Package 'cwhmisc'

August 28, 2018

Type Package

Version 6.6

Date 2018-08-24, 10:40:10

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

Author Christian W. Hoffmann

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

Depends R (>= 2.0), lattice, grid

Description Miscellaneous useful or interesting functions.

URL http://www.echoffmann.ch

License GPL (>= 2)

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-08-28 11:34:25 UTC

R topics documented:

---

cwhmisc-package *cwhmisc*

Description

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

Details

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

Author(s)

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

Examples

```
## Not run:  #  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") )

## End(Not run)
```

adaptlob *Numerically evaluate integral using adaptive rules.*

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,...) allows coefficients P1, ... to be passed directly to the function f: g <- f(x,P1,P2,...)

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
...	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

```
## Not run:
options(digits=7)
FexGander <- function(xx) ifelse(xx < 1,xx+1,ifelse(xx <= 3, 3 - xx, 2 ))
adaptsim(sin,0,pi,2.0e-3,TRUE)$Q - 2.0 #   -1.686905e-05
adaptsim(sin,0,pi,2.0e-23)$Q - 2.0  # 0
adaptsim(FexGander,0,5)$Q - 7.5 # -7.993606e-15 instead of 0
adaptlob(FexGander,0,5,2.0e-6,TRUE) # 7.500002 instead of 7.5
adaptlob(FexGander,0,5,2.0e-6)$Q - 7.5 #  1.781274e-06 instead of 0
adaptlob(FexGander,0,5)$Q-7.5 #   instead of  -8.881784e-16, with warnings
#  that required tolerance is too small.
adaptlob(FexGander,0,5,5.0*.Machine$double.eps)$Q-7.5 #  -5.329071e-15

## End(Not run)
```

`arcs` *Convert and reduce arcs*

Description

Functions for conversions and 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 $0 <= minU_{new}, V_{new} < ref$.

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

See: http://www.kayelaby.npl.co.uk/general_physics/2_7/2_7_2.html
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
cSBYEMY = 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 = 130000000.0 , 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( yt_e, xt_n )
YX2MK( yt_e, xt_n )
```

Arguments

`long,lat`	Real, geogr. longitude, latitude
`yt_e, xt_n`	Real, Swiss coordinates, east, north positive

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 ) # 7.4395833  46.9524055
YX2MK ( 600, 200) #  =   0
```

cap *Functions for 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)
strReverse(str)
```

Arguments

str	a character vector.
ff	a function.
...	additional parameters for function ff.
char	a single letter.
strReverse	the reverse of str

Value

The same type as the argument.

Note

capply has been reverse engineered from the help page on strsplit. strReverse <- function(x) 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

```
IsCounterCl2( U, V, ref )
IsCounterCl3( 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
IsCounterCl2 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
IsCounterCl3 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

Usage

```
GreatestIntAsRealF()
```

Details

c38 := sqrt(c3Q)
c3Q := .Machine$double.xmax^0.75, used for computations below Inf, also
GreatestIntAsRealF Find the greatest integer K which is distiguishable from (K+1), both represented as real
ASCII := ASCII characters corresponding to (0), 1..256,
HexDig := '1' - '9', 'A' - 'F', 'a' - 'f'
HexagesDig := '1' - '9', 'A' - 'Y', 'a' - 'y'
EXPCHAR := "z", exponential marker used for bases other then 10 (for base 10 "e" is used as usual);
TAU := (1+sqrt(5))/2 = golden section constant = 1.6180

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

See Also

r2Be

---

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)
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-'O', with 'O' =c(0,0,0) = origin.
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) <-> polar c(r,phi); phi = angle(x-axis,point).

toSph, toXyz: Rectangular c(x,y,z) <-> spherical coordinates c(r,theta,phi); theta = angle(z-axis,P-'O'), phi= angle[plane(P,z-axis), plane(x-z)].

Value

toPol: c(r, phi), r=Mod(z), phi= Arg(z); Re(z)=x, Im(z)=y
toRec: c(x , y), x=Re(z) , y=Im(z); Mod(z)=r, Arg(z)=phi
toSph: c(r, theta, phi), r=sqrt(x^2+y^2+z^2), theta=atan2(z,v),
phi=atan2(y,x) ; v=sqrt(x^2+y^2)
toXyz: c(x, y, z), x=r*sin(phi)*sin(theta), y=r*cos(phi)*sin(theta), z=r*cos(theta)
rotZ: c(x', y') = rotated (x, y) by angle phi, counter clockwise,
– Rotation matrices:
rotA: Ratation matrix to rotate around axis P - 'O'.
rotV: Ratation matrix to rotate v into w.
rotL: Matrix m for multiplication m %*% 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

rotZ: see toPol angle: uses acos and asin
v %v% w : same as vecprod(v, w)
v %s% w : same as 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(c(x,y,z)) )  # r= 3.7416574, theta= 0.64052231, phi= 1.1071487
  c(R[1],180/pi*(R[2:3])) #  3.741657 36.6992252 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
 ## Not run:
  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])

## End(Not run)
  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, cposV finds the first position of a substring;
cposR returns a list with starting and ending positions, works only with a single string;
issubstr checks if is a substring

Usage

```
cpos(str,sub,start=1)
cposV(vstr,sub,vstart=1)
cposR(str, sub, restrict)
issubstr(str,sub,start=1)
```

Arguments

str	string to examine
vstr	vector of strings to examine
sub	(vector of) substring to find
start,vstart	(vector of) integer, position(s) of start of search
restrict	vector of lower and upper index the search should be restricted to. If missing, whole 'str' is taken.

