMONTH = c(31,28,31,30,31,30,31,31,30,31,30,31,31)

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

astroGeo

Convert geographical coordinates to and from Swiss topo coordinates

Description

Geographic and Swiss topo rectangular coordinates, X positive to the north, Y positive to the east (!)

Usage

LB2MK(long, lat)
 LB2YX(long, lat)
 YX2LB(yToEast, xToNorth)
 YX2MK(yToEast, xToNorth)

Arguments

long,lat, yToEast, xToNorth
 Real

Details

LB2MK From geogr. longitude and latitude to planar meridian convergence [gon].
 LB2YX From geogr. longitude and latitude to Swiss coordinates.
 YX2LB From Swiss coordinates to geogr. longitude and latitude.
 YX2MK From Swiss coordinates North and East to planar meridian convergence [gon].
 LongBerne, LatBerne geogr. coordinates of Berne, 7deg26'22.50" east, 46deg57'08.66" north.
 yToEastBerne, xToNorthBerne Swiss topo coordinates of refernce point near Berne.

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch> after H.Matthias, lecture 'Amtliche Vermessungswerke 1', ETH Zurich, 1986.

Examples

```
LB2MK( LongBerne, LatBerne) # 7.21188e-16 [gon]
LB2MK( 9.132582913360895, 46.18669420448755) # somewhere in Switzerland , 1.37472
LB2YX( LongBerne, LatBerne) # 600.0, 200.0
YX2LB ( yToEastBerne, xToNorthBerne ) # 7.439583333333334 469524055555557
YX2MK ( 600, 200) # = 0
```

 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.
...	additional parameters for function ff.

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 <christian@echoffmann.ch>

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"
```

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 <christian@echoffmann.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
```

clocksense *Functions for directed arcs*

Description

Functions for clocksense, i.e. directed arcs

Usage

```
IsCounterC12( U, V, ref )
IsCounterC13( U, V, W, ref )
ClockSense2( U, V, ref )
ClockSense3( U, V, W, ref )
```

Arguments

ref,U,V,W Real

Details

CounterClock, NoneClock, Clockwise = "clckws", "Cntclck","noneclck", "clckws"
 ClockSense2 Return the clock sense of U and V
 ClockSense3 Return the clock sense of U, V, W
 IsCounterC12 Check if the directed angle from U towards W is counter clockwise, including U==W. Ref is the measure of a full circle, 360 for degrees, 2*Pi for radians, 400 for gon
 IsCounterC13 Check if U, V, W form a counterclock wise sequence.

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
ClockSense2(0,220,360) # "clckws"
ClockSense2(0,170,360) # "Cntclck"
ClockSense2(0,0,360) # "noneclck"
```

Const *Constants*

Description

Constants

Details

```

c3Q := .Machine$double.xmax^0.75
c38 := sqrt(c3Q)
ASCII := ASCII characters corresponding to (0), 1..256
HexDig := '1' - '9', 'A' - 'F'
TAU := (1+sqrt(5))/2 # golden section constant = 1.6180

```

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

coords

convert coordinates, angles, simple vector operations

Description

Functions for conversion of coordinates; rotation matrices for post(pre)-multiplication of row(column) 3-vectors; Vector product(right handed), length of vector, angle between vectors.

Usage

```

toPol( x, y=0 )
toRec( r, phi=0 )
toSph( x, y, z )
toXyz( r, theta, phi )
rotZ( x, y, phi )
rotA( phi, P=c(0,0,1) )
rotV(v, w=c(0,0,1))
rotL(phi,k=1,m=2,N=3)
getAp( M )
angle(v,w)
lV( v )
scprod(v, w)
vecprod(v, w)
v %v% w

```

Arguments

x,y,z,r,theta,phi	Real, rectangular, spherical coordinates; x, y, z may be combined as c(x,y,z), and r, theta, phi as c(r,theta,phi)
P	c(x,y,z), coordinates of point or projection direction P-'0', with '0' =c(0,0,0).
v, w	3-vectors (x, y, z).
N	Order of the square rotation matrix >= 2.
k,m	Integers (m != k) describing the plane of rotation. m==k gives Unit matrix.
M	3x3 rotation matrix.

Details

toPol, toRec: Convert plane rectangular $c(x,y)$ \leftrightarrow polar $c(r,\phi)$; $\phi = \text{angle}(x\text{-axis,point})$.

toSph, toXYZ: Rectangular $c(x,y,z)$ \leftrightarrow spherical coordinates $c(r,\theta,\phi)$; $\theta = \text{angle}(z\text{-axis,P-'O'})$, $\phi = \text{angle}[\text{plane}(P,z\text{-axis}), \text{plane}(x-z)]$.

Value

toPol: $c(r, \phi)$, $r = \text{Mod}(z)$, $\phi = \text{Arg}(z)$; $\text{Re}(z) = x$, $\text{Im}(z) = y$
 toRec: $c(x, y)$, $x = \text{Re}(z)$, $y = \text{Im}(z)$; $\text{Mod}(z) = r$, $\text{Arg}(z) = \phi$
 toSph: $c(r, \theta, \phi)$, $r = \sqrt{x^2 + y^2 + z^2}$, $\theta = \text{atan2}(z, v)$,
 $\phi = \text{atan2}(y, x)$; $v = \sqrt{x^2 + y^2}$
 toXYZ: $c(x, y, z)$, $x = r \cdot \sin(\phi) \cdot \sin(\theta)$, $y = r \cdot \cos(\phi) \cdot \sin(\theta)$, $z = r \cdot \cos(\theta)$
 rotZ: $c(x', y')$ = rotated (x, y) by angle ϕ , counter clockwise,
 – Rotation matrices
 rotA: Rotation matrix to rotate around axis P - 'O'.
 rotV: Rotation matrix to rotate v into w.
 rotL: Matrix m for multiplication $m \%*\% \text{vector}$.
 getAp: List with rotation axis and rotation angle corresponding to input matrix.
 – Other:
 angle angle between vectors
 lV Euclidean (spatial) length of vector
 scprod scalar product
 vecprod vector product = cross product

Note

angle uses [acos](#) and [asin](#)
 $v \%v\% w$ same as $\text{vecprod}(v, w)$
 $v \%s\% w$ same as $\text{scprod}(v, w)$

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
pkg <- TRUE # FALSE for direct use
x <- toPol(1.0, 1.0) # $r 1.41421, $p 0.785398 = pi/4
y <- toRec(2.0,pi) # $x -2, $y 2.44921e-16
toPol(y[1], y[2]) # 2, pi
toRec(x[1], x[2]) # 1, 1
rotZ(1, 0, pi/2) # 6.123032e-17 1.000000e+00
x <- 1; y <- 2; z <- 3
(R <- toSph(x,y,z)) # r= 3.7416574, theta= 0.93027401, phi= 1.1071487
c(R[1],180/pi*(R[2:3])) # 3.741657 53.300775 63.434949
(w <- toXYZ(R[1], R[2], R[3])) # = x,y,z
rotZ(1,2,pi/2) # -2, 1
opar <- par(mfrow=c(2,4))
x <- seq(0,1,0.05)
```

```

phi <- c(pi/6,pi/4,-pi/6)
Data <- matrix(c(x^2*10,(x^2-10*x)*4,(x+10)*1.5),ncol=3)
## Data <- matrix(c(rnorm(99)*10,rnorm(99)*4,rnorm(99)*1.5),ncol=3)
lim <- range(c(Data,-Data))*1.5
RD <- Data %%% rotL(phi[1],1,2) # !! # rotate around z-axis
RD2 <- RD %%% rotL(phi[2],2,3) # !! # rotate further around x
RD3 <- RD2 %%% rotL(phi[3],1,2) # !! # rotate back around z
plot(Data[,-3],xlim=lim,ylim=lim,xlab="x",ylab="y",pty="s")
plot(RD[,-3],xlim=lim,ylim=lim,xlab="RD x",ylab="y",pty="s",pch=5,col="red")
plot(RD2[,-3],xlim=lim,ylim=lim,xlab="RD2 x",ylab="y",pch=6,col="blue")
plot(RD3[,-3],xlim=lim,ylim=lim,xlab="RD3 x",ylab="RD3 y",col="magenta")
plot(Data[,1],RD3[,1])
plot(Data[,2],RD3[,2])
plot(Data[,3],RD3[,3])
m <- rotL(phi[1],1,2) %%% rotL(phi[2],2,3) %%% rotL(phi[3],1,2) # !! #
if (pkg) {
  m <- rotL(phi[1],1,2) %%% rotL(phi[2],2,3) %%% rotL(phi[3],1,2) # !! #
  round(m %%% t(m),2) #!! # composite rotation matrix and orthogonality,
  # should be diag(3)
} else {
  m <- rotL(phi[1],1,2) %%% rotL(phi[2],2,3) %%% rotL(phi[3],1,2) # !! #
  round(m %%% t(m),2) #!! # composite rotation matrix and orthogonality,
  # should be diag(3)
}
eye <- c(0.5,2.5,4)
re <- rotV(eye)
getAp(re) # $A [1] -9.805807e-01 1.961161e-01 -1.193931e-16
# $phi [1] 0.5674505
round(rotA(pi/1.5, c(1,1,1)),2) # 60 degrees around octant bisector
# [1,] 0 1 0 is permutation of axes 1 -> 2 -> 3 -> 1
# [2,] 0 0 1
# [3,] 1 0 0

