Package ‘cwhmisc’

September 9, 2017

Version  6.5
Date     2017-08-18, 15:10
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)
NeedsCompilation no
Repository CRAN
Date/Publication 2017-09-09 19:36:04 UTC

R topics documented:

adaptlob .................................................. 3
arcs .......................................................... 4
astroC ....................................................... 5
astroGeo .................................................... 6
cap ........................................................... 7
clean.na ..................................................... 8
clocksense .................................................. 9
Const ......................................................... 10
coords ....................................................... 10
cpos .......................................................... 13
cwhmisc ..................................................... 14
cwhmisci .................................................... 15
datetime .................................................... 15
dcm ........................................................... 16
Ddim .......................................................... 17
delayt ......................................................... 18
### R topics documented:

- `delstr` .......................................................... 19
- `digits` ......................................................... 19
- `div.prot` ...................................................... 20
- `dt2str` .......................................................... 21
- `ellipse` ......................................................... 22
- `eql` .............................................................. 23
- `f.log` ............................................................. 24
- `factor` ........................................................... 25
- `FinneyCorr` .................................................... 26
- `formatFix` ...................................................... 27
- `frac` ............................................................. 28
- `functions` ...................................................... 29
- `ggrep` ........................................................... 32
- `Halton` .......................................................... 34
- `hours` ........................................................... 35
- `int2` ............................................................. 36
- `interpol` ......................................................... 37
- `invgauss` ....................................................... 39
- `is.constant` .................................................... 40
- `jitterNA` ........................................................ 41
- `Julian date` ..................................................... 41
- `libs` .............................................................. 43
- `lowess.bygroup` ................................................ 43
- `lpr` ............................................................... 44
- `ls.functions` ................................................... 45
- `mult.fig.p` ..................................................... 45
- `my.table` ......................................................... 47
- `n22dig` ........................................................... 48
- `n2c` ............................................................... 49
- `NA2str` .......................................................... 50
- `normalize` ...................................................... 50
- `num.ident` ...................................................... 52
- `num2Latex` ...................................................... 53
- `numberof` ....................................................... 53
- `numer` ........................................................... 54
- `padding` ........................................................ 55
- `panel` ........................................................... 56
- `parsecheck` ..................................................... 56
- `paste00` ........................................................ 57
- `pasteRound` .................................................... 58
- `plotSymbols` ................................................... 58
- `plt` .............................................................. 59
- `pointfit` ......................................................... 61
- `printP` ........................................................... 63
- `progress.meter` ................................................ 65
- `qnorm.appr` .................................................... 66
- `r2B` ............................................................... 68
- `RCA` ............................................................. 70
adaptlob

**Description**

`adaptlob` and `adaptsim` 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**

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


References


Examples

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

Description

Convert and reduce arcs

Usage

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

\( U, V, \text{ref}, \text{radian}, \text{degree} \)

Real

Details

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

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

\[
\begin{align*}
\text{deg(pi/2)} & \quad \# \; 90 \\
\text{rad(180)} & \quad \# \; 3.141593 \\
\text{reda(580,360)} & \quad \# \; -140 \\
\text{reda2(200,120,70)} & \quad \# \; 130, \; 50 \\
\text{reda2(100,-200,70)} & \quad \# \; 310, \; 10
\end{align*}
\]

astroC  Astronomical constants

Description

Astronomical constants

Details

\[
\begin{align*}
cJDJ2000 & = 2451545.0 \; , \; \text{Julian day number of the epoch J2000.0} \\
cDAYPJULCENT & = 36525.0 \; , \; \text{days per julian century} \\
cDAYPYEARTROP & = 365.242198781 \; , \; \text{days per tropical year} \\
cDAYPYEARSID & = 365.25636042 \; , \; \text{days per sidereal year} \\
cDAYPMONSYN & = 29.53058868 \; , \; \text{days per synodical month} \\
cDAYPMONSID & = 27.321655 \; , \; \text{days per sidereal month} \\
cK & = 0.01720209895 \; , \; \text{Gravitational constant, GAUSSian definition} \\
cC & = 299792458.0 \; , \; \text{[m/s] defined speed of light} \\
cRE & = 6378140.0 \; , \; \text{[m] radius of earth's equator} \\
cMY & = 0.0123999576 \; , \; \text{ratio mass of moon/mass of earth} \\
cPRECESS & = 5029.0966 \; , \; \text{[arc sec] precession per year at 2000.0} \\
cEPSOB & = 23.43929111 \; , \; \text{[deg] inclination of ecliptic at 2000.0} \\
cAE & = 1.49597870E11 \; , \; \text{[m] distance Earth to Sun} \\
cBYE & = 332946.0 \; , \; \text{ratio mass of Sun/mass of Earth} \\
cSBYEMY & = 328900.5 \; , \; \text{ratio mass of Sun/mass of, Earth+Moon}
\end{align*}
\]
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
cSIDEYSOL = 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**

