Package 'jwutil'

October 18, 2016

Title Tools for Data Manipulation and Testing

Version 1.1.1

Date 2016-10-11

Description This is a set of simple utilities for various data manipulation and testing tasks. The goal is to use base tools well, without bringing in many dependencies. Main areas of interest are semi-automated data frame manipulation, such as converting factors in multiple binary indicator columns. There are testing functions which provide 'testthat' expectations to permute arguments to function calls. There are functions and data to test extreme numbers, dates, and bad input of various kinds which should allow testing failure and corner cases, which can be used for fuzzing your functions. The test suite has many examples of usage.

URL https://github.com/jackwasey/jwutil

Depends R (>= 3.0.0)

Imports checkmate, methods, Rcpp, stats, utils

Suggests testthat, knitr, lintr

License GPL-3

BugReports https://github.com/jackwasey/jwutil/issues

Encoding UTF-8

LinkingTo Rcpp

RoxygenNote 5.0.1

NeedsCompilation yes

Author Jack O. Wasey [aut, cre, cph]

Maintainer Jack O. Wasey <jack@jackwasey.com>

Repository CRAN

Date/Publication 2016-10-18 10:46:33

R topics documented:

jwutil-package . . . 3
add_time_to_date . . . 3
affixFields . . . 4
allIsInteger . . . 5
allIsNumeric . . . 5
areNumeric . . . 6
asCharacterNoWarn . . . 6
asNumericNoWarn . . . 7
bad_input . . . 7
binary_col_names . . . 8
buildLinearFormula . . . 8
combn_subset . . . 9
countIsNa . . . 9
countNonNaCumulative . . . 10
countNonNaPairs . . . 10
countNotNumeric . . . 11
countNumeric . . . 11
dput_expect_equal . . . 12
dropDuplicateFields . . . 12
dropRowsWithNAField . . . 13
expandFactors . . . 13
expect_that_combine_all_args . . . 14
extreme_numbers . . . 15
factorToDataframeLogical . . . 15
fillMissingCombs . . . 16
filterBetter . . . 16
flattenList . . . 17
getDropped . . . 17
getFactorNames . . . 18
getNAFields . . . 18
invwhich . . . 19
is.Date . . . 19
isFlat . . . 20
isRowSorted . . . 20
isValidTime . . . 21
listTrim . . . 21
listTrimFlat . . . 22
logicalToBinary . . . 22
lsf . . . 23
lsos . . . 23
lsp . . . 24
mergeBetter . . . 24
mergeLists . . . 25
numbers_to_long_and_float . . . 26
opt_binary_brute . . . 26
permute . . . 27

permuteWithRepeats . . . 27
platformIsLinux . . . 28
propIsNa . . . 28
propNaPerField . . . 29
propRowSorted . . . 29
random_test_dates . . . 30
random_test_numbers . . . 30
read.zip.url . . . 31
read_xlsx_linux . . . 32
rm_r . . . 32
set_attr_in_place . . . 33
shuffle . . . 34
source_purl . . . 34
strip . . . 35
stripForFormula . . . 35
strMultiMatch . . . 36
trim . . . 36
zeroes . . . 37
zero_na . . . 37
%nin% . . . 38

Index **39**

`jwutil-package` *Tools for data manipulation not found elsewhere, and testthat extensions*

Description

This is a set of simple utilities for various data manipulation and testing tasks. The goal is to use base tools well, without bringing in many dependencies. Main areas of interest are semi-automated data frame manipulation, such as converting factors in multiple binary indicator columns. There are testing functions which provide 'testthat' expectations to permute arguments to function calls. There are functions and data to test extreme numbers, dates, and bad input of various kinds which should allow testing failure and corner cases, which can be used for fuzzing your functions. The test suite has many examples of usage.

`add_time_to_date` *convert separate lists of dates and times to POSIXlt objects*

Description

Some datetime data is presented as a separate dates and times. This function restores the full datetime.

Usage

```
add_time_to_date(tms, dts, verbose = FALSE)
```

Arguments

tms	vector of times, i.e. number in range 0 to 2400, as string or integer, with or without trailing zeros
dts	vector of dates, in string format %Y-%m-%d or simple R Date objects
verbose	single logical value, if TRUE then produce verbose messages

Value

vector of POSIXlt date-times

---

affixFields *update a set of data frame field names*

Description

prefix or suffix

Usage

```
affixFields(fields, affix, skip = NULL, renameHow = c("suffix", "prefix"),
  sep = ".")
```

Arguments

fields	char vector
affix	character
skip	char vector, defaults to include all fields
renameHow	should be "suffix" or "prefix", default is suffix
sep	default "."