```

cpos

Find the position of a substring

Description

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

Usage

```

cpos(str, sub, start=1)
cposR(str, sub, restrict)

```

Arguments

```

str          string)
sub          string)

```

start integer
 restrict vector of lower and upper index the search should be restricted to. If missing, whole str is taken.

Value

cpos number, if found, NA otherwise.
 cposR list(first,last) for each occurrence
 If there is none of first=NA,last=NA.

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

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

 cwhmisc

A collection of useful functions and constants

Description

Useful functions and constants for mathematics, astronomy, plotting, printing, data manipulation, statistics, string manipulation, etc.

Details

Package: cwhmisc
 Type: Package
 Version: 6.0
 Date: 2015-07-30
 License: GPL (>= 2)
 Author: Christian W. Hoffmann <christian@echoffmann.ch> www.echoffmann.ch
 Maintainer: same

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
# Show use of 'SplomT'
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="")
# splom(~data,cex=0.2)
try( SplomT(data,mainL="SplomT with random data",hist="d",cex.diag=0.6,hist.col="green") )
```

cwhmisci

Functions not to be called directly by the user.

Description

Recursive internal functions to adapt..

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	function to be integrated.
a	starting abscissa of integral.
b	ending abscissa of integral.
fa, fm, fb	function values at a, (a+b)/2, b.
is	parameter to control precision.
trace	should intermediate steps be traced
...	additional parameters for function f.

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 <christian@echoffmann.ch>

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 <christian@echoffmann.ch>

Examples

```
datetime() #[1] "2014-10-03, 16:00:03"  
mydate()  #[1] "2014-10-03"  
mytime()  #[1] "16:00:09"
```

dcm	<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 <code>formatFix</code> , 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 = integer` .
 dc uses `frac`, dcn uses `sprintf`.

Author(s)

Christian W. Hoffmann <christian@echoffmann.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"
mpf(pi,5); mpf(-pi,5) # "+ 3.14159" "- 3.14159" Note the space after the sign.

#### In example 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 then 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
```

Ddim	<i>dim of vectors and arrays</i>
------	----------------------------------

Description

Get length of vectors and dimension of arrays in a unified manner.

Usage

```
Ddim(x)
```

Arguments

x vector or array

Value

Integer vector containing length of vector or dimension of array.

Author(s)

Christian W. Hoffmann, <christian@echoffmann.ch>

Examples

```
Ddim(matrix(1:12,3,4)) # 3 4
Ddim(rep(0,5)) # 5
```

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 <christian@echoffmann.ch>

Examples

```
Sys.time(); nrof <- delayt(5); Sys.time()
print(nrof) # 26596 on my machine (2.33 GHz MacBook Pro)
```

delstr

Delete a substring from a string

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 <christian@echoffmann.ch>

Examples

```
delstr("Ich gehen heute in den Garten hinein","en")
# "Ich geh heut in d Gart hinein"
delstr("12345","x") # "12345"
```

digits *Test, convert numbers*

Description

Test, convert numbers

Usage

```
allDigits( str )
isNumeric(str)
xToBase( x, base=2 )
```

Arguments

base	Integer
x	Real
str	String

Details

`allDigits` Convert literally a string to a number.
`isNumeric` Test whether the elements of a character vector represent legal numbers only.
`str2dig` Convert a string to a vector of integers.
`int` integer part truncate towards 0.
`xToBase` return `res$a`, `res$e` so that
 $x = a * \text{base}^e$, $\text{abs}(a)$ in $[1, \text{base})$

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
allDigits(c("1231", "89a8742")) # TRUE FALSE
isNumeric(c("1231", "8.9e-2", ".7d2")) # [1] TRUE TRUE FALSE
str2dig("10010") # [1] 1 0 0 1 0
int(10^(7:10)) # 10000000 100000000 1000000000 NA
x <- 8.16617
xToBase(x, 2); xToBase(x, 3); xToBase(x, 4)
# 8.16617 = 1.02077*2^3 = 2.72206*3^1 = 2.04154*4^1
```

div.prot	<i>Protected division</i>
----------	---------------------------

Description

num/den, but num/0 -> .Machine\$double.xmax^(3/4)

Usage

```
div.prot(num,den)
```

Arguments

den, num real, numerator and denominator

Value

num/den, if is.infinite(num/den) then .Machine\$double.xmax^(3/4), the ^(3/4) for getting something well below Inf.

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
de <- .Machine$double.eps
v<-c(0,de/c(1,2,4,8))
div.prot(1,v)
# 1.55252e+231 4.50360e+15 9.00720e+15 1.80144e+16 3.60288e+16
```

dt2str	<i>Convert time difference to string.</i>
--------	---

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, with dec decimals in seconds.

Note

Enclosing the above statements in a function is likely to show zero time.

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
t1 <- unclass(Sys.time())
x <- 0; for (i in 1:1.e6) x <- x+1
t2 <- unclass(Sys.time())
dt2str(t2-t1,3) # 00:00:2.835
```

ellipse

Generate ellipses

Description

Given the axes a , b and an angle ϕ (in radian, counter clockwise), and the midpoint $c(0,0)$, points on the ellipse $(y)' * A^{(-1)} * (y) = const^2$ (with $A = rotL(\phi, 3) * matrix(c(a, 0, 0, b), 2, 2)$) will be generated, see [rotL](#).

Usage

```
ellipse1(k, a, b=a, ra=c(-1,361), phi=0 )
conf.ellipse(k, a, b, phi, df1, df2, level = 0.95)
```

Arguments

k	the number of generated points on the ellipse.
a	major axis
b	minor axis
ra	range of arc [deg]
ϕ	angle in radian describing the counter clockwise rotation from the x-axis to the major axis.
$df1$, $df2$, $level$	degrees of freedom and probability level of F-distribution.

Value

Matrix with columns consisting of x and y coordinates
 ellipse .. of the ellipse. conf.ellipse .. of the confidence ellipse according to
 qf(level, df1, df2), see [qf](#).

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
opar <- par(mfrow=c(1,1))
k <- 60; m <- c(0,0); a <- 2; b <- 1; phi <- pi/7
df1 <- 2; df2 <- 20
# show F for different confidence levels:
p <- c(0.5, 0.75, 0.8, 0.95)
qf(p, df1, df2) # 0.717735 1.486984 1.746189 3.492828
el7 <- conf.ellipse(k,a,b,phi,df1,df2,p[2]) + m
plot(el7*1.3,type="n",xlab="Different confidence ellipses",ylab="")
lines(conf.ellipse(60,a,b,phi,df1,df2,p[1]) + m,lty=2,col="red")
lines(conf.ellipse(60,a,b,phi,df1,df2,p[3]) + m,lty=2,col="green")
lines(conf.ellipse(60,a,b,phi,df1,df2,p[4]) + m,lty=2,col="blue")
lines(el7,lty=2,col="orange")
legend(x="bottom",paste(as.character(p*100),rep("%",length(p)),sep=""),
col = c("red", "orange", "green", "blue"), text.col="black", lty = c(2,2,2,2),
merge = TRUE, bg='white', cex=0.9)
par(opar)
```

