Package ‘funchir’

July 22, 2025

Version 0.3.0-1

Title Convenience Functions by Michael Chirico

Depends R (>= 3.2.2)

Description YACFP (Yet Another Convenience Function Package). get_age() is a fast & accurate tool for measuring fractional years between two dates. stale_package_check() tries to identify any library() calls to unused packages.

Imports data.table

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

License MIT + file LICENSE

URL https://github.com/MichaelChirico/funchir

Encoding UTF-8

NeedsCompilation no

Author Michael Chirico [aut, cre]

Maintainer Michael Chirico <MichaelChirico4@gmail.com>

Repository CRAN

Date/Publication 2025-01-08 10:00:01 UTC

Contents

funchir-infix *Convenient Infix Operators*

Description

An infix operator as convenient shorthand for set modulation (A\B)

Usage

```
A %\% B
```

Arguments

A, B	Objects which can be treated as sets.

Value

This is just a wrapper for setdiff

Examples

```
set1 <- 1:5
set2 <- 4:6

set1 %\% set2 # c(1,2,3)
```

funchir-plot *Convenience Functions for Plotting*

Description

tile.axes is used in for loops to generate axes in a multi-panel plot with shared x & y axes (within row and column).

xdev2in is the inverse of graphics::xinch; namely, it converts from plotting device units into inches.

Usage

```
tile.axes(n, M, N, params = list(x = list(), y = list()),
          use.x = TRUE, use.y = TRUE)
xdev2in(x = 1)
ydev2in(y = 1)
xydev2in(xy = 1)
```

Arguments

`n`	Integer. Cell in `mfrow` to which to apply the axes; fills by *row*, following base functionality.
`M`	Integer. Number of rows specified in `mfrow`.
`N`	Integer. Number of columns specified in `mfrow`.
`params`	A length-2 `list`. `params$x` is a `list` of parameters to be passed to the x-axis. `params$y` is a `list` of parameters to be passed to the y-axis.
`use.x`	`logical`. Should the x-axis be printed?
`use.y`	`logical`. Should the y-axis be printed?
`x`	`numeric` value to convert into inches (along the horizontal axis).
`y`	`numeric` value to convert into inches (along the vertical axis).
`xy`	`numeric` value to convert into inches (along both axes simultaneously).

Details

`tile.axes` provides a simple way to incorporate the plotting of axes into a loop which creates the plots in a matrix of plots (e.g., by using `par(mfrow=c(2, 2))`) *when the axes are shared by all plots*. x axes are only printed on the bottom row of plots, and y axes are only printed on the first column of plots–this saves potentially wasted / white space by eliminating redundant axes, yet can still be done in a loop.

Some graphics functions specify some arguments with units in inches (namely, `graphics::arrows`' `length` argument). `graphics::xinch` provides the inverse functionality enabling conversion from inches into plotting units; up to numerical accuracy, then, `graphics::xinch(xdev2in(x))` == `x`.

See Also

`xinch`

Examples

```
smpl <- rnorm(100)

par(mfrow = c(2, 1), mar = c(0, 0, 0, 0), oma=c(5, 4, 4, 2) + .1)
for (ii in 1:2){
  hist(smpl[sample(length(smpl), 100, rep = TRUE)], xaxt = "n", yaxt = "n")
  tile.axes(ii, 2, 1)
}
```

funchir-utils *Miscellaneous utile functions*

Description

Several odds-and-ends functions for data manipulation & representation, etc. See details and examples.

Usage

```
stale_package_check(con)
embed.mat(mat, M = nrow(mat), N = ncol(mat), m = 1L, n = 1L, fill = 0L)
quick_year(dates)
quick_mday(dates)
quick_yday(dates)
```

Arguments

`con`	A file/connection where output should be written.
`mat`	A matrix.
`M`	An integer specifying the number of rows in the enclosing matrix.
`N`	An integer specifying the number of columns in the enclosing matrix.
`m`	An integer specifying the row at which to insert `mat`.
`n`	An integer specifying the column at which to insert `mat`.
`fill`	An atomic vector specifying how to fill the enclosing matrix.
`dates`	A vector of `Date`s.

Value

`stale_package_check` (DEPRECATED in favor of lintr::unused_import_linter) reads a file (with `readLines`) and checks which functions are actually used from each loaded package. Currently only checks for `library` (i.e., not `require`) calls.

`embed.mat` inserts a supplied matrix into a (weakly) larger enclosing matrix, typically filled with 0s, at a specified position.

`quick_year` converts a `Date` object into its year efficiently; also ignores concerns of leap centuries. `quick_mday` returns the day of the month. `quick_yday` returns the day of the year. Returns as an `integer`.

Examples

```
inmat <- matrix(1:9, ncol = 3L)
embed.mat(inmat, M = 4L, N = 4L)
embed.mat(inmat, N = 6L, n = 4L, fill = NA)

d1 = as.Date('1987-05-02')
```

```
d2 = as.Date('2016-02-23')
quick_year(d1)
quick_mday(d1)
```

`get_age` *Calculate an exact age in fractional years*

Description

For someone born May 1, 1990, what is their age on May 2, 2000? 10 years, but what if we want more precision? They are 1 day older, and May 1, 2001 is in 364 days, so they are 10 + 1/365 years old.

Things get more complicated when we include consideration of leap years, when the next birthday might be 366 days away.

`get_age()` solves this problem.

Note that it assumes there are no leap centuries (and hence may will be incorrect for dates before March 1, 1900 or after February 28, 2100). It also takes the stance that leap babies (those born February 29) increment their age on March 1 in non-leap years.

Usage

```
get_age(birthdays, ref_dates)
```

Arguments

`birthdays`	A vector of `Date`s (or input coercible with `as.Date()`). Each entry is someone's birthday.
`ref_dates`	A vector of `Date`s (or input coercible with `as.Date()`). Each entry is a "current date" at which to calculate the corresponding age.

Value

Numeric vector of years (including fractional parts) between each `ref_dates` and `birthdays` entry.

Index