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

**Arguments**

```
long, lat, yToEast, xToNorth
```
Real

**Details**

LB2MK From geogr. longitude and latitude to planar meridian convergence [gon].
LB2YX From geogr. longitude and latitude to Swiss coordinates.
YX2LB From Swiss coordinates to geogr. longitude and latitude.
YX2MK From Swiss coordinates North and East to planar meridian convergence [gon].
LongBerne, LatBerne geogr. coordinates of Berne, 7deg26'22.50" east, 46deg57'08.66" north.
yToEastBerne, xToNorthBerne Swiss topo coordinates of reference 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( 9.13258291336895, 46.1866942048755 ) # somewhere in Switzerland, 1.37472
LB2XY( LongBerne, LatBerne ) # 600.0, 200.0
YX2LB ( yToEastBerne, xToNorthBerne ) # 7.4395833 46.9524055
YX2MK ( 600, 200 ) # = 0
```

Functions for strings

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.
clocksense

Examples

\begin{verbatim}
x <- matrix(c(1,NA,2,5),2,2)
clean.na(x,1)
\end{verbatim}

\begin{verbatim}
#    [,1] [,2]
# [1,]  1  2
clean.na(x,2,TRUE)
# [1] 2 5
\end{verbatim}

clocksense Functions for directed arcs

Description

Functions for clocksense, i.e. directed arcs

Usage

\begin{verbatim}
iscounterclH uL vL ref I
iscounterclSH uL vL wL ref I
clocksenseRH uL vL ref I
clocksenseSH uL vL wL ref I
\end{verbatim}

Arguments

\begin{verbatim}
ref, uL vL wL      Real
\end{verbatim}

Details

\begin{verbatim}
CounterClock, NoneClock, Clockwise = "clkws","Cntclck","noneclck","clkws"
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.
\end{verbatim}

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

\begin{verbatim}
ClockSense2(0,220,360) # "clkws"
ClockSense2(0,170,360) # "Cntclck"
ClockSense2(0,0,360)   # "noneclck"
\end{verbatim}
### Description

**Constants**

**Details**

\[ c_{3Q} := .Machine\$\text{double.\text{xmax}}^{0.75}, \text{used for computations below Inf, also} \]
\[ c_{38} := \sqrt{c_{3Q}} \]

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**

\[ \text{toPol}(x, y=0) \]
\[ \text{toRec}(r, \phi=0) \]
\[ \text{toSph}(x, y, z) \]
\[ \text{toXYZ}(r, \text{theta}, \phi) \]
\[ \text{rotZ}(x, y, \phi) \]
\[ \text{rotA}(\phi, P=c(0,0,1)) \]
\[ \text{rotV}(v, w=c(0,0,1)) \]
\[ \text{rotL}(\phi, k=1, m=2, N=3) \]
\[ \text{getAp}(M) \]
\[ \text{angle}(v, w) \]
\[ \text{scp}(v, w) \]
\[ \text{vecprod}(v, w) \]
\[ v \text{ \%v\%} w \]
Arguments

\(x, y, z, r, \theta, \phi\)

Real, rectangular, spherical coordinates; \(x, y, z\) may be combined as \(c(x,y,z)\), and \(r, \theta, \phi\) as \(c(r,\theta,\phi)\)

\(P\)

\(c(x,y,z)\), coordinates of point or projection direction \(P = '0'\), with \(0' = c(0, 0, 0) = \text{origin.}\)

\(v, w\)

3-vectors \((x, y, z)\).

\(N\)

Order of the square rotation matrix \(\geq 2\).

\(k, m\)

Integers \((m \neq k)\) describing the plane of rotation. \(m==k\) gives Unit matrix.

\(M\)

3x3 rotation matrix.

Details

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

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

Value

**toPol**: \(c(r, \phi), r=\text{Mod(z)}, \phi=\text{Arg(z)}; \text{Re(z)}=x, \text{Im(z)}=y\)

**toRec**: \(c(x, y), x=\text{Re(z)}, y=\text{Im(z)}; \text{Mod(z)}=r, \text{Arg(z)}=\phi\)

**toSph**: \(c(r, \theta, \phi), r=\sqrt{x^2+y^2+z^2}, \theta=\text{atan2(z,v)}, \phi=\text{atan2}(y,x); v=\sqrt{x^2+y^2}\)

**toXYZ**: \(c(x, y, z), x=r*\sin(\phi)*\sin(\theta), y=r*\cos(\phi)*\sin(\theta), z=r*\cos(\theta)\)

**rotZ**: \(c(x', y') = \text{rotated}(x, y) \text{ by angle } \phi, \text{ counter clockwise,}\)

- Rotation matrices:
  - **rotA**: Rotation matrix to rotate around axis \(P = '0'\).
  - **rotV**: Rotation matrix to rotate \(v\) into \(w\).
  - **rotL**: Matrix \(m\) for multiplication \(m \times \times \times \text{ vector}\).
  - **getAP**: List with rotation axis and rotation angle corresponding to input matrix.

- **Other**:
  - **angle**: angle between vectors
  - **lV**: Euclidean (spatial) length of vector
  - **scprod**: scalar product
  - **vecprod**: vector product = cross product

Note

**rotZ**: see **toPol** angle: uses \(\cos\) and \(\sin\)

\(v \%v\% w\) : same as \(\text{vecprod}(v, w)\)

\(v \%s\% w\) : same as \(\text{scprod}(v, w)\)

Author(s)

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

```r
pkg <- TRUE # FALSE for direct use
(x <- toPol(1.0, 1.0)) # $x 1.41421, $y 0.785398 = pi/4
(y <- toRec(2.0, pi)) # $x -2, $y 2.44921e-16
toPol(y[1], y[2]) # 2, pi
toRec(x[1], x[2]) # 1, 1
rotZ(1, 0, pi/2) # 6.123032e-17 1.000000e+00
x <- 1; y <- 2; z <- 3
(R <- toSph(x, y, z)) # r 3.7416574, theta 0.64052231, phi 1.071487
c(R[1], 180/pi*(R[2:3])) # 3.741657 36.6992252 63.434949
(w <- toTyz(R[1], R[2], R[3])) # = x, y, z
rotZ(1, 2, pi/2) # -2, 1
par <- 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) # ! #
rround(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) # ! #
rround(m %*% t(m), 2) #!!! # composite rotation matrix and orthogonality,
should be diag(3)
}
eye <- c(0.5, 2.5, 4)
re <- rotV(eye)
#$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
# [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**

```r
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**

```r
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))
```
cwhmisc  

A collection of useful functions and constants

Description

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

Details

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

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```r
## 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( splom(~data,cex=0.2) )
## Not run
```
Functions not to be called directly by the user.