Value

cpos,cposL,cposV number, if found, NA otherwise.
cposR list(first,last) for each occurrence of sub within the restriction restrict of str; If there is none, then first=NA,last=NA.

Note

parameters in cposV will be recycled, so that all have the same (maximum) length.

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
  cpos(" Baldrian","a",5) #  3
  cpos("Baldrian","B",15) # NA
  cposR(" Baldabcrian  abcf","abc")
#$first  6 15
#$last   8 17
  cposR(" Baldabcrian  abcf","abc",c(2:16))
```

```
#$first 6
#$last  8
  cposV(c("Xcdbeesh","withh "),c("X","h","ees"),c(1,5))
# 1 4 5
issubstr("Today is a wonderful day","wonder")
```

---

cwhmisci *Functions not to be called directliy 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 *Show date and time in ISO format*

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"
```

dcm *Convert number for table columns, for equations*

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 >= 1, will be forced internally.
ch	Substitute "." by ch
after	See formatFix, the number of decimals after ".".
r	real value.

Value

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

Note

dc = dcn, except for `x = 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 thenhave
##   $$z = \Sexpr{sprintf(%.4f,a)} \Sexpr{mpf(b,3)} x \Sexpr{mpf(c,5)} y$$
#### end T.Rnw
###  Sweave: T.Rnw .. T.tex .. T.dvi
```

Ddim *dim of vectors and arrays*

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 *Waiting loop for program execution*

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) # 116596 on my machine (2.33 GHz MacBook Pro)
```

---

delstr *String handling*

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("Don't enter my garden","en")
# -> "Don't ter my gard"
delstr("12345","2") # "1345"

strReverse(c("abc", "Statistics")) #  "cba" "scitsitatS"
```

digits *Test, convert numbers*

Description

Test, convert numbers

Usage

```
allDigits( str, base=10 )
isNumeric(str)
str2dig( str )
```

Arguments

str	Vector of strings
base	Integer, base of number representation used in r2B

Value

allDigits The strings contain digits only which are allowable in base 'base'.
isNumeric Test whether the elements of a character vector represent legal numbers only.
str2dig Convert a string to a vector of integers.

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("13245.") #  1 3 2 4 5 NA
  # for comparison, big numbers:
int(10^(7:10)) # 10000000 100000000 1000000000 NA
```

div.prot *Protected division*

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 *Convert time difference to string.*

Description

Convert time difference in seconds to string depending on switch.

Usage

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

Arguments

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

Value

String representing the time difference, 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) # 00:00:0.070, Macbook Pro 2016, 2.2 GHz, 16GB RAM
```

`ellipse` *Generate ellipses*

Description

Given the axes a, b (major and minor) and angle phi (in radian, counter clockwise from x-axis), and the midpoint c(0,0), points on a rotated ellipse will be generated. The major axis is rotated from the positive x-axis by the angle phi.

Usage

```
ellipseC(mid, a, b=a, ra=c(-1,361), phi=0, k=a*100 )
ellipse1( a, b=a, ra=c(-1,361), phi=0, k=a*100 )
conf.ellipse( a, b, phi, df1, df2, level = 0.95, k)
```

Arguments

`mid`	Complex, center of ellipse
`b`	Real > 0, minor axis
`a`	Real > 0, major axis
`ra`	Integer, range of arc [deg]
`phi`	Real, angle in radian describing the counter clockwise rotation from the x-axis to the axis given by 'a'.
`k`	Integer, the number of generated points on the ellipse.
`df1, df2, level`	degrees of freedom and probability level of F-distribution.

Value

`ellipseC` complex coordinates of the ellipse. `ellipse1` (x,y)-coordinates of the ellipse. `conf.ellipse` (x,y)-coordinates 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(a,b,phi,df1,df2,p[2], k) + m
  plot(el7*1.8,type="n",xlab="Different confidence ellipses",ylab="")
  lines(conf.ellipse(a,b,phi,df1,df2,p[1],60) + m,lty=2,col="red")
  lines(conf.ellipse(a,b,phi,df1,df2,p[3],60) + m,lty=2,col="green")
  lines(conf.ellipse(a,b,phi,df1,df2,p[4],60) + m,lty=2,col="blue")
  lines(el7,lty=2,col="orange")
  leg1 <- paste(as.character(p*100),rep("percent",length(p)),sep="")
#  leg1 <- paste(as.character(p*100),rep("%",length(p)),sep="")
  col1 <- c("red", "orange","green","blue")
  legend(x="bottom",leg1,col=col1,
text.col="black",lty=c(2,2,2,2), merge=TRUE, bg='white', cex=0.9)
par(opar)
for(ii in 0:15){ x <- ellipseC(40,1,2,phi=pi/15*ii);lines(x,col=ii%%3+1)}
```