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 <christian@echoffmann.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,NA,3),c(NA,NaN,1,2,3)) #> 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 <christian@echoffmann.ch>

Examples

```
x <- c(rep(0,20), exp(rnorm(1000,0.05)))
fx <- f.log(x)
## Not run:
oldpar <- par(mfrow = c(2, 3))
plot(x,main="exp(normal)+zeros")
qqnorm(x)
T3plot(x)
plot(fx,main="optimized offset")
qqnorm(fx)
```

```
T3plot(fx)
par(oldpar)

## End(Not run)
```

factor

Create primes, factor an integer, combine factors, check if prime

Description

Create primes, determine the prime factors of an integer together with their multiplicities, recombine factors, primitive version of the sieve of Eratosthenes.

Usage

```
primes(n)
Eratosthenes(n)
factorN(n)
prodN(fp)
is.prime(n)
```

Arguments

n	positive integer, number of primes, number to be factored, to be tested
fp	2-column matrix with prime factors and multiplicities

Value

primes	Generate the first n primes, also found in PRIMES.
Eratosthenes	Execute the sieve of Eratosthenes.
factorN	Determine the prime factors together with their multiplicities.
prodN	Recombine factors, inverse of factorN.
is.prime	Check if positive integer is prime.
PRIMES	The first primes up to 17389.

Author(s)

Christian W. Hoffmann <hristian@echoffmann.ch>

Examples

```
(p <- factorN( 423))
## [1,] 3 47
## [2,] 2 1
# meaning 423 = 3^2 * 47^1
prodN(p) # 423
is.prime(.Machine$integer.max) # TRUE
is.prime(16) # FALSE
## check speed of your machine
s <- Sys.time(); p<-primes(10^4);difftime(Sys.time(),s)
## Time difference of 2.906242 secs on my machine
x <-factorN(.Machine$integer.max)
```

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 $s_{\ln x}$.

Usage

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

Arguments

`s` Standard deviation s_{\ln} of log data, in note.
`n` Number of data points.
`lmobj` Result of an `lm(log(y) ~ .)`

Value

FinneyCorr Finney's correction from standard deviation and degrees of freedom. `FC.lm` Finney's correction from `lmobj`. `R2.lm` R-squared from `lmobj`. `s.lm` Comprehensive output from `lmobj`.

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 <christian@echoffmann.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 "-"), with 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 $abs(x) \geq 10^8$, values very near 10^k cannot be represented exactly, so the normal `format` will be used.

Names are retained. The vector or array structure will be preserved

Usage

```
formatFix(x, after=2, before=1, 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 <christian@echoffmann.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" "***" "***" "***" "***" "-3" "-92" "-0" " 0" " NA" " 0" " 92"
#> [13] " 3" "***" "***" "***" "***" "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; continuous fractions

Description

Split off fractional part of a number, compute and evaluate continuous fractions.

Usage

```

int( x )
frac(x,d)
contfrac( x, depth = 13, f=floor )
evalcfr( cf )
toCFrac( x, depth=5)
toCFrac2( x, depth=5)

```

Arguments

x	Real
d	If not missing, determines number of decimals after "."
f	function to use, normally 'floor', otherwise 'round' or 'trunc'
cf	Vector of integers representing the continued fraction of a real number
depth	Integer

Value

int integer part truncate towards 0.
frac fractional part, if d is missing; else
round($10^d * \text{fractionalpart}$), i.e. the fractional part as "integer" (rounded).
contfrac Convert to simple continued fraction representation, $cf := a_1 + 1/(a_2 + 1/(a_3...))$.
evalcfr Evaluate simple continued fraction to corresponding real.
toCFrac Build rational approximation num/den to x using forward continued fraction recursion to a depth of depth. Stopping criterion: either depth is reached, or $\text{abs}(x - \text{num}/\text{den})$ is increasing again.
toCFrac2 same as toCFrac, but vectors of partial numerators and denominators are returned.

Note

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

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
int(c(0,pi,2*pi,30*pi)) # 0 3 6 94
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
(pcf <- contfrac(pi)) # 3, 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1, 14, 2, (1)
## last integer incorrect due to rounding errors
evalcfr(pcf)-pi # 0
## To see the first approximants of pi, all of them famous:
for(ii in 1:15) {x<-toCFrac(pi,ii)
print(paste(ii,":",x$num,"/",x$den,"="))
print(paste(formatFix(x$num/x$den,15)," error = ",x$num/x$den-pi))}
# Note how the approximations taper off after depth 5:
# 10 3959189 / 1260249 = 3.141592653515298 -7.44955208631382e-11"
## Same, all at once:
F <- toCFrac2(pi,5) # $num 3 22 333 355 $den 1 7 106 113
toCFrac(pi, 10) #
```

 functions

Functions for testing and other

Description

Functions for testing on equality, exactly or with a tolerance, functions usable as parameters in other functions.

Usage

```

inrange(x,r)
mod( m, n )
modS( m, n )
modR( x, y )
one ( x )
zero ( x )
onebyx( x )
sqr( x )
powr( a, x )
equal( x, y )
equalFuzzy( x, y, prec=8*.Machine$double.eps, rel=TRUE )
quotmean(x,y)
safeDiv( num, den )
solveQeq(a,b,c)
chsvd(s)
divmod(x,y)
submod(x, v)
dsm( x, w )
chsvd(s)

```

Arguments

m, n, num, den	Integer
a, b, c, prec, x, y	Real
r	Real vector
rel	Boolean
s	square matrix, result of svd
v	real vector > 0, preferably cumsum of some other positive vector
w	real vector > 0

Details

mod = $m \% \% n$
 modS : modulo symmetric to 0.
 modR: modulo for reals, towards negative infinity.

zero returns 0.0 on any argument
 one returns 1.0 on any argument
 onebyx = 1.0/x
 sqr = x^2
 powr = a^x, with 0^0 := 1, 0^x := 0
 equal x == y
 equalFuzzy One can choose between relative and absolute precision
 safeDiv Compute quotient, set 0/0 -> 1, and safeguard r/0 <- c3Q otherwise
 quotmean Compute quotient of means of non-NA elements of x by y
 solveQeq Solve the quadratic equation given by the coefficients, return two solutions if a != 0, else one solution, possibly NA
 inrange Check if x is in the range r
 chsvd Check for [svd](#) to reproduce matrix
 divmod div, mod combined for ease of use
 submod analog to divmod for unequally spaced data
 dsm combination of divmod and submod, used in [Date](#)

Note

see also examples of [date](#)