Description

Recursive internal functions to adapt.

Usage

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>function to be integrated.</td>
</tr>
<tr>
<td>term</td>
<td>function to be integrated.</td>
</tr>
<tr>
<td>a</td>
<td>starting abscissa of integral.</td>
</tr>
<tr>
<td>b</td>
<td>ending abscissa of integral.</td>
</tr>
<tr>
<td>fa, fm, fb</td>
<td>function values at a, (a+b)/2, b.</td>
</tr>
<tr>
<td>is</td>
<td>parameter to control precision.</td>
</tr>
<tr>
<td>trace</td>
<td>should intermediate steps be traced</td>
</tr>
<tr>
<td>...</td>
<td>additional parameters for function f.</td>
</tr>
</tbody>
</table>

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

```r
dc
mpf
```

Usage

```r
dc(x, d = 1, 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.
DDim

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':

```r
<<echo=TRUE>>=

a <- -2; b <- -4; c <- 7

@

The coefficients are: $a = \text{sexpr}(a)$, $b = \text{sexpr}(b)$, $c = \text{sexpr}(c)$.

##

For the linear combination $z = a + bx + cy$ we then have:

```r

z = \text{sexpr(sprintf("%.4f",a))} \text{sexpr(mpf(b,3))} x \text{sexpr(mpf(c,5))}

##

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

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

---

delayt

Waiting loop for program execution

Description

Wait for approximately sec seconds during program execution

Usage

```r
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

```r
Sys.time(); nrof <- delayt(5); Sys.time()
p <- print(nrof) # 26596 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 10000000 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

```r
d <- .Machine$double.eps
v <- c(0, d, 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

```r
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

```r
t1 <- unclass(Sys.time())
x <- 0; for (i in 1:1e6) 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**

```r
ellipseC(mid, a, b=a, ra=c(-1,361), phi=0, k=a*100 )
```

```r
e llipse1( a, b=a, ra=c(-1,361), phi=0, k=a*100 )
```

```r
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**

```r
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
```
eql <- conf.ellipse(a,b,phi,df1,df2,p[2], k) + m
plot(e17+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(e17,lty=2,col="orange")
legl <- paste(as.character(p*100),rep("percent",length(p)),sep="")
# legl <- paste(as.character(p*100),rep("%",length(p)),sep="")
col1 <- c("red","orange","green","blue")
legend(x="bottom",legl,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  

Description

`eql` checks two vectors on equality; two NA's and two NaN's are compared as equal.

Usage

```r
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

```r
eql(c(1,2,3),c(1,3)) #> TRUE FALSE FALSE
eql(c(1,2,3),c(2,1,3)) #> TRUE TRUE FALSE
eql(c(NA,NaN,2,NA,3),c(NA,NaN,1,2,3)) #> TRUE TRUE FALSE FALSE TRUE
```
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)
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 <christian@echoffmann.ch>

Examples

```r
(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 $x = e^{\ln x} \times K$ (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

$K := e^{\frac{s_{\ln}^2}{2}} \left\{ 1 - \frac{s_{\ln}^2}{\frac{1}{3n}} (s_{\ln}^2 + 2) + \frac{s_{\ln}^4}{90n^2} (3s_{\ln}^4 + 44s_{\ln}^2 + 84) \right\}$

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

References

formatFix

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

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 ".".
befor e 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

```r
# Not run:
xxbig <- c(1.2e9,3.51e23,6.72e128,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" "NaN"
#> [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" "NaN"
#> [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

```r
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 
  \( \text{round}(10^d \times \text{fractional part}) \), 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}`.

**Author(s)**

Christian W. Hoffmann <christian@echoffmann.ch>

**Examples**

```r
(pcf <- contfrac(pi)) # 3, 7, 15, 1, 292, 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 ) #
```

---

**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, l, d )
int( x )
inrange( x, y )
K(z)
Km(z)
last( x )
LE( x )
LS( )
IV( 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

a, b, c, prec, l, r Real
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
x, y Real vector
w real vector > 0
functions

z complex number

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’.
modR: same as ’mod’, but towards negative infinity.
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 given by the co-
efficients, return two solutions if a != 0, else one solution, possibly NA
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)/(1 - z), inverse transformation of K normR: 2-norm.
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.
sqrt tH: 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


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() =
mod(((-3:5), 4) # 1 2 3 0 1 2 3 1
modS((-3:5), 4) # -3 -2 -1 0 1 2 3 0 1
x <= 200; y <- x + 0.1
equalFuzzy(x, y, 0.1*c(10^(-3:0))) # FALSE TRUE TRUE TRUE
equalFuzzy(x, y, 0.1*c(10^(-3:0)),FALSE) # FALSE FALSE FALSE TRUE
safeDiv(0.3, c(0, 0:2)) # 1.552518e+231
signp(c(-1:1,NA,NaN,Inf, -Inf)) # -1 1 1 1 1 1
# instead of sign() =
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$sd)
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) # 1.810000e+02 3.8414499e-23
```

---

**ggrep**

Convenience functions for grep

Description

Grepping in (my) R directory
**Usage**

grepnot(str,x,value=TRUE)
ggrep(opt="in",str,dir="~/Users/hoffmann/R/",pkg="",split=FALSE, lines=10, out=FALSE)
countChar(str, dir="~/Users/hoffmann/R/", pkg="",split=FALSE, out=FALSE)

**Arguments**

- **str**: string to do grep for
- **x**: array of strings to check with grep.
- **value**: 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
Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```r
## 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/hoffmann/R/test/ may be a directory
# grep: /Users/hoffmann/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

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

Arguments

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

Value

A matrix of `K` columns containing the sequences.