---

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

Description

eql checks two vectors on equality; two NA's and two NaN's are compared 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 TRUE
```

`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 log10(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 (first row) together with their multiplicities (second row), recombine factors, primitive version of the sieve of Eratosthenes.

Usage

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

Arguments

n	positive integer, number of primes, number to be factored, to be tested
fp	2-columnn 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.
allFactors	generate all factors of n: 1..n.
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 1.578922 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 $abs(x) >= 10^8$, 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`	Real,the number to be represented.
`after`	integer, The number of decimals after ".".
`before`	Integer, the minimum number of decimals before ".".
`extend`	Logical, extend string if necessary.

Value

The string representing the fixed point format of `x`.

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
## Not run:
xxbig <- c(1.2e9,3.51e23,6.72e120,NaN); xx <- c(0.001,92,exp(1),1000*pi)
t(t(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"
t(t(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"

## End(Not run)
```

`frac` *Fractional part of number, continuous fractions*

Description

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

Usage

```
contfrac( x, depth = 13, f=floor )
evalcfr( cf )
toCFrac( x, depth=5)
toCFrac2( x, depth=5)
```

Arguments

`x`	Real
`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
$round(10^d * fractionalpart)$, 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 abs(x - num/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
For confrac see also link[MASS]{fractions}. from Mathematics 5335 Fall 2009 The Euclidean Algorithm and Continued Fractions

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
(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, pythagorean sums, etc.

Usage

```
chsvd( s )
chsvd( s )
divmod( i, n )
divmodL( i, n )
dsm( x, w )
equal( x, y )
equalFuzzy( x, y, prec=8*.Machine$double.eps, rel=TRUE )
exch( x, L, R )
frac(x,d)
int( x )
inrange( x, y )
Ko(z)
Km(z)
last( x )
LE ( x )
loop.vp( ve, overlap=1 )
LS ( )
lV ( x )
mod( x, y )
modR( x, y )
modS( x, y )
norm2( x )
one ( x )
onebyx( x )
powr( a, x )
pythag( a, b )
quotmean( x, y )
safeDiv( num, den )
signp( x )
solveQeq( a, b, c )
sqr( x )
sqrtH( x )
submod( x, v )
zero ( x )
```

Arguments

prec,L,R	Real
a,b,c,z	Complex
i	Integer vector
d	If not missing, 'frac' shows 'd' decimals after "." as integer
n,num,den	Integer
rel	Boolean
s	square matrix, result of svd
v	real vector > 0, preferably cumsum of some other positive vector

ve	real any vector or matrix
overlap	integer vector, giving element indices/column numbers to be appended at the end, see examples
x,y	Real vector
w	Real vector > 0

Details

BEWARE of NAs !!
chsvd Check for svd to reproduce matrix.
divmod rbind(div, mod) for ease of use.
divmodL list(d = div, m = mod)
dsm combination of divmod and submod, used in Jul2Dat
equalFuzzy One can choose between relative and absolute precision
equal x == y, of same length.
inrange Check if 'x' (scalar) is in the range (min(y),max(y)).
int returns 'x' as integer in fix format
last return the last element of a vector.
LE short for 'length(x)'.
LS short for '.Last.value'.
loop.vr: loop around vector with overlap.
LS short for '.Last.value'.
modS: same as 'mod', symmetric to 0.
mod = x %% y, x and y with same number of elements.
onebyx = 1.0/x
one returns 1.0, same length as 'x'
powr = x^y, with 0^0 := 1, 0^y := 0, any y
quotmean Compute quotient of means of non-NA elements of x by y
safeDiv Compute quotient, set 0/0 -> 1, and safeguard r/0 <- c3Q otherwise
signp(0) -> 1, signp(complex) -> NA !
solveQeq Solve the quadratic equation a*z^2+ b*z+c given
the coefficients a,b,c OR c(a,b,c), returns always *two*
solutions; if a = b = 0 returns c(Inf, Inf)
sqr = x^2
submod analog to divmod for unequally spaced data, c(greatest index gi of v s.t. v < x, x - v['gi']
zero returns 0.0, same length as 'x'

Value

exch: Exchanges elements 'L' and 'R': x[which(x == L)] <- R; x[which(x == R)] <- L
K: Cayley transform (z - i)/(z + i)
Km: (1 + z)i/(1 - z), inverse transformation of K norm2: 2-norm.
last: last element of vector LE: = length of vector pythag: c(A,B,C), A=final a' = sqrt(a^2 + b^2) without squaring and
taking the square root, avoiding overflow and underflow, B=final b',
C=residual = final (b'/a')^2, see note.
signp: ifelse(is.na(x) | (!is.finite(x) | x>=0),1,-1), avoiding

NA, NaN and 0 in the result.
`sqrtH`: Square root with Halley's hyperbolical method.

Note

see also examples of `date`
Note that `1` results with `signp( 0 )` ;
It is *not* possible to discriminate between `Inf` and `-Inf`, by definition in R,
but: as.character(-Inf) = "-Inf". `pythag`: The invariant of the iteration is sqrt(a^2 + b^2), iterating a':=max(a,b) and reducing b':=min(a,b).

