# Package 'funchir'

October 13, 2022

**Version** 0.2.2

**Title** Convenience Functions by Michael Chirico

**Author** Michael Chirico

**Maintainer** Michael Chirico <MichaelChirico4@gmail.com>

**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. abbr_to_colClass() is a much more concise way of feeding many types to a colClass argument in a data reader. stale_package_check() tries to identify any library() calls to unused packages.

**Imports** data.table

**Suggests** testthat, jsonlite

**License** GPL (>= 2)

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

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2022-04-17 17:22:28 UTC

## R topics documented:

## funchir-infix *Convenient Infix Operators*

### Description

Several infix operators which are convenient shorthand for common set operations, namely, *modulation* (A\B), *union* (AUB) and *intersection* (A & B).

### Usage

```
A %\% B
A %u% B
A %^% B
```

### Arguments

| | |
|---|---|
| A | A set A. |
| B | *idem* A. |

### Value

The above are simply wrappers for the base functions `setdiff`, `union`, and `intersect`, respectively, so output is exactly as for those functions.

### See Also

`setdiff` , `union`, `intersect`

### Examples

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

set1 %\% set2 # c(1,2,3)
set1 %u% set2 # c(1,2,3,4,5,6)
set1 %^% set2 # c(4,5)
```

## funchir-io *Convenient Functions for Data Reading*

### Description

Functions which come in particular handy for process of reading in data which can turn verbose code into readable, clean code.

### Usage

```
abbr_to_colClass(inits, counts)
```

### Arguments

| | |
|---|---|
| inits | Initials of data types to be passed to a colClasses argument (most typically in fread from data.table for me). See details. |
| counts | Corresponding counts (as an *unbroken string*) of each type given in inits. See details |

### Details

abbr_to_colClass was designed specifically for reading in large (read: wide, i.e., with many fields) data files when it is also necessary to specify the types to expect to the reader for speed or for accuracy.

Currently recognized types are blank, character, factor, logical, integer, numeric, Date, date, text and skip, which are abbreviated to their first initials: "b", "c", "f", "l", "i", "n", "D", "d", "t" and "s", respectively.

Since like types are often found in sequence, the counts argument can condense the call considerably– if three integer columns appear in a row, for example, we could specify inits="i" and counts="3" instead of the breathier inits="iii", counts="111".

Note that since counts is read digit-by-digit, sequences of length greater than 9 must be broken up into size-9 (or smaller) chunks, e.g., if there are 20 Date fields in a row, we could set inits="ddd", counts="992". This approach was taken (rather than, say, requiring counts to be an integer vector of counts) as I find it speedier and more concise, and the direct parallel to inits can elucidate issues which arise directly in the code instead of, say, checking cbind(strsplit(inits, split = "")[[1L]], counts).

### Examples

```
abbr_to_colClass(inits = "ncifdfd", counts = "1234567")
```

---

## 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-table *Convenient Wrappers for creating and printing tables*

### Description

Here are wrappers for common table creation/manipulation/printing operations.

### Usage

```
sanitize2(str)
```

### Arguments

| | |
|---|---|
| str | character vector. |

### Details

sanitize2 is a replacement to the internal sanitize function used by default in xtable. Adds items for fixing left and right square brackets, which are (in the current–2017/03/03–version of print.xtable ) by default left alone, which can cause errors.

### Examples

```
sanitize2('$\\mathcal{B}$')
```

## funchir-utils *Miscellaneous utile functions*

### Description

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

### Usage

```
create_quantiles(x, num, right = FALSE, na.rm = FALSE,
                 include.lowest = TRUE, labels = 1:num)
to.pct(x, dig = Inf)
nx.mlt(x, n)
divide(x, n, na.rm = FALSE)
dol.form(x, dig = 0L, suff = "", tex = FALSE)
ntostr(n, dig = 2L)
write.packages(con)
stale_package_check(con)
embed.mat(mat, M = nrow(mat), N = ncol(mat), m = 1L, n = 1L, fill = 0L)
get_age(birthdays, ref_dates)
```

```
quick_year(dates)
quick_mday(dates)
quick_yday(dates)
```

## Arguments

| | |
|---|---|
| x | A numeric vector. |
| num | A number, typically an integer, specifying how many equal-count intervals into which to divide the data. |
| right | logical, indicating if the intervals should be closed on the right (and open on the left) or vice versa. |
| na.rm | logical passed to quantile with the usual interpretation. |
| include.lowest | logical, indicating if an x[i] equal to the lowest (or highest, for right = FALSE) breaks value should be included. |
| labels | character vector of length num; the labels to be applied to the resulting factor. |
| dig | The number of digits to be included past the decimal in output; sent directly to round. |
| suff | The suffix to appended/unit in which to express x. Currently one of c("", "k", "m", "b"), corresponding to plain units, thousands, millions, and billions, respectively. |
| tex | Should $ be printed as \$ for direct copy-pasting to TeX files? |
| n | For nx.mlt, divide and ntostr, a number; see details. For embed.mat, an integer specifying the column at which to insert mat. |
| 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. |
| fill | An atomic vector specifying how to fill the enclosing matrix. |
| birthdays | A vector of Dates. |
| ref_dates | A vector of Dates. |
| dates | A vector of Dates. |

## Value

create_quantiles is a parsimonious function for generating quantiles of a vector (e.g., quartiles for num=4 or quintiles for num=5). Basically a wrapper for the cut function; the type of the output is factor. Fails for vectors with overlapping quantiles (e.g., with >50% of values of x equal to zero) unless the correct number of labels (i.e., the number of unique quantile breaks) is given in the labels argument.

to.pct converts a number (probably a proportion, i.e., typically between 0 and 1) to a percentage; also has an argument (dig) which can be used to round the output inline.

`nx.mlt` returns the least multiple of `n` which (weakly) exceeds `x`. Convenient for making axes ticks land on pretty numbers.

`divide` divides the range (min through max) of `x` into `n` points (basically a shorthand for `seq`).

`dol.form` takes a financial input and converts it to a (American-formatted, American-currency) string for printing–appending a dollar sign (`"\$"`) and inserting commas after every third digit from the left of the decimal point.

`ntostr` converts `n` to a `character` vector with each element width `dig`. This is particularly nice for converting 99:100 to "99" and "100".

`write.packages` captures the current package environment (inspired by `sessionInfo()` and writes it as a JSON to `con` with `writeLines`; a `list` version of this object is returned. This may be essential for tracking across time which package versions were being used.

`stale_package_check` 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.

`get_age` returns the accurate, fractional age (in years) of each individual, quickly. Accuracy deteriorates when non-leap century years are involved (i.e., any year congruent to 0 mod 100 but not 0 mod 400); designed for use with currently-relevant birthdays and ages.

`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`.

## See Also

`cut`, `prettyNum`

## Examples

```
x <- runif(100)

# Return which multiple of 1/7 least
#   exceeds each element of x
create_quantiles(x, 7)

to.pct(x)
to.pct(x, dig = 2) #output of the form xxx.xx

nx.mlt(x, 1/3)

dol.form(x, dig=2L)

ntostr(999:1000, dig = 3L) # c("999","000")
ntostr(999:1000, dig = 2L) # c("99","00")

library(stats)
write.packages()

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')
get_age(d1, d2)
quick_year(d1)
quick_mday(d1)
```

# Index