Author(s)

Christian W. Hoffmann, <hoffmann@ws1.ch>

Source

Description

Functions for conversion of hour representations

Usage

\[
\begin{align*}
Hd(\ h,\ m,\ s) \\
Hms(\ hd) \\
Hdms(\ hms)
\end{align*}
\]

Arguments

\[
\begin{align*}
h,\ m,\ s & \quad \text{Real, representing hours, minutes, seconds} \\
hd,\ hms & \quad \text{Real, decimal hours, and concatenated h,m,s}
\end{align*}
\]
Value
Hd(hours) Hms(h.m.s) Hdmsh.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 rounding

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, int20ct, int2Hex vector of strings represented by 'n'
strRound real, represented by x

**Author(s)**

Christian W. Hoffmann <christian@echoffmann.ch>

**Examples**

```r
NdM(10^4) # 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,int20ct)[1,])
unlist(sapply(1:50,int2Hex)[1,])
strRound(pi*10^4,0)/10^4 == strRound(pi,4) # TRUE
```

**Description**

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

**Usage**

```r
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 minimum exists in x.

lerp: linearly interpolated point, t=0 -> p1, t=1 -> p2

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

References


Examples

```r
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)
```
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 positive 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

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

Author(s)

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

References


Examples

n <- 10;
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**

**calendar conversions**

**Description**

calendar 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

\texttt{yr, mo, dy} integer, year, month, day of date
\texttt{hr, JD} real, hrs, Julian date
\texttt{leap} Boolean, = is given year a leap year ?
\texttt{dk} integer, day in year

Value

\texttt{Dat2Jul: JD, year year BC is to be given as \(-\text{year-1}, \text{e.g. 4 BC = -3, 1 BC = 0} \)!!}
\texttt{Jul2Dat: date (year, month, day, hours).}
\texttt{monthsN: cumulative sum of days in months.}
\texttt{Mnames: names of months.}
\texttt{Dnames: names of weekdays.}
\texttt{mdiny: c(number of month, day in (leap) year).}
\texttt{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_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 \texttt{Dat2Jul}. The reverse is done by \texttt{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 )))
# -1, 1, 1, 12; JD=1720693 , start of last year BC

c(Jul2Dat(dd <- Dat2Jul( -1,12,31 )))
# -1, 12, 31, 12; JD=1721057 , last day BC

c(Jul2Dat(dd <- Dat2Jul( 0, 1, 1 )))
# 0, 1, 1, 12; JD=1721058 , first day AD

c(Jul2Dat(dd <- Dat2Jul( 1, 1,1 )))
# 1, 1, 1, 12; JD=1721424

c(Jul2Dat(dd <- Dat2Jul( 1582,10, 4 )))
# 1582, 10, 4, 12; 2299160, last day of Julian calendar

c(Jul2Dat(dd <- Dat2Jul( 1582,10,15 )))
# 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**

```r
libs(Lib)
```

**Arguments**

- **Lib**
  
  package name, if missing see above

**Author(s)**

??

**Examples**

```r
## 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**

```r
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 smoothing
- **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

```r
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

```r
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**

```r
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**

```r
mult.fig.p(nr.plots, mfrow, mfcol,  
marP = rep(0, 4), mpg = 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* ("Plus") to default `mar`, see below.
mgp argument for \texttt{par(mgp= . )} with a smaller default than usual.

mar argument for \texttt{par(mar= . )} with a smaller default than usual, using the \texttt{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 \texttt{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 \texttt{mtext} for main and sub.

Value

A \texttt{list} with two components that are lists themselves, a subset of \texttt{par()},

\begin{verbatim}
new.par the current \texttt{par} settings.
old.par the \texttt{par before} the call.
\end{verbatim}

Author(s)

Martin Maechler, <maechler@stat.math.ethz.ch>,
modified by Christian W. Hoffmann, <christian@echoffmann.ch>

See Also

\texttt{par, layout}.

Examples

\begin{verbatim}
## 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$old.par)
## Look at the \texttt{par} setting \*AFTER* the above:
str(do.call("par", as.list(names(rr$new.par))))
\end{verbatim}
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))
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

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

```r
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**

```r
n2c(c(10e20,-10e5,10,(10:0)/10,0.05))
# "x" "6" "1" "0" "&" "@" "#" "x" "e" "+" "-" ";" "," "," "," "," "," "," ","
# attr("legend")
# [1] ">=1.0, 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)
```
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
normalize

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).