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

References

Moler, C. and Morrison. D, 1983 *Replacing Square Roots by Pythagorean Sums*, IBM J.Res.Devel., **27**, 6, 577–589.
jjam(Prime numbers); Mathpath(Formula for Halley); Wikipedia(Derivation of Halley)

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
y <- c( Inf, -Inf,NA,  NaN, -NaN,-1, 0, 1 )
signp(c(-1:1,NA,NaN,Inf, -Inf)) # -1   1   1  1  1     1  1
#  instead of sign() =            -1   0   1  NA NaN   1  -1
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
loop.vp(1:4) #  1 2 3 4 1
loop.vp(matrix(1:12,nrow=3),c(2,4))
#     [,1] [,2] [,3] [,4] [,5] [,6]
#[1,]    1    4    7   10    4   10
#[2,]    2    5    8   11    5   11
#[3,]    3    6    9   12    6   12
safeDiv(0:3,c(0,0:2)) # 1.552518e+231
signp(c(-1:1,NA,NaN,Inf, -Inf)) # -1   1   1  1  1     1  1
#  instead of sign() =            -1   0   1  NA NaN   1  -1
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
exch(LETTERS, "A", "Y") # "Y" "B" ... "W" "X" "A" "Z"
exch(1:5, "2", "Y") # "1" "Y" "3" "4" "5"
pythag(19,180) # 1.8100000e+02 3.8414499e-23
```

---

ggrep *Convenience functions for grep*

Description

Grepping in a (my) R directory

Usage

```
grepnot(str,x,value=TRUE)
ggrep( opt = "inr", str, exclude = "", dir =  "/Users/hoffmannc/R/",
           pkg = "", split = FALSE, lines = 10, out = FALSE)
countChar( str, dir="/Users/hoffmannc/R/", pkg="",split=FALSE,out=FALSE)
```

Arguments

str	string to do grep for
exclude	string to exclude from grep
x	array of strings to check with grep.
value	Boolean, third argument to grep ('ignore.case').
opt	options for 'grep' without leading '-'
dir	name of root directory to do grep in
pkg	package name to do grep in; may be "" if dir itself is meant.
split	should str be split in single characters? If so, only unique characters will be searched.
lines	a maximum of 'lines' lines will be returned
out	logical, should intermediate results be printed

Value

ggrep	grep output, with line numbers and pertaining line, or "No file with given string found".
grepnot	grep output, combination of invert=FALSE and =TRUE.
countChar	count the individual (if split=TRUE) characters in str

Note

ggrep, countChar: both use system(grep ...)
The composed file string and the input string to grep are shown for checking.
length(ggrep()) shows number of found entries only.
If file denotes a directory,and no "r" is given, then opt : contains e.g.
- "c": count lines in same one file only,
- "i": ignores case,
- "n": give file and line number,
- "r": recurse below directory one level (only),
- "v": inverts matches,
- "w": complete word matches only,
- "x": matches must be whole lines only

ggrep("-v","xxx",exclude="") is the same as
ggrep(,"",exclude="xxx"), except that the former lacks line numbers.

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
## Not run:
  length(ggrep("cnr","pad ",,"test*/*") )   # (dir), 10 files, not shown
  ggrep("cnr","pad ",,"test*/*")     # is dir, 10 files visited
  ggrep("cr","n2str",,"test/")
# /Users/hoffmannc/R/test/  may be a directory
# grep: /Users/hoffmannc/R/test/: No such file or directory
# NA

## End(Not run)
```

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.

Note

Halton, J.H., Smith, G.G., 1961. Algorithm 247 Radical-inverse quasi-random point sequence Computes a sequence of N quasi-random points lying in the K-dimensional unit cube given by 0 < x_i < 1, i = 1,2,..,K. The i-th component of the m-th point is stored in Q[m,i]. The sequence is initiated by a zero-th point stored in P, and each component sequence is iteratively generated with parameter R[i]. E is a positive error-parameter. K, N, E, P[i], R[i], i=1..K, are to be given.

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,n,c(2,2),c(0,pi/10))
q2 <- HS247(2,n,c(2,3))
q3 <- HS247(2,n,c(2,5))
q4 <- HS247(2,n,c(17,19)) # prone to correlations
q5 <- HS247(2,n,c(2,3),c(pi/10,pi/10))
of <- 0.2
q6 <- HS247(2,n,c(2,3),c(pi/10+of,pi/10+of))
## Not run:
  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="-")

## End(Not run)
```

---

hours *convert hours*

Description

Functions for conversion of hour representations

Usage

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

Arguments

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

Value

Hdhours Hmsc(h,m,s) Hdmshh.mmss Hmsd(Decimal) hours

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
Hd( 12,25,17) # 12.421389
Hms(1.421389) # 1h 25m 17.0004s
Hmsd(12.421389) # 1h 42m 13.89 -> 12.703858 h
Hdms(12.703858) # 12.421389 h
```

int2 *convert integers, string to integer vector*

Description

Functions for conversion to string representation of integers to arbitrary bases

Usage

```
NdM( x, B=10 )
int2ASCII( n)
int2B( n, B=10, space, plus=lead, lead="",just=c("left","right","center","none") )
int2Oct( n )
int2Hex( n )
strRound( str, digits = getOption("digits"), B=10)
```

Arguments

str	String representing a real
n	Integer vector
B	1 < integer < 17, base of representation
space	Integer, space for conversion
plus	string for signifying positive values, usually "" or "+"
lead	string for insertion between sign and first significant digit, usually "" or "0"
just	String for choosing kind of justification within 'space', partial matching allowed
x	Vector of reals
digits	no. of digits for roeunding

Details

int2Oct Convert integer to octal representation.
int2Hex Convert integer to hex representation