Value

character vector, same length as fields

`allIsInteger` *check whether vector represents all integer values, not that the same as* `is.integer`

Description

check whether all the items of input vector are integer as.integer

Usage

```
allIsInteger(x, tol = 1e-09, na.rm = TRUE)
```

Arguments

`x`	is a vector to be tested
`tol`	single numeric, default if less than 1e-9 from an integer then considered an integer.
`na.rm`	single logical, passed on to `all`

Value

logical scalar

`allIsNumeric` *check whether character vector represents all numeric values*

Description

check whether all the items of input vector are numeric without throwing warning derived from Hmsic package

Usage

```
allIsNumeric(x, extras = c(".", "NA", NA))
```

Arguments

`x`	is a character vector to be tested
`extras`	is a vector of character strings which are counted as NA values, defaults to '.' and 'NA'. Also allow `NA`.

Value

logical scalar

areNumeric *which elements of a vector are numeric*

Description

test without throwing a warning

Usage

```
areNumeric(x, extras = c(".", "NA", NA))
```

Arguments

x	vector
extras	character vector containing acceptable alternatives to numeric values which will result in returning TRUE for that element. Default is c(".", "NA", NA).

Value

logical vector of same length as input

Examples

```
areNumeric(c("1","2","3"))
areNumeric(c("1L", "2.2"))
areNumeric(c("NA", NA, ".", "", "-1.9"))
```

asCharacterNoWarn *convert factor or vector to character without warnings*

Description

correctly converts factors to vectors, and then converts to character, which may silently introduce NAs

Usage

```
asCharacterNoWarn(x)
```

Arguments

x	is a vector, probably of numbers of characters

Value

character vector, may have NA values

asNumericNoWarn *convert factor or vector to numeric without warnings*

Description

correctly converts factors to vectors, and then converts to numeric or integer, which may silently introduce NAs. Invisible rounding errors can be a problem going from numeric to integer, so consider adding tolerance to this conversion. `asIntegerNoWarn` silently `floor`s.

"are" functions return a value for each input, where is "allIs" functions return a single logical.

Usage

```
asNumericNoWarn(x)

asIntegerNoWarn(x)

areIntegers(x, tol = 1e-09, na.ignore = FALSE)
```

Arguments

`x`	is a vector, probably of numbers of characters
`tol`	tolerance when considering if two numbers are integers, default 1e-9
`na.ignore`	logical, if TRUE will pass through NA values, otherwise, they are marked FALSE.

Value

numeric vector, may have NA values

logical vector

bad_input *bad input data for tests*

Description

a variety of horrible data

Usage

```
bad_input
```

Format

An object of class `list` of length 42.

binary_col_names *names of fields which are numeric, binary or combinations thereof*

Description

Doesn't make any allowance for factors.

Usage

```
binary_col_names(x, invert = FALSE)

binary_cols(x, invert = FALSE)

numeric_col_names(x, invert = FALSE)

numeric_cols(x, invert = FALSE)
```

Arguments

x	data frame
invert	single logical, if true, will return non-binary columns

Value

vector of column names

Examples

```
dat <- data.frame(c("a", "b"), c(TRUE, FALSE), c(1, 0), c(1L, 0L),
                  c(1L, 2L), c(0.1, 0.2), c("9", "8"))
names(dat) <- c("char", "bin", "binfloat", "binint",
                "int", "float", "charint")
binary_cols(dat)
binary_col_names(dat)
binary_col_names(dat, invert = TRUE)
```

buildLinearFormula *build simple linear formula from variable names*

Description

build simple linear formula from variable names given by two character vectors. TODO: allow unquoted names.

Usage

```
buildLinearFormula(left, right)
```

Arguments

left	character vector
right	character vector

Value

formula

---

combn_subset *all unique combinations of a vector and all its non-zero subsets*

Description

all unique combinations of a vector and all its non-zero subsets

Usage

```
combn_subset(x)
```

Arguments

x	vector to be subsetted and combined

Value

list of vectors with all combinations of x and its subsets

---

countIsNa *count NA in vector*

Description

count the number of NAs in a vector

Usage

```
countIsNa(x)
```

Arguments

x	vector

Value

integer

countNonNaCumulative *running totals of number of non-NA values in consecutive fields*

Description

counts non-NA fields in first field, then progreses through fields, OR new field and saves running total for each field TODO: tests

Usage

```
countNonNaCumulative(d)
```

Arguments

d	data.frame

Value

vector of cumulative non-NA counts with names corresponding to the given data frame

countNonNaPairs *count which combinations of fields have at least one non-NA*

Description

cycles through the given data frame twice, and applies logical OR to all elements of each column it then counts how many of these pairs are not-na, i.e. have at least one non-NA value TODO: tests