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```

mod((-3:5),4 ) # 1 2 3 0 1 2 3 0 1
modS((-3:5),4) # -3 -2 -1 0 1 2 3 0 1
x <- 200; y <- x + 0.1
equalFuzzy(x,y,0.1*c(10^(-3:0))) # FALSE TRUE TRUE TRUE
equalFuzzy(x,y,0.1*c(10^(-3:0)),FALSE) # FALSE FALSE FALSE TRUE
safeDiv(0:3,c(0,0:2)) # 1.552518e+231
solveQeq(0,0,1) # NA NA
solveQeq(0,1,0) # 0
solveQeq(0,1,1) # -1
solveQeq(1,0,0) # 0 0
solveQeq(1,0,1) # 0-1i 0+1i
solveQeq(1,1,0) # -1 0
solveQeq(1,1,1) # -0.5-0.866025i -0.5+0.866025i
solveQeq(sample(1:4,1),sample(1:4,1),sample(1:4,1))
  x <- matrix(rnorm(9),3,3)
  s <- svd(x)
  lV(s$d)
norm(chsvd(s) - x) # 9.4368957e-16
submod(8.1,c(10.3, 31) ) # 0.0 8.1
submod(18.1,c(10.3, 31) ) # 1.0 7.8

```

 grepno

Grep utility

Description

Show elements passing or not a grep.

Usage

```
grepnot(s,x,value=TRUE)
```

Arguments

s,x	String
value	Boolean

Details

s String regular expression, see `link{grep}`.
 x Array of strings to check with grep.
 value Third argument to grep proper.
 (strings which TRUE) "\ (strings with FALSE), "\ stands for "NOT"

Value

Elements

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
x <- c("Ab"," d","#12"," ",".1", "m.",",", "$y")
y <- c(".x", ".","c.", "$y", "_p", "12")
grepnot("^A[:alpha:]]+",x) # "Ab" "\ " d" "#12" " " ".1" "m." "" "$y"
grepnot("A[:alpha:]]+",x) # "Ab" "\ " d" "#12" " " ".1" "m." "" "$y"
grepnot("a[:alpha:]]+",x) # "\" ...
grepnot("#[:alnum:]]+",x) # "#12" "\" ...
grepnot("$",x) # "" "\" ...
grepnot("^[[:alnum:]]+)",x) # "\" ...
grepnot("^[[:alnum:]]+)",x) # "\" ...
grepnot("^[[:alpha:]]+",x,FALSE) # "1" "2" "\ " 3" "4" "5" "6" "7"
```

Halton

Halton's quasi-random numbers

Description

Generating quasi-random numbers by Halton's radical inversion algorithm.

Usage

```
HS247(K,N,R,P=rep(0,K))
```

Arguments

K	Integer, number of random sequences
N	Integer, length of the random sequences
R	Integer 1..K, roots of inversion, should be prime
P	Integer 1..K, starting points of inversion

Value

A matrix of K columns containing the sequences.

Author(s)

Christian W. Hoffmann, <hoffmann@wsl.ch>

Source

J. H. Halton, 1964. Algorithm 247: Radical-inverse quasi-random point sequence, Communications of the ACM, Vol.7,12, pp. 701 - 702 .

References

http://en.wikipedia.org/wiki/Halton_sequences

Examples

```
par(mfrow=c(2,2))
n <- 400
q1 <- HS247(2,400,c(2,2),c(0,pi/10))
q2 <- HS247(2,400,c(2,3))
q3 <- HS247(2,400,c(2,5))
q4 <- HS247(2,400,c(17,19)) # prone to correlations
q5 <- HS247(2,400,c(2,3),c(pi/10,pi/10))
of <- 0.2
q6 <- HS247(2,400,c(2,3),c(pi/10+of,pi/10+of))
plot (q1,pch="+",col="blue",cex=0.5,xlab="roots = (2,2), +blue, green")
points(q2,pch=4, col="green",cex=0.5)
```



```

plot (q2,pch=4,col="green",cex=0.5,xlab="roots = (2,3),
      :green, (2,5) :red, (17,19) magenta")
points(q3,pch=":",col="red")
points(q4,pch=4,col="magenta",cex=0.5)
plot (q2,pch=4,col="green",cex=0.5,xlab="roots = 2, 2, green, red")
points(q5,pch=5,cex=0.5,col="red")
plot (q5,pch=5,cex=0.5,col="red",xlab="roots = 2, 3, red")
points(q6,pch="-")

```

hours

convert hours

Description

Functions for conversion of hour representations

Usage

```

Hd( h, m, s )
Hdms( hd )
Hmsd( hms )

```

Arguments

h, m, s, hd, hms

Real, representing hours, minutes, seconds, decimal hours, and concatenated h,m,s

Details

Hd Return hours from h, min, sec.

Hms Return h, min, sec from hours, is inverse of Hd.

Hdms Return hh.mmss from hours.

Hmsd Decimal hours from hours.mmss, is inverse of Hdms.

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```

Hd( 12,25,17) # 12.421389
Hms(1.421389) # $h 1, $m 25, $s 17.0004
Hmsd(1.421389) # 1h 42m 13.89 -> 1.703858
Hdms(1.703858) # 1.421389

```

int2

*convert integers, string to integer vector***Description**

Functions for conversion of integers to bases other than decimal

Usage

```
intToASCII(i)
intToBase( i, base=2 )
intToOct( i )
intToHex( i )
```

Arguments

`i, base` Integer

Details

`isNumeric` Test whether the elements of a character vector represent legal numbers 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 integer to octal representation.

`intToHex` Convert integer to hex representation.

`contfrac` Convert to continued fraction representation.

`evalcfr` Evaluate continued fraction to corresponding real.

`toFrac` Build rational approximation num/den to x using forward continued fraction recursion to a depth of depth. Stopping criterion: either depth is reached, or $\text{abs}(x - \text{num}/\text{den})$ is increasing again.

`toFrac2` same as `toFrac`, but vectors of partial numerators and denominators are returned.

`xToBase` return `res$a`, `res$e` so that $x = a \cdot \text{base}^e$, $\text{abs}(a)$ in $[1, \text{base})$

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
intToASCII(1:255)[121:129] # "x" "y" "z" "{" "|" "}" "~" "\177" "\200"
sapply(1:50, intToBase, 2)
sapply(1:50, intToBase, 7)
sapply(1:50, intToOct)
sapply(1:50, intToHex)
```

interpol

*Polynomial and rational interpolation***Description**

Determine the argument of the minimum by polynomial or rational interpolation of given points x , y .

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 (latter for $add=FALSE$)

Value

setupInterp	Generate structure ss for evaluation in evalInterp
minInterp, quadmin	x -value of the minimum. NA if too few points are given or no minimum exists in x .

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

References

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

Examples

```

opar <- par(mfrow=c(2,2))
x <- c(1,2,4,6); y <- 1/x
pint <- function( x, y, add, dopoly, ylab="" ) {
  print(paste(" minimum at = ", minInterp(x,y,add=add,doPoly=dopoly) ) )
  xP <- setupInterp(x,y,TRUE)
  xT <- setupInterp(x,y,FALSE)
  x0 <- seq(0,7,0.1); yP <- evalInterp(x0,xP)
  yT <- evalInterp(x0,xT)
  plot(x,y,xlim=c(-0.5,7.5),ylim=c(min(y)-2,max(y)+2),cex=2,ylab=ylab)
  lines(x0,yP,col=2,cex=0.5)
  lines(x0,yT,col=4,cex=0.5,pch="+")
  legend(x="bottom",c("polynomial", "rational"), col = c(2,4),
    text.col= "black", lty = 1, merge = TRUE, bg='white')
}
pint(x,y,add=FALSE,dopoly=TRUE,"1/x") # 6 ?? = minimum
pint(x, (x-3)^2,add=FALSE,dopoly=TRUE,"(x-3)^2") # 3
pint(x,x+1.0/x,add=FALSE,dopoly=FALSE,"x+1.0/x dopoly=F") # 1 -1
pint(x,x+1.0/x,add=TRUE,dopoly=TRUE,"x+1.0/x dopoly=T") # 8.3471982 0.3194685
par(opar)

```

 invgauss

Inverse Gaussian Distribution

Description

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

Usage

```

dinvgauss(x, mu = stop("no shape arg"), lambda = 1)
pinvgauss(q, mu = stop("no shape arg"), lambda = 1)
rinvgauss(n, mu = stop("no shape arg"), lambda = 1)

```

Arguments

n	Integer
q, x	Real
mu, lambda	positve array of integers, means and scaling parameter

Value

dinvgauss: Inverse Gaussian distribution function
 pinvgauss: Random variates from inverse Gaussian distribution
 rinvgauss: Quantiles of the inverse Gaussian distribution

Note

$$p(x; \mu, \lambda) = \sqrt{\frac{\lambda}{2\pi x^3}} e^{-\frac{\lambda(x-\mu)^2}{2x\mu^2}}$$

Author(s)

Gordon Smyth, <gks@maths.uq.edu.au>, from sources of <paul.bagshaw@cnet.francetelecom.fr> e.a.