Value

NdM maximum number of decimal places needed for trunc(x)
int2ASCII,int2B,int2Oct,int2Hex vector of strings represented by 'n'
strRound real, represented by x

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
NdM(10^(1:4)) # 5
int2ASCII(1:255)[121:129] # "x" "y" "z" "{" "|" "}" "~" "\177" "\200"
int2B(1:50,2) # all of same length
int2B(1:50*(-1)^(1:50),just="r") # left flush
unlist(sapply(1:50,int2B,2,just="l")[1,] ) # individual lengths
unlist(sapply(1:50,int2B,7)[1,] ) # individual lengths
unlist(sapply(1:50,int2B)[1,] )
unlist(sapply(1:50,int2Oct)[1,] )
unlist(sapply(1:50,int2Hex)[1,] )
strRound(pi*10^4,0)/10^4 == strRound(pi,4) # TRUE
```

`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)
lerp(p1, p2, t)
```

Arguments

`x`	vector of x-coordinates
`y`	vector of y-coordinates
`xi`	argument x of interpolation
`p1,p2`	point coordinates for linear interpolation
`t`	0 <= t <= 1, linear interpolation distance
`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 minimumm exists in `x`.
`lerp`	linerly interpolated point, t=0 -> p1, t=1 -> p2

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@acelerate.com>, expanded by Christian W. Hoffmann <christian.hoffmann@wsl.ch>

See Also

identical, all.equal

Examples

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

---

jitterNA *Jitter vector containing NA*

Description

Extension of jitter to deal with NA entries

Usage

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

Arguments

x	Data to be jittered, may be vector, matrix, or numerical data frame.
...	Other parameters for jitter.

Value

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

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

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

`Julian date` *calender conversions*

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 given year a leap year ?
`dk`	integer, day in year

Value

`Dat2Jul:` JD, year year BC is to be given as -|year-1|, e.g. 4 BC = -3, 1 BC = 0 !!
`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

```
c(Jul2Dat(dd <- Dat2Jul( -4712,1,1) ) )
   # -4712, 1, 1, 12;  JD=0  i.e. Start of Julian day numbering
c(Jul2Dat(dd <- Dat2Jul( -1, 1, 1)),dd)
   #    -1,  1,  1, 12; JD=1720693 , start of last year BC
c(Jul2Dat(dd <- Dat2Jul( -1,12,31)),dd)
   #    -1, 12, 31, 12; JD=1721057 , last day BC
c(Jul2Dat(dd <- Dat2Jul( -0, 1, 1)),dd)
   #     0,  1,  1, 12; JD=1721058 , first day AD
c(Jul2Dat(dd <- Dat2Jul(  1, 1 ,1)),dd)
   #     1,  1,  1, 12; JD=1721424
c(Jul2Dat(dd <- Dat2Jul( 1582,10, 4 )),dd)
   #  1582, 10,  4, 12; 2299160, last day of Julian calendar
c(Jul2Dat(dd <- Dat2Jul( 1582,10,15)),dd)
   #  1582, 10, 15, 12; 2299161, first day of Gregorian calendar
round(c(Jul2Dat(dd <- Dat2Jul( 1582,10,15, 0.0168)),dd),1 )
   #  1582, 10, 15, 12; 2299160.5 first day of Gregorian calendar
c(Jul2Dat(dd <- Dat2Jul( 2001,1,1)),dd)
   #  2001,  1,  1, 12; 2451911
 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

```
 ## Not run:
  libs()
  libs(base)

## End(Not run)
```

`lowess.bygroup` *Plot data in groups, each group with separate lowess smoothing*

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 (non-)functions in the current work space.

Usage

```
ls.functions()
ls.notfunctions()
```

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, mfcol,
         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

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

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

```
## Not run:
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))))

## End(Not run)
```

---

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() #
```

NA2str *Convert NA, NaN, Inf to a string*

Description

Conversion of indefinite values

Usage

```
NA2str( x )
```

Arguments

x	A numerical vector.

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
NA2str(c(NA,NaN) ) # "NA"  "NaN"
```

---

normalize *base power and multiplier, significant places*

Description

Split a number into base power and multiplier so that x = a*base^e, with abs(a) in [1, base); check normalization; compute number of significant places

Usage