dominate: data-frame with one column c(a,e,base) for each x, such that x = a*base^e, abs( a ) in [l, 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, l for x=0.
sigplaces: number of places necessary for printing trunc(x); c(2,3,4,3) for c(NA,Inf,-Inf,NaN).
checkNormalize: reconvernt 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 <- normalize(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 2.0000000  2.0000000  1  1  1 4.00  7.00000  2.0000000  5.0000000
# b 2.0000000  2.0000000  2  2  2 2.00  2.0000  2.0000000  2.0000000

(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
num.ident

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)
(aaaaa <- outer(xxxx, xxxx, function(x, y) num.ident(x, y)))
all((aaaaa & !is.na(aaaa)) == (row(aaaa) == col(aaaa)))
# aaaa has TRUE only on the diagonal, i.e. identity works correctly
**num2Latex**

*Convert numeric containing e+-power*

**Description**

Latex string with power notation

**Usage**

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

**Arguments**

- **x**: numerical vector
- **digits**: digits to show, see also options `scipen`

**Value**

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

**Author(s)**

<dimitris.rizopoulos@med.kuleuven.be>

**Examples**

```r
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, 8120000000, 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))
```

---

**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 a string with justification, insertion

Description
Pad a string, insert substring.

Usage
```
pad( str, space, loc = c("right", "left", "center", "none"), with=" " )
justify( str, space, loc = 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.
- `loc`: Mode of padding, of justification, one of "left","right","center", partial matching is allowed. If missing, "right" is taken, meaning for pad(loc="r") right-ways extended (i.e. flush left), for justify(loc="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 with 'ins' inserted after character number 'point' of 'str'.

Note
- `pad(loc="r")` right-ways extended (i.e. flush left),
- `justify(loc="r")` right-justified,
- `loc="none"` returns `str` unchanged.

Author(s)
Christian W. Hoffmann <christian@echoffmann.ch>

Examples
```
pad("My string",25,"c","XoX")
# [1] "XoXXXXXoMy stringXoXXoXXoXX"
pad("My string",25) # right aligned
(str <- paste0(LETTERS)) # "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
insstr(str," $ ",7) # "ABCDEF $ GHIJKLMNOPQRSTUVWXYZ"
```
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"))

check files for parsing errors

 parsecheck

Description

check files for parsing errors

Usage

parsecheck(str="/Users/hoffmannc/R/test0/R")

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"

---

```r
pastePP
```

Description

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

Usage

`pastePP(...)`

Arguments

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

Value

pasted strings using `collapse=""`.

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

```r
# 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"
paste0 (a,b,s="-",c=";") # "1 3 - ;" "2 4 - ;" "1 5 - ;"
paste00(0:9)      # "0123456789"
paste00(LETTERS)  # "ABCDEFHIJKLMNOPQRSTUVWXYZ"
```
pasteRound

**Paste rounded values**

**Description**

Paste rounded values

**Usage**

```r
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**

```r
x <- rnorm(3)
x
matrix(pasteRound("x1=" , x[1] , " , x2=" , x[2] , " , x3=" , x[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))))
SplomT(data, mainL = deparse(substitute(data)), xlabL = "",
       hist = "h", adjust = 1, hist.col = trellis.par.get("strip.background")$col[5],
       cex.diag = 1, h.diag=0.4, colYonX = "red", colXonY = "blue", ...)

Arguments

m        Numerical matrix

tit      Overall title for plot. A vector of one or two elements. If an element is an
         expression, plotmath will be used.

rows     Number of rows of panels

cols     Number of columns of panels

f        A function to plot the individual plot panels. It can also be a statement sequence
         {...}.

cex      Font size used for tit

reset    Should previous rows, cols be restored after execution. See note

outer    Passed on to mtext.

oma      Outer margin used in initial par(...).

mar      Lines of margin used in initial par(...).

data     Matrix or dataframe containing data, variables in columns

breaks   Breaks for histogram

mainL    Label on top of scatterplot matrix or matrix of histograms

mainC    Labels on top of each of the histograms, should be character vector of length =
         number of columns of data

xlabL    Label for x axis

hist     "h" = histogram, "d" = density curve, "b" = both

adjust   factor to adjust smoothing window for density curve

hist.col  colour for the bars of the histograms

cex.diag  correction factor for font height of correlations and names in the diagonal

h.diag    placement of the variable name in the diagonal panel, =0 means on the lower
         border, = 0.5 in the middle between lower and upper border

colYonX, colXonY  colour of smoothing lines, y on x and x on y

...  Parameters passed on to upper.panel,lower.panel,diag.panel
Value

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

WARNING

The sequence of functions contained in f MUST NOT contain any call to \texttt{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 \texttt{pltRCT}. Uses \texttt{splom}, \texttt{[lattice:extend.limits]}\texttt{extend.limits}, and \texttt{datetime}.

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

Author(s)

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

Examples

\begin{verbatim}
x <- rnorm(100); y <- rnorm(100)+1; z <- y+rlnorm(100) pltRCT(1,1,f={plot(x,y,xlab="data with trend");
  abline(reg=lm(y~x),lty=2);points(x,z,pch=3)}) nr <- 100; nc <- 8;
data <- as.data.frame(matrix(rnorm(nr*nc),nrow=nr,ncol=nc))
data[,nc] <- data[,nc-2] + 0.3*data[,nc-1] #generate higher correlations
data[,nc-1] <- data[,nc-1] + 0.9*data[,nc]
colnames(data)<-paste("vw",letters[1:nc],sep="")
SplomT(data,main="",hist="d",cex.diag=0.6,hist.col="green")
SplomT(data,main="",hist="b",adjust=0.4,cex.diag = 0.5)
pltRCT(1, 1, tit="1 by 1 plot", f=plot(y,x-3*y) )
nr <- 25; nc <- 16
pltRCT(1, 2, f=(plot(x,y,xlab="my x")
  m <- matrix(rnorm(nr*nc),nrow=25,ncol=nc)
  pltCharMat(m,cex=0.5,col="red")
})
\end{verbatim}
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 $xi : xi = f * Q * x + tr + \text{error}$. 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


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,
3.3907,1.9054),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))
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

progressMeter(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 1:10^3) {
    kk <- kk+1  # do something time consuming
  }
  progressMeter(i)
}
cat(""
# 0.000+...+ 1.000+...+ 2.000+...+ 3.000+...+ 4.000+...+
# 50.000+...+ 6.000+...+ 7.000+...+ 8.000+...+ 9.000+...

qnorm.appr

Approximation to the inverse normal distribution function.

Description

qnorm.appr 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.app16(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
**qnorm.appr**

**Arguments**

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

**Value**

`qnorm.appr` gives the quantile function for the different approximations.

**Warning**

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

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

**Note**

`qnorm.appr` 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**


**Examples**