References

Chhikara and Folks, The Inverse Gaussian Distribution, Marcel Dekker, 1989. http://en.wikipedia.org/wiki/Inverse_Gaussian_distribution

Examples

```
n <- 10;
```

```
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	<i>Jitter vector containing NA</i>
----------	------------------------------------

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 <code>jitter</code> .

Value

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

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

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

Julian date	<i>calender conversions</i>
-------------	-----------------------------

Description

calender conversions, Julian day number from civil date and back, names of months, weekdays.

Usage

```
Dat2Jul( yr, mo, dy, hr=12 )
Jul2Dat( JD )
monthsN( leap )
Mnames
Dnames
mdiny( dk, leap )
Wday( JD )
Yday( mo, dy, leap )
```

Arguments

yr, mo, dy	integer, year, month, day of date
hr, JD	real, hrs, Julian date
leap	Boolean, = is concerning year a leap year ?
dk	Integer

Value

Dat2Jul: JD, year Y BC is to be given as -|Y-1|, e.g. 4 BC = -3 !!
 Jul2Dat: date (year, month, day, hours).
 monthsN: cumulative sum of days in months.
 Mnames: names of months.
 Dnames: names of weekdays.
 mdiny: c(number of month, day in (leap) year).
 Wday: name of weekday from dk mod 7. Yday: number of day, from 0 = Jan 1.

Note

See also
http://www.onlineconversion.com/julian_date.htm
http://en.wikipedia.org/wiki/Julian_day#Converting_Julian_or_Gregorian_calendar_date_to_Julian_Day_Number

Julian date is a continuous numbering of days since the biblical day of creation in 4713 BC, Jan. 1, 12 hours. The Julian calendar date 1582 Oct 4 was succeeded by the Gregorian calendar date 1582 Oct 15.

Conversion of Julian and Gregorian dates to Julian day number is done by Dat2Jul. The reverse is done by Jul2Dat.

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
Dat2Jul(-4712,01,01) # = 0 i.e. Start of Julian day numbering
Jul2Dat( 0 ) # -4712 1 1 12
formatFix(Dat2Jul(c(1582,10,4,23+59/60)),8,10) # 2299160.499.. = last day of Julian calendar
Jul2Dat( 2299160 ) # 1582 10 4 12
Dat2Jul(c(2000,3,2)) # 2451606
Jul2Dat(2451606.0) # 2000 3 2 12
mdiny(1,TRUE) # 1 1
mdiny(60,TRUE) # 2 29
```

libs

List all installed packages, or all functions in a package

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)
```

lowess.bygroup	<i>Plot data in groups, each group with separate lowess smoothing</i>
----------------	---

Description

data in groups (shown by variable group) are plotted

Usage

```
lowess.bygroup(x, y, group, span=2/3, col=seq_along(x), lty=seq_along(x))
```

Arguments

x, y	coordinate vectors of equal length
group	grouping variable, must be a vector of same length as x and y
span	span of smooting
col	colour of lines
lty	line type

Value

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

Author(s)

Christian W. Hoffmann, <christian@echoffmann.ch>

Examples

```
par(mfrow=c(1,1))
gr <- c(rep(1,20),rep(2,30),rep(3,50))
x <- seq_along(gr); y <- jitter(0.01*(x-50)^2 + 1,1000)
plot(x,y,pch=".",cex=4,xlab="Lowess, with spans = 0.2 (r,g,mag), 0.4 (blue) ")
lowess.bygroup(x,y,gr,span=0.2,col=c("red","green","magenta"),lty=rep(2,3))
lowess.bygroup(x,y,gr,span=0.4,col="blue")
```

lpr *Print an object*

Description

Print a given object

Usage

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

Arguments

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

Author(s)

Ray Brownrigg <ray@mcs.vuw.ac.nz>
modified by Christian W. Hoffmann <christian@echoffmann.ch>

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 plot, 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, mfc col,
           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>mfc col</code>	<i>instead of nr.plots</i> : integer(2) vector giving the rectangular figure layout for <code>par(mfc ol= .)</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, <christian@echoffmann.ch>

See Also

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

Examples

```
AA <- mult.fig.p(5, main= "Sine functions of different frequencies")
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= "Sine functions", 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 <christian@echoffmann.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 <= x <=10) in a compact way

Description

n22dig shows as two characters: "0.ab" as "ab", "1.00" as " I", "0" as " 0" (note the blank).

Usage

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

Arguments

x	A numerical vector or matrix with elements <= 1.
symm	If symm = TRUE then upper triangle will be shown as " ".

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 <christian@echoffmann.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 attribute `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`. `explainLegend` gives a more readable version of attribute `legend`.

Usage

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

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 matrix, as explained in *attribute* legend. n2cCompact pack charMat list(x,y,txt)

Note

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

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
n2c(c(10e20,-10e5,10,(10:0)/10,0.05))
# "X" "6" "1" "0" "&" "%" "#" "*" "=" "+" "-" ":" ", " " "
# attr("legend")
# [1] ">=1:log, >=0. 9& 8% 7# 6* 5= 4+ 3- 2: 1, 05. ' ' "
```

```
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)
explainLegend() #
```

normalize

base power and multiplier

Description

split a number into base power and multiplier, $x = a \cdot \text{base}^e$

Usage

```
normalize( x, base=2 )
```

Arguments

x	Real
base	Base of power

Details

normalize Return $c(a, e)$ so that $x = a \cdot \text{base}^e$, $\text{abs}(a) \in [1, \text{base})$

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
normalize(100,pi) ## 1.0265982 4.0000000
```

num.ident

Check numerical values for identity

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, FALSE

Note

No check is made whether x or y are numeric

Author(s)

Christian W. Hoffmann <christian@echoffmann.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

Count the number elements that satisfy a condition.

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 <christian@echoffmann.ch>

Examples

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

numer

Number theoretic functions

Description

Simple number theoretic functions

Usage

```
scm( m, n )  
EulerPhi( n )  
gcd( a, b )  
Euclid( a, b )  
Inv(a, n)  
modexp( a, b, n )
```

Arguments

a,b,m,n Integer

Value

EulerPhi Eulers totient function = number of divisors of n. scm, gcd Smallest common multiple, Greatest common divisor. Euclid Computes x, y from a, b such that the equation $a*x + b*y = \text{gcd}(m,n)$ is satisfied. Inv Modular inverse in a finite ring, NA if not exists. modexp Exponentiation modulo an integer via binary decomposition of exponent.

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
scm(35,133) # 665
gcd(35,133) # 7
Euclid(35,133) # 4 -1 7, meaning 4*35 +(-1)*133 = 7
EulerPhi(60) # 16
modexp(3,10,7) # 4
```

padding

Padding a string with justification

Description

padding Pads a string.

Usage

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

Arguments

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

Value

A string

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
padding("My string",25,"XoX","center")
# [1] "XoXxoXxoMy stringXoXoXoX"
```

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, b objects, will be coerced to character vectors.

Details

```
%% is currently defined as
"%%" <- function(a,b) paste0(a,b)
```

Value

see Details.