```
normalize( x, base=2 )
checkNormalize( no )
Nd(x, base=10)
sigplaces(x, base=10, rnd=0)
checkNormalize( no )
```

Arguments

x	Real vector
base	Base of power
no	result of normalize
rnd	Integer >0 / <0, rounding to r digits after/before "."

Details

normalize(c(+-Inf, NA)) will result in c(+-Inf,NA,1).

Value

`normalize:` data-frame with one column c(a,e,base) for each x, such that $x = a*base^e$, abs(a) in *[1, base)*, but *a:=x, e:=0* for *x=0, NA, +-Inf*.
`normalize1:` as `normalize:`, but abs(a) in *[1/base, 1)*
`Nd:` log to base `base`, *1 for x=0* .
`sigplaces:` number of places necessary for printing trunc(x); c(2,3,4,3) for c(NA,Inf,-Inf,NaN).
`checkNormalize:` reconvert argument to number.

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
(xx <- c(exp(1),pi,NA, Inf, -Inf,10,100,c(1,10)*exp(1)) )
(x2 <- normalize(xx,2))
#          A         B  C   D   E    F      G         H         I
# a 1.3591409 1.5707963 NA Inf Inf 1.25 1.5625 1.3591409 1.6989261
# e 1.0000000 1.0000000  0   0   0 3.00 6.0000 1.0000000 4.0000000
# b 2.0000000 2.0000000  2   2   2 2.00 2.0000 2.0000000 2.0000000

(x32 <- normalize1(xx,2))
#           A          B   C   D   E     F       G          H         I
# a 0.67957046 0.785398 NA Inf Inf 0.625 0.78125 0.67957 0.849463
# e 2.00000000 2.000000  1   1   1 4.000 7.00000 2.00000 5.000000
# b 2.00000000 2.000000  2   2   2 2.000 2.00000 2.00000 2.000000

(x10 <- normalize(xx,10))
#           A          B  C   D   E  F  G          H          I
# a  2.7182818  3.1415927 NA Inf Inf  1  1  2.7182818  2.7182818
# e  0.0000000  0.0000000  0   0   0  1  2  0.0000000  1.0000000
# b 10.0000000 10.0000000 10  10  10 10 10 10.0000000 10.0000000

(x7 <- normalize(xx,7))
#           A         B  C   D   E         F         G         H         I
# a 2.7182818 3.1415927 NA Inf Inf 1.42857 2.0408 2.71828 3.8832598
# e 0.0000000 0.0000000  0   0   0 1.00000 2.0000 0.00000 1.0000000
# b 7.0000000 7.0000000  7   7   7 7.00000 7.0000 7.00000 7.0000000

sigplaces(-9.999) #
sigplaces(pi/100) #

all.equal(checkNormalize(x2), checkNormalize(x7)) # TRUE
```

`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, xe-y -> $x . 10 ^-y$

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 = gcd(m,n) is satisfied. `Inv` Modular inverse in a finite ring, NA if not exists. `modexp` Exponentiation a^b mod n using repeated squaring via binary decomposition of exponent.

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

References

modexp: http://mvngu.wordpress.com/2008/08/01/parigp-programming-for-basic-cryptography/

Examples

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

padding *Padding a string with justification, insertion*

Description

Pad a string, insert substring.

Usage

```
pad(str, space,just = c("right", "left", "center", "none"), with=" " )
justify(str, space, just = c("right", "left", "center", "none"), with=" ")
insstr(str, ins, point=nchar(str) )
```

Arguments

str, ins	String to be modified, to insert.
space	Integer, resulting length of padded string.
just	Mode of padding, of justification, one of "left","right","center", partial matching is allowed. If missing, "right" is taken, meaning for pad(just="r") right-ways extended (i.e. flush left), for justify(just="r") right-justified; "none" returns str unchanged.
with	String to pad with, will be repeated as often as necessary.
point	Integer, place of insertion. Appending is done for default value.

Value

pad, justify: The string padded with with.

insstr The string contained in ins inserted after character number point of str.

Note

pad(just="r") right-ways extended (i.e. flush left),

justify(just="r") right-justified,

just="none" returns str unchanged.

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
pad("My string",25,"c","XoX")
 # [1] "XoXXoXXoMy stringXXoXXoXX"
pad("My string",25) # left aligned
(str <- paste00(LETTERS)) # "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
pad(str, 37, "right", " $ ")
insstr(str," $ ",7) # "ABCDEF $ GHIJKLMNOPQRSTUVWXYZ"
```

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"))
```

parsecheck *check files for parsing errors*

Description

check files for parsing errors

Usage

```
parsecheck(str)
```

Arguments

str	Directory containing *.R files to examine

Value

file name and place where parsing error occurred; mostly missing brackets/braces

Author(s)

Duncan Murdoch via "unable to collate and parse R files"

`paste00` *paste with collapse=""*

Description

`paste00` is defined as paste0(..., collapse="").

Usage

```
paste00(...)
```

Arguments

...	list of items to paste, coerced to string

Value

pasted strings using collapse="".

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
# Note the differences:
  a <- 1:2;  b <- 3:5
  paste  (a,b)             # "1 3" "2 4" "1 5"
  paste0 (a,b)             #  "13" "24" "15"
  paste00(a,b)             #  "132415"
  paste0 (a,b,c=";")       #  "13;" "24;" "15;"
  paste  (a,b,s="-")       #  "1-3" "2-4" "1-5"
  paste  (a,b,s="-",c=";") #  "1 3 - ;" "2 4 - ;" "1 5 - ;"
  paste00(0:9) # "0123456789"
  paste00(LETTERS) # "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

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.parent(substitute(data))))
```

Arguments

m	Numerical matrix
...	Additional arguments for text
tit	Overall title for plot. A vector of one or two elements. If an element is an expression, plotmath will be used.
rows	Number of rows of panels
cols	Number of columns of panels
f	A function to plot the individual plot panels. It can also be a statement sequence {...}
cex	Font size used for tit
reset	Should previous rows, cols be restored after execution. See note
outer	Passed on to mtext.
oma	Outer margin used in initial par(...).
mar	Lines of margin used in initial par(...).
data	Matrix or dataframe containing data, varibles in columns
breaks	Type of 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

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 splom, [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;
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$Q + outer(rep(1,nrow(A)),pf$tr))  ## !!!!!!  %*% 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` *Print without square brackets, expression values together with their call strings*

Description

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

Usage