```r
prec <- function(x,y,z=y) max(abs((x-y)/z)) # relative precision
x2 <- -0.6744897501960017; p2 <- 0.25
x0 <- -3.090232306167814; p0 <- 0.001
xm <- -9.262340099798408; pm <- 1.0e-20
x <- c((100:0)/10,x2,x0,xm)
p <- qnorm(x)
x3 <- qnorm.appr3(p)
x4 <- qnorm.appr4(p)
x1 <- qnorm.appr16(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 ?
```
Conversion of real to string and rounding, in given base

Description

Functions for conversion of real to string in given base, and back, 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 of vectors of strings representation 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 `EXPCCHAR` 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 signifiers "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

```r
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 

c(x, 10, 7)$s
strB2r("- 9167.8", 10)
strB2r("800z3", 15 ) # 8353125

(ii <- r2B( 8353125, 32, 4 )$s) # "7UT5.0000"

strB2r( ii, 32 ) # 8353125

roundB(c(0.4,0.3),2,16) # 0.399999390.30000305
```
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, codelinkdirname of package(s).
pkg character, codelinkbasename 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 0:6, 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.
Permissions for the vignette *.Rnw should be changed by system("chmod u=rwx ...Rnw "), if necessary.

Author(s)

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

## Not run:
RCA("mypackage", 2:4)

## End(Not run)

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.

Author(s)

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

Examples

```r
 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

```r
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

```r
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

```r
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, 0.05, 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, if by is in the "wrong" direction, see examples.

Usage

seqm(from, to, by=1)

Arguments

from       starting value of sequence.
to         (maximal) end value of the sequence.
by         increment of the sequence.

Value

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

Author(s)

Christian W. Hoffmann <christian@echoffmann.ch>

Examples

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

sets

Description
Check whether one set is included within another.

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

Arguments
- `x, X` sets.

Value
TRUE, if `x` is included in `X`.

Author(s)
Christian W. Hoffmann <christian@echoffmann.ch>

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

shapiro.wilk.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.
smoothed.df

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

```r
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")
```

---

**Description**

Function for rounding real given as string representation

**Usage**

```r
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 representation of `x`, round to `round` decimal digits

**Author(s)**

Christian W. Hoffmann <christian@echoffmann.ch>

**Examples**

```r
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

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


Examples

par(mfrow=c(2,2))
T3plot(rnorm(100))
T3plot(rnorm(10000))
T3plot(rnorm(1000)+runif(1000)*0.1,"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 label
tpos position of captions: "a" for above table, "b" for below table
stretch optional vector with two entries, giving the baselinestretch for the caption (stretch[1]) and the columns of the table (stretch[2]); default: no adjustment of baselinestretch
adjust adjusts the columns of the \LaTeX\ table, default: "r" (right), also possible: "l" (left) and "c" (centre) or user defined: "adjust=c("l","c","r",...)
   " yields \{l|c|r...\}
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

```r
m <- matrix(rnorm(100), nrow=10, ncol=10, dimnames=list(LETTERS[1:10], colnames=letters[1:10]))
text.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\n# B & 0.18 & 0.39 & 0.78 & -0.10 & -0.25 & -0.61 & -0.04 & -0.71 & -0.14 & 1.21\n# ...\n```

---

**triplot**

*Ternary or Triangular Plots.*

Description

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

Usage

```r
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>
w.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)) # 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)) # 7
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**

```r
cat(1)
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 *sunk* (where it would hang the process).

**Value**

None.