Author(s)

christian@echoffmann.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" %% "very" %% "hungry" # [1] "I am hungry"
```

pasteRound

Paste rounded values

Description

Paste rounded values

Usage

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

Arguments

... list of arguments to be pasted.
digits Integer, argument to [round](#).
sep, collapse Character, arguments to [paste](#).

Value

The concatenation of formatted values

Author(s)

Dimitris Rizopoulos <dimitris.rizopoulos@med.kuleuven.ac.be>, adapted by Christian Hoffmann <christian@echoffmann.ch>

Examples

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

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

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

Value

list of	
ch	character value of symbol
dec	decimal value of symbol
hex	hex value of symbol
oct	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, <christian@echoffmann.ch>

Examples

```
# A first impression:
opar <- par(mfrow=c(1,2))
n<-1:34; plot(n,pch=n) # There is a gap between 25 and 34
plotSymbols(TRUE)
par(opar)
```

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 with a) covariances (with script size proportional to size) in the upper triangle, b) histograms (with smoothing) and variable names in the diagonal, and c) scatterplot with smoothes in y and x direction in the lower triangle, stressing high correlations by nearly parallel lines. See figure in other documentation.

Usage

```
pltCharMat(m,...)
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 = colnames(eval(substitute(data))))
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=0.4, colYonX = "red", colXonY = "blue", ...)
```

Arguments

m	Numerical matrix
tit	Overall title for plot. A vector of one or two elements. If an element is an expression , <code>plotmath</code> 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 <code>{...}</code> .
cex	Font size used for tit
reset	Should previous rows, cols be restored after execution. See note
outer	Passed on to <code>mtext</code> .

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, <christian@echoffmann.ch>, with the assistance of Deepayan Sarkar <Deepayan.Sarkar@r-project.org>.

Examples

```

x <- rnorm(100); y <- rnorm(100)+1; z <- y+rlnorm(100)
pltRCT(1,1,f={plot(x,y,xlab="data with trend");
  abline(reg=lm(y~x),lty=2);points(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="")
SplomT(data,mainL="",hist="d",cex.diag=0.6,hist.col="green")
SplomT(data,mainL="",hist="b",adjust=0.4,cex.diag = 0.5)
pltRCT(1, 1, tit="1 by 1 plot", f=plot(y,x-3*y) )
nr <- 25; nc <- 16
pltRCT(1, 2, f={plot(x,y,xlab="my x")
  m <- matrix(rnorm(nr*nc),nrow=25,ncol=nc)
  pltCharMat(m,cex=0.5,col="red")
})

```

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 : $xi = f * Q * x + tr + error$. 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 : $xi = f * Q * x + tr$. 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 <christian@echoffmann.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

`rotL` to generate rotation matrices

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
# theta <- runif(3)
theta <- c(pi/4, pi/15, -pi/6)
# orthogonal rotations to construct Qr
Qr <- rotL(theta[3]) %*% rotL(theta[2],1,3) %*% rotL(theta[1],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)) ## !!!!! %*% instead of %*%
# As a final check we generate the orthogonal matrix S from the computed angles
# theta and compare it with the result pf$Q
Ss <- rotL(theta[3])
range(svd(Ss*pf$factor - pf$Q)$d) # 6.652662e-17 1.345509e-01

```

printP	<i>Print without square brackets, expression values together with their call strings</i>
--------	--

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

x	A numerical vector or matrix.
before	See <code>formatFix</code> , the number of decimals before "."
after	See <code>formatFix</code> , the number of decimals after "."

rownam	Should row names be printed.
colnam	Should column names be printed.
xsv	A string representing a vector expression.
xs	A string representing an expression.
...	Additional parameters for <code>print</code> .

Author(s)

Christian W. Hoffmann <christian@echoffmann.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 $(i \% \% 10) \% \% 10$ if $i \% \% 10 == 0$. If $i \% \% 50 == 0$, a line break will be written and i printed.

Usage

```
progress.meter(i) # inside a function or loop
```

Arguments

`i` Integer.

Value

invisible(NULL).

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```

n <- 1 # adjust
for (i in 0:250) {
  kk <- 0
  for (mm in 1:10^n) {
    kk <- kk+1 # do something time consuming
  }
}

```

```

    }
    progress.meter(i)
  }
  cat("")
# 0...+...1...+...2...+...3...+...4...+...
# 50...+...6...+...7...+...8...+...9...+...
```

qnorm.appr

Approximation to the inverse normal distribution function.

Description

qnorm.ap* 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.ap* gives the quantile function for the different approximations.

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 <christian@echoffmann.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)) # relative precision
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)
# Check relative precision of approximations
prec(x,x3,1) # 0.002817442
prec(x,x4,1) # 0.0004435874
prec(x,x1,1) # 0.1571311 why so bad ?
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
```

RCA

Check, build, install package in a unified manner.

Description

Check, build, install package in a unified manner.

Usage

```
RCA(dir, pkg, sw=0:6, echoonly=FALSE, verbose=TRUE)
```

Arguments

dir	character, path of (collection of) package(s)
pkg	character, name of package

echoonly	boolean give echo of R CMD ..., more verbosely
verbose	boolean give only echo of intended 'R CMD ...'
sw	switch for alternatives, must be in 0:6, see note

Note

"RCA" calls system("R CMD <options> path-to-package ") with options sw
 0 = (show sw alternatives),
 1 = "Rd2pdf --no-clean --force",
 2 = "check",
 3 = "build --force",
 4 = "check --as-cran <pkg>.tar.gz",
 5 = "check --as-cran",
 6 = "install".
 The order 2 to 6 is suggested by 'Writing extensions'.
 sw = 1 shows errors present in the creation of the manual.
 sw = 4 is provided for checking as required by CRAN policy.
 'Rd.sty' should be provided in the directory where RCA is run.

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
## Not run:
RCA(<R-dir>,"mypackage", 2:4)

## End(Not run)
```

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, deprecated!

Usage

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

Arguments

<code>str</code>	The string to be altered.
<code>char</code>	The character to be replaced.
<code>newchar</code>	The character to replace with.

Value

The altered string.

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch> adapted from tjoelker@redwood.rt.cs.boeing.com (Rod Tjoelker 865-3197)

Examples

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

scode	<i>Generate the significance codes as in summary.lm</i>
-------	---

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 <christian@echoffmann.ch>

Examples

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

select.range	<i>Select values from a vector depending on a range in a second vector.</i>
--------------	---

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(data, groupvec, min, max)
```

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(2.1,4.3,3.2,5.1,4.2,5.7,7.1,6.5,4.1,5,6,8,7,9 ,8 ,NA,NA)
agevec  <- c(10 ,13 ,14 ,25 ,29 ,32 , 34, 45, 48, 55, 62,67,69,70,74)
select.range(testvec,agevec,25,34.5) # 5.1 4.2 5.7 7.1
```

seqm *sequences, empty if "by" not conforming*

Description

Generate sequences, but unlike "seq", return NULL, when "seq" would generate a backward sequence. 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

from	starting value of sequence.
to	(maximal) end value of the sequence.
by	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 <christian@echoffmann.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-traingle
  }
  cat("\n")
}
# 1      diag : 2 : 3
# 2        1 diag : 3
# 3        1  2 diag

```

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](#)**Examples**

```
shapiro.wilk.test(rnorm(100, mean = 5, sd = 3)) # $p 0.169547
shapiro.wilk.test(runif(100, min = 2, max = 4)) # $p 6.09393e-06
```

signp*Sign Function*

Description

signp returns a vector with the signs of the elements of x: ifelse(x >= 0, 1, -1), avoiding 0.

Usage

```
signp(x)
```

Arguments

x a numeric vector

Note

signp does not operate on complex vectors.