```
catn(...)
catE(...)
prinE(...,digits=4)
prinV(x,after=2,before)
prinM(x,after=2,before)
prinT(x,rownam=FALSE,colnam=FALSE)
prinP(xs)
```

Arguments

`...`	See 'note'.
`x`	A numerical vector or matrix.
`digits`	Integer, number of digits, see `print`
`before`	Integer, the number of decimals before "."
`after`	Integer, the number of decimals after ".".
`rownam`	Logical, should row names be printed.
`colnam`	Logical, should column names be printed.
`xs`	A string representing an expression.

Note

`-catn()` is shorthand for cat("\n") which is awkward for me to type.
`-catE,prinE` print string expressions ... and their evaluation in the form "expression = (newline) evaluation", in vector form.
`-catE` is like 'prinE', but can handle annotating (non-variable) strings, given as starting with '\t'. If line feed is wanted, start with '\n'. It *cannot* handle matrices.

-prinP prints a string argument and evaluates it i.e. the body of the function evaluated should contain print and cat statements.
-prinV prints a vector without [], in fix format.
-prinM prints a matrix without [], in fix format.
-prinT prints an array, TAB delimited.
The variants N... prepend a linefeed.

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

See Also

formatFix

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   OK
# 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

prinT(c(c1=5,c2=7,c3=1,c4=3),TRUE)
# 5 7 1 3
 opt <- options(digits=3)
prinE("x")
prinE("'This is a comment: ';3+5;pi-3",digits=4)
prinE("x")
# x =    c1    c2     c3
# r1  5 7.000  3.142
# r2  3 8.235  0.000
# r3  2 2.718 99.000
```

```
catt <- function(x) {cat(paste0("This function 'catt' will write '",x,"' on one line\n")) }
y <- prinP("catt(32)");
# catt(32)
# This function will write ' 32 ' on one line

prinE("y ")
# y  = NULL

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) = Prob(X <= 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 <= x), otherwise, P(X > x).

Value

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

Warning

If p <= 0 or p >= 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
```

---

r2B *Conversion of real to string and rounding, in given base*

Description

Functions for conversion of real to string and back, in given base, in fixed and exponential format; and rounding in base number system.

Usage

```
r2B(x, base = 10, rnd = 0, space = 0, plus = "", lead = "",
  just = c("right","left","center","none") )
r2Be(x, base = 10, space = 4,plus = "+",
  just = c("right","left","center","none") )
roundB( x, base=10, rnd=0 )
strB2r( STR, base=10 )
strB2i( STR, base=10 )
```

Arguments

x	Real or integer, vector
STR	Vector of strings representing reals in a given base
base	2 <= integer <= 60, base of representation
space	Integer, space for resulting string; if too small, only necessary space will be taken. All components of the result will be of common length, justified according to 'just'.
rnd	Integer, number of places (after ".") to be rounded; = 0: rounded integer, no decimal point; = 0.5: rounded integer "." no following digits shown; < 0: rounding 'rnd' places *before* last integer digit; If too negative, 0 will result; see examples; 'rnd' prevalent over 'space', i.e. space will be expanded if necessary.
plus	use "+" to show sign for positive values.
lead	use "" or " "; or "0" for leading zeros; this will be inserted after sign.
just	Choice of insertion of justification, can be abbreviated, see justify

Value

roundB: vector of arguments rounded to rnd places according to base base representation

r2B: list(s=vector of strings represention of x, rounded to rnd decimal digits, base=base)

r2Be: like r2B, but in exponential representation, if space too small. The exponent marker is "e" for base==10 and EXPCHAR otherwise. strB2r: real corresponding to string representation. NaN is returned if str contains characters not in HexagesDig[1:base] (as are generated by r2B, r2Be).

strB2i: integer (str2i) corresponding to string representation, used in strB2r

Note

r2B(.) and strB2(.) are inverses of each other.

r2Be chooses between fixed and exponential format depending on available space, adjusting rounding accordingly. just="left" works best with lead = "" = default.

strB2r can convert strings with exponent signifyers "e" (for decimals) and "z" for others, "." is allowed for fractional parts.

strB2i works on strings *without* "e","z","." ONLY!

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
  x <- c(0,-0.0012345, 1.5234, 543, 8123456,NA,Inf,-Inf,NaN,1,pi)
  y <- c( 0, 1, pi*10^c(-27,-8,0, 8) )
# "+3.1e27" "    NA " "   NaN " "   Inf   )
r2B(x,,space=10,lead=" ",plus="",rnd=3)$s
```

```
# "          0 " "-      0.001" "       1.523" "     543.000"
# " 8123456.000" "          NA" "         Inf" "        -Inf"
# "         NaN" "       1.000" "       3.142"
  r2B(x,16,space=10,lead=" ")$s
# "         0 " "-        0" "         2" "       21F"
# "    7BF440" "        NA" "       Inf"" "      -Inf"
# "       NaN" "         1" "         3"
  r2B( x, 60, 4 )$s
  r2Be( y,10,7)$s
# "     0 " "+1.0000" "+3.e-27" "+3.1e-8" "+3.1416" "+3.14e8"
# "+3.1e27" "    NA " "   NaN " "   Inf "
  strB2r("-    9167.8  4",10)
  strB2r("B00z3",15 )  # 8353125
  (ii <- r2B( 8353125, 32, 4 )$s) # "7UTB5.0000"
  strB2r( ii, 32)  # 8353125
  roundB(c(0.4,0.3),2,16) # 0.39999390 0.30000305
```

RCA *Check, build, install package in a unified manner.*

Description

Check, build, install package in a unified manner.

Usage

