Package 'funchir'

March 6, 2017

Version 0.1.4

Title Convenience Functions by Michael Chirico

Author Michael Chirico

Maintainer Michael Chirico <MichaelChirico4@gmail.com>

Depends R (>= 3.2.2)

Description A set of functions, some subset of which I use in every .R file I write. Examples are table2(), which adds useful functionalities to base table (sorting, built-in proportion argument, etc.); lyx.xtable(), which converts xtable() output to a format more easily copy-pasted into LyX; pdf2(), which writes a plot to file while also displaying it in the RStudio plot window; and abbr_to_colClass(), which is a much more concise way of feeding many types to a colClass argument in a data reader.

Imports data.table

License GPL (>= 2)

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

NeedsCompilation no

Repository CRAN

Date/Publication 2017-03-06 11:13:06

R topics documented:

funchir-infix *Convenient Infix Operators*

Description

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

Usage

```
s1 %+% s2
A %\% B
A %u% B
A %^% B
```

Arguments

s1	A string, or anything that can be understood in the first argument of paste0.
s2	*idem* s1.
A	A set A.
B	*idem* A.

Value

The above are simply wrappers for the base functions paste0, setdiff, union, and intersect, respectively, so output is exactly as for those functions.

See Also

paste0, setdiff , union, intersect

Examples

```
directory <- "~/some/directory/"
fil <- "some_file.ext"
# very convenient for repeated calls to a directory outside getwd()
directory %+% fil

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-plot *Simulcast Plot Wrappers*

Description

Several wrapper functions which make it convenient / parsimonious to simultaneously write a plot to a file (currently limited to .png and .pdf types) *and* to print the plot to the Rstudio graphics window.

Usage

```
pdf2(...)
png2(...)
dev.off2(typ = "pdf")
tile.axes(n, M, N, params = list(x = list(), y = list()),
          use.x = TRUE, use.y = TRUE)
rel_coord(ax, lambda = 0)
```

Arguments

`...`	For `pdf2`, `png2`: list of arguments to be passed directly to `pdf` or `png`.
`typ`	Either `"pdf"` or `"png"`, must match the plot type currently open.
`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`.
`ax`	Either `"x"` or `"y"`; axis for which we're specifying coordinates.
`lambda`	Either a mixing weight, giving the percentage of the way between the lower and higher endpoint of the specified axis (0 : smaller endpoint, 1 : larger endpoint), or one of the following relative position characters: `c("top", "bottom", "left", "right")`.
`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?

Details

Instead of the standard `pdf("file.pdf") [plot code] dev.off()` approach used to write a plot to file, simply replacing the former and latter with these convenience functions will write the same file, but also print the output in the RStudio plotting window.

`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.

`rel_coord` gives a simple way to place text, labels, etc. in a plot by specifying only relative coordinates. The first form mimics the flexibility of `legend`, which accepts such relatively-defined positions as `"top"` and `"left"`. For example, specifying `lambda = "right"` will return an x coordinate close to the right endpoint of the currently defined axes. Alternatively, an exact proportion can be supplied – if the desired coordinate is 75% of the way to the right endpoint, for example, specify `rel_coord("x", .75)`.

See Also

`pdf`, `png` , `dev.off`

Examples

```
smpl <- rnorm(100)

## Not run:
  #Runs in RStudio
  pdf2(file.path(temp.dir(), 'test.pdf'))
  hist(smpl)
  dev.off2()

  png2(file.path(temp.dir(), 'test.png'))
  hist(smpl)
  dev.off2(typ="png")

## End(Not run)

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-read` *Convenient Functions for Data Reading*

Description

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

Usage

```
read.xlsx3(...)
abbr_to_colClass(inits, counts)
```

Arguments

...	Arguments to be passed to read.xlsx2.
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

read.xlsx3 is a simple wrapper for read.xlsx2 (from the package xlsx) which subsequently calls gc (invisibly) due to an apparent memory leak problem in read.xlsx2.

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 character, factor, integer, numeric, Date, date, and text, which are abbreviated to their first initials: "c", "f", "i", "n", "D", "d", and "t", 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

```
## Not run:
    fl <- "~/path/to/Excel/file.xlsx"
    read.xlsx3(fl)

## End(Not run)

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

---

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

Description

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

Usage

```
table2(..., dig = if (prop) 2L else NULL, prop = FALSE, ord = FALSE, pct = FALSE)
lyx.xtable(...)
sanitize2(str)
```

Arguments

`...`	For both functions, the code to be passed to the wrapped function. For `table2`, can also include arguments to `prop.table`.
`dig`	Number of digits to be printed in the output. Defaults to 2 for proportion tables because output tends to be ugly if unrestricted.
`prop`	Print a proportional table? This will activate a call to `prop.table`.
`ord`	Should the table be ordered? Three inputs are understood: 1) `FALSE`; returns table in standard order, 2) `TRUE`; returns table in *increasing* order, 3) `"dec"`; returns table in *decreasing* order.
`pct`	Should table values be converted to a percentage (i.e., multiplied by 100)? Typically used with `prop = TRUE`; note that `round` is applied *after* percentage conversion, so keep this in mind if using `dig` simultaneously.
`str`	`character` vector.

Details

`table2` is especially useful for two common usages of `table`–producing ordered frequency tables of a variable (to find most/least common realizations) and producing proportional two-way tables of two variables.

`lyx.xtable` (based on package `xtable`) is designed specifically for users of the free WYSIWYM LaTeX editor LyX, which converts copy-pasted LaTeX to indecipherable gibberish unless lines of code are separated by two newlines. Note that `lyx.xtable` wraps `xtable::print.xtable`.

`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.

See Also

`table`, `prop.table`

Examples

```
x <- sample(10, size = 100, replace = TRUE)
y <- sample(3, size = 100, replace = TRUE)
tbl <- table2(x, y, prop = TRUE, margin = 1)
table2(x, ord = "dec")
table2(y, ord = TRUE)
table2(y, dig = -1L)

## Not run:
  lyx.xtable(xtable(tbl), sanitize.text.function = sanitize2)

## End(Not run)
```

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, include.lowest = TRUE, labels = 1:num)
to.pct(x, dig = Inf)
nx.mlt(x, n)
dol.form(x, dig = 0L, suff = "", tex = FALSE)
ntostr(n, dig = 2L)
write.packages(file)
embed.mat(mat, M = nrow(mat), N = ncol(mat), m = 1L, n = 1L, fill = 0L)
get_age(birthdays, ref_dates)
quick_year(dates)
quick_wday(dates)
quick_mday(dates)
D(...)
```

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.
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 and ntostr, a number; see details. For embed.mat, an integer specifying the column at which to insert mat.
file	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.
...	Items and possibly arguments to (methods of) as.Date.

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.

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.

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 date & time and writes the output to the specified file. This may be essential for tracking across time which package versions were being used.

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_wday returns the day of the week (Sunday is 1L and Saturday is 7L). quick_mday returns the day of the month. Returns as an integer.

D is designed with the simplicity of c in mind to create vectors of Dates.

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(file.path(tempdir(), "test.txt"))

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

D("01-01-2016", format = "%m-%d-%Y")
D("2004-03-02")
D(1L, 2L, origin = D("1980-01-01"))

get_age(D("1987-05-02"), D("2016-02-23"))
quick_year(D("1987-05-02"))
quick_wday(D("1987-05-02"))
quick_mday(D("1987-05-02"))
```

Index