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

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

```
signp(-1:1) # -1 1 1 instead of sign(-1:1) = -1 0 1
(m <- matrix(rnorm(9),3,3))
m %% diag(signp(diag(m)))
```

smoothed.df	<i>Fit cumulative distribution from kernel estimate.</i>
-------------	--

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")
```

T3plot

T3plot

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>,
with cosmetics by Christian W. Hoffmann, <christian@echoffmann.ch>

References

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

Examples

```
par(mfrow=c(2,2))  
T3plot(rnorm(100))  
T3plot(rnorm(10000))  
T3plot(rnorm(1000)+runif(1000)*0.1,"Example 3")  
T3plot(rnorm(1000)+runif(1000)*10,"Example 4")
```

tex.table	<i>Convert a data matrix into LaTeX code.</i>
-----------	---

Description

These functions convert a data matrix into L^AT_EX.

Usage

```
tex.table(dm, bare = FALSE, prec = if (bare) "NA" else 2,
  rnames = if (bare) "-1" else dimnames(dm)[[1]], cnames = if (bare)
  "-1" else dimnames(dm)[[2]], caption = NULL, label = NULL,
  tpos = "b", stretch = NULL, adjust = "r", file = NULL)
tex.tab0(dm, prec = 2, rnames = NULL, cnames = NULL,
  caption = NULL, label = NULL, tpos = "b", stretch = NULL,
  adjust = "r", file = NULL)
```

Arguments

dm	data matrix
bare	TRUE: prec, rnames, cnames will get useful defaults, FALSE: set these parameters yourself
prec	precision of rounding within the L ^A T _E X table, if NA, then no transformation to numeric is done
rnames	row names
cnames	column names
caption	caption for L ^A T _E X table, default: no caption
label	L ^A T _E X label for the table, default: no label
tpos	position of captions: "a" for above table, "b" for below table
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 L ^A T _E X 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.

tex.table	generate complete minimal Tex-able .tex file, including 'footnotesize'
tex.tab0	same as 'tex.table' but without 'footnotesize'

Author(s)

Adapted by: Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```

m <- matrix(rnorm(100),nrow=10,ncol=10,dimnames=list(LETTERS[1:10],colnames=letters[1:10]))
tex.table(m,file="tex.table.tex")
# \begin{tabular}{r|rrrrrrrrrr}
# \hline
# & a & b & c & d & e & f & g & h & i & j \\ \hline
# A & -0.63 & 1.51 & 0.92 & 1.36 & -0.16 & 0.40 & 2.40 & 0.48 & -0.57 & -0.54 \\
# B & 0.18 & 0.39 & 0.78 & -0.10 & -0.25 & -0.61 & -0.04 & -0.71 & -0.14 & 1.21 \\
# ...

```

triplot

Ternary or Triangular Plots.

Description

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

Usage

```
triplot(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.

Author(s)

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

Examples

```
# some random data in three variables
c1 <- runif(25)
c2 <- runif(25)
c3 <- runif(25)
# basic plot
par(mfrow=c(1,2))
triplot(c1,c2,c3)
# plot with different symbols and a grid
triplot(c1,c2,c3, symb=7, grid=TRUE)
```

w.median

Weighted median

Description

Compute the weighted median.

Usage

```
w.median (x,w)
```

Arguments

x,w Real, data and weights

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

See Also

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

Examples

```
w.median(c(7,1,2,4,10,15),c(1,1/3,1/3,1/3,1,1)) # 7
w.median(c(1,2,4,7,10,15),c(1/3,1/3,1/3,1,1,1)) # 7
w.median(c(7,7/3,10,15)) # 7
# '1','2','4 of weights='1/3' are replaced by '7/3' (weight=1)
w.median(c(7,1,2,4,10),c(1,1/3,1/3,1/3,1)) # 7
w.median(c(7,1,2,4,10)) # 4
w.median(c(7,1,NA,4,10),c(1,1/3,1/3,1/3,1)) # 7
```

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

Description

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

Usage