```
RCA(dir=getwd(),pkg, Rsty, sw=c(2, 5:7), echoonly=FALSE, verbose=TRUE)
```

Arguments

dir	character, dirname of package(s).
pkg	character, basename of package
Rsty	full path name of 'Rd.sty'
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 0:7, see note

Note

If the complete filepath of the package source is given in 'dir', 'pkg' must be *empty* !

"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 –no-build-vignettes",

- 4 = "check –as-cran <pkg>.tar.gz",
- 5 = "check –as-cran",
- 6 = "install"
- 7 = "Sweave ", "/vignettes/",".Rnw"

The order 2 to 6 is suggested by https://cran.r-project.org/doc/manuals/r-release/R-exts.pdf.
sw = 1 shows errors present in the creation of the manual.
sw = 4 is provided for checking as required by CRAN policy.
'Rd.sty' must be provided in `Rsty`, mine is in `/Users/hoffmannc/Rtest/Rd.sty`. Permissions for the vignette *.Rnw should be changed by system("chmod u=rwx ...Rnw "), if necessary.

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

`remove.dup.rows` *Remove duplicate rows*

Description

Removes duplicate rows from a dataframe.

Usage

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

Arguments

`dfr`	A dataframe

Details

Uses the function eql.

Value

The dataframe with only one copy of identical rows.

Note

Method: Sort the dataframe, figure out which rows have all values identical to their successor. This gives logical vector, in the order of the sorted values, so reorder it. Finally select nondups. As a "bonus feature", I think this will also remove any row containing all NA's...

A major stumbling block is that you'll want two NAs to compare equal, hence the eql() function.

Actually, I think you can do away with the isdup array and do

all.dup <- do.call("pmin", lapply(dfr[o,], function(x) eql(x,c(x[-1],NA))))

and there may be further cleanups possible.

One dirty trick which is much quicker but not quite as reliable is

dfr[!duplicated(do.call("paste",dfr)),]

(watch out for character strings with embedded spaces and underflowing differences in numeric data!)

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 *Replace a character in a string by another*

Description

replacechar replaces a character in a string by another, deprecated!

Usage

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

Arguments

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

Value

The altered string.

Author(s)

Christian W. Hoffmann <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 *Generate the significance codes as in summary.lm*

Description

Generate the significance codes as in summary.lm

Usage

```
scode(p)
```

Arguments

p	Probability

Value

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

Note

lifted from stats::printCoefmat

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

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

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

Description

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

Usage

```
select.range(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 in the case where 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
```

`sets` *set inclusion*

Description

Check whether one set is included within another.

Usage

```
setincl( x, X )
```

Arguments

`x, X`	sets.

Value

`TRUE`, if set `x` is contained in set `X`.

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
  setincl(2:3, 1:7) # TRUE
# compare this to:
  grep( "15926",as.character(pi)) == 1 # TRUE
```

`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
```

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

Description

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

Usage

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

Arguments

`d`	kernel density estimate

Value

The spline function approximating the df.

Author(s)

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

Examples

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

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

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

SplomT *splom with title and time stamp*

Description

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

```
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

hist.col	string, color of the histogram; like "(hash)ffccff"
data	Matrix or dataframe containing data, variables in columns
mainL	Label on top of scatterplot matrix or matrix of histograms
xlabL	Label for x-axis
hist	"h" = histogram, "d" = density curve, "b" = both
adjust	factor to adjust smoothing window for density curve
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

This function is called for its side effect to produce a plot.

Author(s)

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

Examples

```
nc <- 8 # number of columns
nr <- 250 # number of rows
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)
```

`str2B` *round real in string*

Description

Function for rounding real given as string representation

Usage

```
str2B(str, base=10, round = 0)
```

Arguments

`str`	String representing a real
`base`	1 < integer < 17, base of representation
`round`	Integer, number of places after "." to be rounded; < 0: rounding places before least significand digit. If too negative, 0 will result.

Value

`str2B` from given string represention of `x`, round to 'round' decimal digits

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```
  x <- paste0("-    ","9167.8")
  str2B(x )
  for ( kk in -5:4) print(str2B(x,10,kk) )
# 0 -10000 -9000 -9200 -9170 -9168 -9167.8 ...
```

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,"Mixture,rather well normal")
T3plot(rnorm(1000)+runif(1000)*10,"Not < 1 percent error for normality")
```

tex.table *Convert a data matrix into LaTeX code.*

Description

These functions convert a data matrix into LaTeX.

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

Value

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

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

`triplot` 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 *Wait for <Return>*

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

Description

whole checks an array whether it consists of whole, i.e. integer , numbers only (x in I).

Usage

```
whole(x)
```

Arguments

x	A numerical array.

Value

TRUE, FALSE

Author(s)

Bill Venables adapted by Christian W. Hoffmann <christian@echoffmann.ch>

Examples

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

Index