**Author(s)**

Christian W. Hoffmann <christian@echoffmann.ch>

**Examples**

```r
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)
whole

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

*Topic **NA**
  jitterNA, 41
*Topic **algebra**
  div.prot, 20
  pointfit, 61
*Topic **arith**
  astroC, 5
  astroGeo, 6
  Const, 10
  dcm, 16
  eql, 23
  frac, 28
  functions, 29
  int2, 36
  num.ident, 52
  seqm, 74
  sets, 75
  str2B, 77
*Topic **array**
  select.range, 73
*Topic **character,arith**
  arcs, 4
  clocksense, 9
  coords, 10
  digits, 19
  hours, 35
  numer, 54
*Topic **character**
  cpos, 13
  padding, 55
  paste00, 57
  r2B, 68
  replacechar, 72
*Topic **chrono**
  datatime, 15
  delayt, 18
  dt2str, 21
*Topic **color**
  plotSymbols, 58

*Topic **data**
  jitterNA, 41
  whole, 82
*Topic **device**
  lpr, 44
  panel, 56
*Topic **distribution**
  f.log, 24
  Halton, 34
  invgauss, 39
  my.table, 47
  numberof, 53
  qnorm.appr, 66
  smoothed.df, 76
  T3plot, 78
*Topic **documentation**
  ls.functions, 45
  pasteRound, 58
*Topic **dplot**
  ellipse, 22
*Topic **hplot**
  lowess.bygroup, 43
  mult.fig.p, 45
  panel, 56
  plotSymbols, 58
  plt, 59
  triplot, 80
*Topic **htest**
  shapiro.wilk.test, 75
*Topic **interface**
  tex.table, 79
*Topic **iplot**
  plotSymbols, 58
*Topic **language**
  parsecheck, 56
*Topic **logic**
  is.constant, 40
  num.ident, 52
  remove.dup.rows, 71
INDEX

* Topic **manip**
  clean.na, 8
  library, 43
  remove.duplicate.rows, 71

* Topic **math**
  adaptlob, 3
  factor, 25
  interp, 37
  Julian date, 41
  normalize, 50

* Topic **misc**
  cpos, 13
  parsecheck, 56
  paste0, 57
  pasteRound, 58

* Topic **models**
  FinneyCorr, 26

* Topic **multivariate**
  ellipse, 22

* Topic **package**
  cwhmisc, 14

* Topic **print**
  cap, 7
  datetime, 15
  delstr, 19
  formatFix, 27
  n22dig, 48
  n2c, 49
  NA2str, 50
  num2Latex, 53
  printP, 63
  progress.meter, 65

* Topic **programming**
  waitReturn, 82

* Topic **regression**
  FinneyCorr, 26
  scode, 72

* Topic **robust**
  w.median, 81

* Topic **structures**
  Ddim, 17

* Topic **symbolmath**
  interp, 37

* Topic **utilities**
  ggrep, 32
  RCA, 70
  .adaptlobstp(cwhmisc), 15
  .adaptsimstp(cwhmisc), 15

% (coords), 10
%v (coords), 10
acos, 11
adaptlob, 3
adaptsim(adaptlob), 3
all.equal, 40
allDigits(digits), 19
allFactors(factor), 25
angle (coords), 10
arcs, 4
ASCII (Const), 10
asin, 11
astroC, 5
astroGeo, 6
availColors (plotSymbols), 58
c38 (Const), 10
c30, 31
c3Q (Const), 10
cAE (astroC), 5
cap, 7
capitalize(cap), 7
capLeading(cap), 7
capply(cap), 7
catE(printP), 63
catn(printP), 63
Cayley(functions), 29
cC(astroC), 5
cDAYPJULCENT (astroC), 5
cDAYPMONSID (astroC), 5
cDAYPMONSYN (astroC), 5
cDAYPYEARSID (astroC), 5
cDAYPYEARTROP (astroC), 5
cEPSOBL (astroC), 5
charMat, 59
ccharMat(n2c), 49
ccheckNormalize(normalize), 50
chsvd(functions), 29
cJDI2000 (astroC), 5
cK(astroC), 5
clean.na, 8
clockSense, 9
ClockSense2(clocksense), 9
ClockSense3(clocksense), 9
clockwise(clocksense), 9
cMY(astroC), 5
compare.ellipse(ellipse), 22
Const, 10
INDEX

contrfrac (frac), 28
coords, 10
countChar (ggrep), 32
counterClock (clocksense), 9
cpos, 13
cposR (cpos), 13
cposV (cpos), 13
cPRECESS (astroC), 5
cRE (astroC), 5
cSBYE (astroC), 5
cSBYEMY (astroC), 5
cSBYJU (astroC), 5
cSBYMA (astroC), 5
cSBYME (astroC), 5
cSBYNE (astroC), 5
cSBYPL (astroC), 5
cSBYSA (astroC), 5
cSBYUR (astroC), 5
cSIDBYSOL (astroC), 5
cSOLBYSID (astroC), 5
cwhmisc, 14
cwhmisc-internal (cwhmisc), 15
cwhmisc-package (cwhmisc), 14
cwhmisc, 15

dat2Jul (Julian date), 41
date, 31
datetime, 15, 61
dAYINMOnTH (astroC), 5
dc, 29
dc (dcm), 16
dcm, 16
dcn (dcm), 16
ddim, 17
deg (arcs), 4
delayt, 18
delstr, 19
dev.copy, 44
digits, 19
dinvgauss (invgauss), 39
div.prot, 20
divmod (functions), 29
divmodL (functions), 29
dnames (Julian date), 41
DPY (astroC), 5
drop, 8
dsm (functions), 29
dt2str, 21
ellipse, 22
ellipse (ellipse), 22
ecl (ellipse), 22
equal (functions), 29
equalFuzzy (functions), 29
Eratosthenes (factor), 25
EulerPhi (num), 54

exp (functions), 29
EXPClAR, 69
EXPClAR (Const), 10
explainLegend (n2c), 49