```
waitReturn(q="",ask=TRUE)
```

Arguments

ask	TRUE will generate the interruption, FALSE will not.
q	String for prompt

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 <christian@echoffmann.ch>

Examples

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

whole.number	<i>Check an array on whole numbers (x in I).</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 <christian@echoffmann.ch>

Examples

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

Index

- *Topic **NA**
 - jitterNA, 38
- *Topic **algebra**
 - div.prot, 20
 - pointfit, 57
- *Topic **arith**
 - astroC, 5
 - astroGeo, 6
 - Const, 9
 - dcm, 15
 - eql, 22
 - frac, 27
 - int2, 34
 - num.ident, 48
 - seqm, 67
 - signp, 69
- *Topic **array**
 - select.range, 66
- *Topic **character,arith**
 - arcs, 4
 - clocksense, 9
 - coords, 10
 - digits, 19
 - functions, 29
 - hours, 33
 - numer, 50
- *Topic **character**
 - cpos, 12
 - grepno, 31
 - padding, 51
 - replacechar, 65
- *Topic **chron**
 - datetime, 15
 - delayt, 17
 - dt2str, 20
- *Topic **color**
 - plotSymbols, 54
- *Topic **data**
 - jitterNA, 38
 - whole.number, 75
- *Topic **device**
 - lpr, 42
 - panel, 52
- *Topic **distribution**
 - f.log, 23
 - Halton, 32
 - invgauss, 36
 - my.table, 44
 - numberof, 50
 - qnorm.appr, 62
 - smoothed.df, 70
 - T3plot, 71
- *Topic **documentation**
 - ls.functions, 42
 - pasteInfix, 52
 - pasteRound, 53
- *Topic **dplot**
 - ellipse, 21
- *Topic **hplot**
 - lowess.bygroup, 41
 - mult.fig.p, 43
 - panel, 52
 - plotSymbols, 54
 - plt, 55
 - triplot, 73
- *Topic **htest**
 - shapiro.wilk.test, 68
- *Topic **interface**
 - tex.table, 72
- *Topic **iplot**
 - plotSymbols, 54
- *Topic **logic**
 - is.constant, 37
 - num.ident, 48
 - remove.dup.rows, 64
- *Topic **manip**
 - clean.na, 8
 - libs, 40

- remove.dup.rows, 64
- *Topic **math**
 - adaptlob, 3
 - factor, 24
 - interpol, 35
 - Julian date, 39
 - normalize, 47
- *Topic **misc**
 - cpos, 12
 - pasteInfix, 52
 - pasteRound, 53
- *Topic **models**
 - FinneyCorr, 25
- *Topic **multivariate**
 - ellipse, 21
- *Topic **package**
 - cwhmisc, 13
- *Topic **print**
 - cap, 7
 - datetime, 15
 - delstr, 18
 - formatFix, 26
 - n22dig, 45
 - n2c, 46
 - num2Latex, 49
 - printP, 59
 - progress.meter, 61
- *Topic **programming**
 - waitReturn, 75
- *Topic **regression**
 - FinneyCorr, 25
 - scode, 66
- *Topic **robust**
 - w.median, 74
- *Topic **structures**
 - Ddim, 17
- *Topic **symbolmath**
 - interpol, 35
- *Topic **utils**
 - RCA, 63
- .adaptlobstp(cwhmisc), 14
- .adaptsimstp(cwhmisc), 14
- %+(pasteInfix), 52
- %s%(coords), 10
- %v%(coords), 10
- acos, 11
- adaptlob, 3
- adaptsim(adaptlob), 3
- all.equal, 37
- allDigits(digits), 19
- angle(coords), 10
- arcs, 4
- as.character, 53
- ASCII(Const), 9
- asin, 11
- astroC, 5
- astroGeo, 6
- availColors(plotSymbols), 54
- c38(Const), 9
- c3Q(Const), 9
- cAE(astroC), 5
- cap, 7
- capitalize(cap), 7
- CapLeading(cap), 7
- capply(cap), 7
- cat, 53
- catn(printP), 59
- cC(astroC), 5
- cDAYPJULCENT(astroC), 5
- cDAYPMONSID(astroC), 5
- cDAYPMONSYN(astroC), 5
- cDAYPYEARSID(astroC), 5
- cDAYPYEARTROP(astroC), 5
- cEPSOBL(astroC), 5
- charMat, 55
- charMat(n2c), 46
- chsvd(functions), 29
- cJDJ2000(astroC), 5
- cK(astroC), 5
- clean.na, 8
- clocksense, 9
- ClockSense2(clocksense), 9
- ClockSense3(clocksense), 9
- Clockwise(clocksense), 9
- cMY(astroC), 5
- conf.ellipse(ellipse), 21
- Const, 9
- contfrac(frac), 27
- coords, 10
- CounterClock(clocksense), 9
- cpos, 12
- cposR(cpos), 12
- cPRECESS(astroC), 5
- cRE(astroC), 5
- cSBYE(astroC), 5
- cSBYEMY(astroC), 5

- cSBYJU (astroC), 5
- cSBYMA (astroC), 5
- cSBYME (astroC), 5
- cSBYNE (astroC), 5
- cSBYPL (astroC), 5
- cSBYSA (astroC), 5
- cSBYUR (astroC), 5
- cSBYVE (astroC), 5
- cSIDBYSOL (astroC), 5
- cSOLBYSID (astroC), 5
- cwhmisc, 13
- cwhmisc-internal (cwhmisci), 14
- cwhmisc-package (cwhmisc), 13
- cwhmisci, 14

- Dat2Jul (Julian date), 39
- Date, 30
- date, 30
- datetime, 15, 56
- DAYINMONTH (astroC), 5
- dc, 28
- dc (dcm), 15
- dcm, 15
- dcn (dcm), 15
- Ddim, 17
- deg (arcs), 4
- delayt, 17
- delstr, 18
- dev.copy, 42
- digits, 19
- dinvgauss (invgauss), 36
- div.prot, 20
- divmod (functions), 29
- Dnames (Julian date), 39
- DPY (astroC), 5
- drop, 8
- dsm (functions), 29
- dt2str, 20

- ellipse, 21
- ellipse1 (ellipse), 21
- eq1, 22
- equal (functions), 29
- equalFuzzy (functions), 29
- Eratosthenes (factor), 24
- Euclid (numer), 50
- EulerPhi (numer), 50
- evalcfr (frac), 27
- evalInterp (interp), 35

- explainLegend (n2c), 46
- expression, 55

- f.log, 23
- factor, 24
- factorN (factor), 24
- FALSE, 48, 76
- FC.lm (FinneyCorr), 25
- FinneyCorr, 25
- for, 67
- format, 26
- formatFix, 26
- frac, 27
- functions, 29

- gcd (numer), 50
- getAp (coords), 10
- GreatestIntAsRealF (Const), 9
- grepno, 31
- grepnot (grepno), 31

- Halton, 32
- Hd (hours), 33
- Hdms (hours), 33
- HexDig (Const), 9
- histRCT (plt), 55
- Hms (hours), 33
- Hmsd (hours), 33
- hours, 33
- HS247 (Halton), 32

- identical, 37
- indexLine (n2c), 46
- inrange (functions), 29
- int (frac), 27
- int2, 34
- interp, 35
- intToASCII (int2), 34
- intToBase (int2), 34
- intToHex (int2), 34
- intToOct (int2), 34
- Inv (numer), 50
- invgauss, 36
- is.constant, 37
- is.prime (factor), 24
- IsCounterCl2 (clocksense), 9
- IsCounterCl3 (clocksense), 9
- isNumeric (digits), 19

- jitter, 38

- jitterNA, 38
- Jul2Dat (Julian date), 39
- Julian date, 39
- LatBerne (astroGeo), 6
- layout, 44
- LB2MK (astroGeo), 6
- LB2YX (astroGeo), 6
- legend, 71
- libs, 40
- list, 44
- LongBerne (astroGeo), 6
- lower (cap), 7
- lowerize (cap), 7
- lowess.bygroup, 41
- lpr, 42
- ls.functions, 42
- lv (coords), 10
- mdiny (Julian date), 39
- median, 74
- minInterp (interpol), 35
- Mnames (Julian date), 39
- mod (functions), 29
- modexp (numer), 50
- modR (functions), 29
- modS (functions), 29
- monthsN (Julian date), 39
- mpf (dcm), 15
- mtext, 43
- mult.fig.p, 43
- my.table, 44
- mydate (datetime), 15
- mytime (datetime), 15
- n22dig, 45
- n2c, 46, 46
- n2cCompact (n2c), 46
- n2mfrow, 56
- nchar, 53
- NoneClock (clocksense), 9
- normalize, 47
- NprinE (printP), 59
- NprinL (printP), 59
- NprinM (printP), 59
- NprinP (printP), 59
- NprinT (printP), 59
- NprinV (printP), 59
- num.ident, 48
- num2Latex, 49
- numberof, 50
- numer, 50
- one (functions), 29
- onebyx (functions), 29
- options, 49
- padding, 51
- panel, 52
- par, 43, 44
- paste, 53
- pasteInfix, 52
- pasteRound, 53
- pinvgauss (invgauss), 36
- plotmath, 55
- plotSymbols, 54
- plotSymbolsFonts (plotSymbols), 54
- plt, 55
- pltCharMat, 46
- pltCharMat (plt), 55
- pltRCT (plt), 55
- pn (printP), 59
- pointfit, 57
- postscript, 56
- powr (functions), 29
- PRIMES (factor), 24
- primes (factor), 24
- prinE (printP), 59
- prinL (printP), 59
- prinM (printP), 59
- prinP (printP), 59
- prinT (printP), 59
- print, 60
- printP, 59
- prinV (printP), 59
- prodN (factor), 24
- progress.meter, 61
- qf, 22
- qinvgauss (invgauss), 36
- qnorm, 62
- qnorm.ap16 (qnorm.appr), 62
- qnorm.app3 (qnorm.appr), 62
- qnorm.app4 (qnorm.appr), 62
- qnorm.appr, 62
- quadmin (interpol), 35
- quantile, 74
- quotmean (functions), 29

R2.lm (FinneyCorr), 25
rad (arcs), 4
RCA, 63
reda (arcs), 4
reda2 (arcs), 4
remove.dup.rows, 64
replacechar, 65
rinvgauss (invgauss), 36
rotA (coords), 10
rotL, 21, 58
rotL (coords), 10
rotV (coords), 10
rotZ (coords), 10
round, 53

s.lm (FinneyCorr), 25
safeDiv (functions), 29
scm (numer), 50
scode, 66
scprod (coords), 10
select.range, 66
seq, 67
seqm, 67
setupInterp (interpol), 35
shapiro.test, 69
shapiro.wilk.test, 68
sign, 69
signp, 69
smoothed.df, 70
solveQeq (functions), 29
source, 55
splom, 56
SplomT (plt), 55
sprintf, 16, 53
sqr (functions), 29
str2dig (digits), 19
strsplit, 53
submod (functions), 29
substr, 53
summaryFs (FinneyCorr), 25
svd, 29, 30
Sweave, 15

T3plot, 71
table, 45
TAU (Const), 9
tex.tab0 (tex.table), 72
tex.table, 72
toCFrac (frac), 27
toCFrac2 (frac), 27
toPol (coords), 10
toRec (coords), 10
toSph (coords), 10
toXYZ (coords), 10
triplet, 73
TRUE, 48, 76

vecprod (coords), 10

w.median, 74
waitReturn, 75
Wday (Julian date), 39
whole.number, 75

xToBase (digits), 19
xToNorthBerne (astroGeo), 6

Yday (Julian date), 39
yToEastBerne (astroGeo), 6
YX2LB (astroGeo), 6
YX2MK (astroGeo), 6

zero (functions), 29