expression, 60

f. log, 24
factor, 25
factorN (factor), 25
FALSE, 52, 83
fC.lm (FinneyCOrr), 26
FinneyCOrr, 26
for, 74
format, 27
formatFix, 27, 64
frac, 28
frac (functions), 29
functions, 29
gcd (num), 54
gAp (coords), 10
ggrep, 32
GreatestIntegralAsRealF (Const), 10
grep, 33
grepnot (ggrep), 32

Halton, 34
Hd (hours), 35
Hdms (hours), 35
HexagesDig, 69
HexagesDig (Const), 10
HexDig (Const), 10
histRCT (plt), 59
Hms (hours), 35
Hmsd (hours), 35
hours, 35
HS247 (Halton), 34

identical, 40
INDEX

indexLine (n2c), 49
inrange (functions), 29
insstr (padding), 55
int (functions), 29
int2, 36
int2ASCII (int2), 36
int2B (int2), 36
int2Hex (int2), 36
int2Oct (int2), 36
interpol, 37
Inv (numeric), 54
invgauss, 39
is.constant, 40
is.prime (factor), 25
isCounterCl2 (clocksense), 9
isCounterCl3 (clocksense), 9
isLeap (Julian date), 41
isNumeric (digits), 19
issubstr (cpos), 13
jitter, 41
jitterNA, 41
Jul2Dat, 31
Jul2Dat (Julian date), 41
Julian date, 41
justify, 68
justify (padding), 55
K (functions), 29
Km (functions), 29
last (functions), 29
LatBerne (astroGeo), 6
layout, 46
LB2MK (astroGeo), 6
LB2YX (astroGeo), 6
LE (functions), 29
legend, 78
lerp (interpol), 37
libs, 43
list, 46
LongBerne (astroGeo), 6
lower (cap), 7
lowerize (cap), 7
lowess.bygroup, 43
lpr, 44
LS (functions), 29
ls (functions), 45
ls.notfunctions (ls.functions), 45
1V (functions), 29
mdiny (Julian date), 41
median, 81
minInterp (interpol), 37
Mnames (Julian date), 41
mod (functions), 29
modexp (numeric), 54
modR (functions), 29
modS (functions), 29
monthsN (Julian date), 41
mpf (dcm), 16
mtext, 46
mult.fig.p, 45
my.table, 47
mydate (datetime), 15
mytime (datetime), 15
n22dig, 48
n2c, 48, 49
n2cCompact (n2c), 49
n2mfrow, 61
NA2str, 50
Nd (normalize), 50
NdM (int2), 36
NoneClock (clocksense), 9
norm2 (functions), 29
normalize, 50
normalize! (normalize), 50
NprintE (printP), 63
NprintM (printP), 63
NprintP (printP), 63
NprintT (printP), 63
NprintV (printP), 63
num.ident, 52
num2Latex, 53
numberof, 53
numeral, 54
one (functions), 29
onebyx (functions), 29
options, 53
pad (padding), 55
padding, 55
panel, 56
par, 45, 46
parsecheck, 56
paste, 58
paste00, 57
pasteRound, 58
pinvgauss (invgauss), 39
plotmath, 60
plotSymbols, 58
plotSymbolsFonts (plotSymbols), 58
plt, 59
pltCharMat, 49
pltCharMat (plt), 59
pltRct (plt), 59
pointfit, 61
postscript, 61
powr (functions), 29
PRIMES (factor), 25
primes (factor), 25
printE (printP), 63
printM (printP), 63
printP (printP), 63
printT (printP), 63
print, 64
printP, 63
printV (printP), 63
prodN (factor), 25
progress.meter, 65
pythag (functions), 29

qf, 22
qinvgauss (invgauss), 39
qnorm, 67
qnorm.ap16 (qnorm.appr), 66
qnorm.app3 (qnorm.appr), 66
qnorm.app4 (qnorm.appr), 66
qnorm.appr, 66
quadmin (interpol), 37
quantile, 81
quotmean (functions), 29

R2.1m (FinneyCorr), 26
r2B, 20, 68
r2Be, 10
r2Be (r2B), 68
rad (arcs), 4
RCA, 70
reda (arcs), 4
reda2 (arcs), 4
remove.dup.rows, 71
replacechar, 72
rinvgauss (invgauss), 39
rotA (coords), 10

rotL, 62
rotL (coords), 10
rotV (coords), 10
rotZ (coords), 10
round, 58
roundB (r2B), 68

s.lm (FinneyCorr), 26
safeDiv (functions), 29
scm (numeric), 54
scode, 72
scprod (coords), 10
select.range, 73
seq, 74
seqm, 74
setincl (sets), 75
sets, 75
setupInterp (interp), 37
shapiro.test, 76
shapiro.wilk.test, 75
signp (functions), 29
sigplaces (normalize), 50
smoothed.df, 76
solveEq (functions), 29
source, 59
splom, 61
SplomT (plt), 59
sprintf, 17
sqr (functions), 29
sqrth (functions), 29
str2B, 77
str2dig (digits), 19
strB2i (r2B), 68
strB2r (r2B), 68
strReverse (cap), 7
strRound (int2), 36
submod (functions), 29
summaryFs (FinneyCorr), 26
svd, 30, 31
Sweave, 16

T3plot, 78
table, 47
tau (const), 10
tex.tab0 (tex.table), 79
tex.table, 79
toCfrac (frac), 28
toCfrac2 (frac), 28
toPol, 11
INDEX

toPol (coords), 10
toRec (coords), 10
toSph (coords), 10
toXyz (coords), 10
triplot, 80
TRUE, 52, 83

vecprod (coords), 10

w.median, 81
waitReturn, 82
Wday (Julian date), 41
whole, 82

xToNorthBerne (astroGeo), 6

Yday (Julian date), 41
yToEastBerne (astroGeo), 6
YX2LB (astroGeo), 6
YX2MK (astroGeo), 6

zero (functions), 29