Usage

```
countNonNaPairs(d)
```

Arguments

d	data.frame

Value

matrix with nrow and ncol being the number of fields in the given dataframe

countNotNumeric *count non-numeric elements*

Description

counts the number of non-numeric elements in a vector, without throwing warnings

Usage

```
countNotNumeric(x)
```

Arguments

x	is usually a charcter vector

Details

did have `extras = c(".", "NA"))`

Value

integer

countNumeric *count numeric elements*

Description

counts the number of numeric elements in a vector, without throwing warnings

Usage

```
countNumeric(x)
```

Arguments

x	is usually a character vector

Value

integer

dput_expect_equal *dput a testthat test*

Description

Generate an R expression containing a testthat expectation for the given expression and its result. This is useful when you know that a certain output is correct, and wish to generate a test case to reflect this.

Usage

```
dput_expect_equal(...)
```

Arguments

...	expressions

Value

character vector with each element containing an R expression with expect_equal test case corresponding to the evaluated input expressions.

Examples

```
dput_expect_equal("a" %nin% c("b", "c", "d"))
```

dropDuplicateFields *drop duplicate fields*

Description

compares all data in each field to every other field, and drops the latter match. Will find multiple matches. Doesn't do any type conversions yet. This is purely by content, not by field name.

Usage

```
dropDuplicateFields(df, verbose = FALSE)
```

Arguments

df	data.frame
verbose	single logical value, if TRUE then produce verbose messages

Value

data frame without duplicate fields

dropRowsWithNAField *drops rows with NA values in specified fields*

Description

employs `stats::complete.cases` which is fast internal C code. Returns a data frame with unused factor levels dropped (these may have been introduced by dropping rows with some NA values)

Usage

```
dropRowsWithNAField(x, fld = names(x), verbose = FALSE)
```

Arguments

`x`	data frame
`fld`	vector with names of fields which must have no NA values
`verbose`	single logical value, if `TRUE` then produce verbose messages

Value

data frame without rows containing NA in the specified data fields. There may be NA values in the resulting data frame in fields which are not listed in fld.

expandFactors *Takes factors from a data frame and converts them to true/false fields with appropriately named fields.*

Description

For a two level factor, this is relatively easy, since we just replace the field with `x==levels(x)[1]` or something like that, and rename the field to indicate that TRUE is level 1 of the factor. This works well for gender. For multi-level factors there is redundancy with multiple new fields now containing FALSE, with only one TRUE for the matching level.

Usage

```
expandFactors(x, consider = names(x), sep = "", na.rm = TRUE,
  verbose = FALSE)
```

Arguments

x	data.frame to search for factors to convert
consider	character vector of field names in the data frame to consider. Defaults to all fields
sep	character scalar used to separate field prefixes from factor values in new column names
na.rm	logical scalar: if NA data and/or NA levels, then covert to NA strings and expand these as for any other factor
verbose	single logical value, if TRUE then produce verbose messages

Value

data.frame with no factors

See Also

PSAgraphics::cv.trans.psa

---

expect_that_combine_all_args

alternative expect_that *from* testthat *which permutes all the inputs to a function which should give the same result where n args >=2 and the function is commutative.*

Description

This makes a lot of assumptions, needs more testing. It can't handle mixed error/no error outcomes after permutation, which is an important feature to consider. The command following this function attaches this function to the testthat namespace. This means that it can call internal testthat functions, but does not mean it appears as testthat::expect_that_combine

Usage

```
expect_that_combine_all_args(object, condition, info = NULL, label = NULL)

expect_that_combine_first_arg(object, condition, info = NULL, label = NULL)
```

Arguments

object	object to test
condition	a function that returns whether or not the condition is met, and if not, an error message to display.
info	extra information to be included in the message (useful when writing tests in loops).
label	object label. When NULL, computed from deparsed object.

Value

testthat result

Examples

```
expect_that_combine_all_args(sum(1, 2, 3),
 testthat::equals(6))
```

---

extreme_numbers *extreme numbers*

Description

very biggest and smallest non-zero numbers the current machine can handle, positive and negative.

Usage

```
extreme_numbers
```

Format

An object of class `numeric` of length 6.

---

factorToDataframeLogical *convert factor into a data.frame of logicals*

Description

converts a single factor into a data.frame with multiple T/F fields, one for each factor

Usage

```
factorToDataframeLogical(fctr, prefix = deparse(substitute(fctr)), sep = "",
  na.rm = TRUE, verbose = FALSE)
```

Arguments

`fctr`	factor
`prefix`	defaults to "f" to pre-pend the factor level when constructing the data frame columns names
`sep`	scalar character, introduced between factor names and levels when forming new data frame column names
`na.rm`	logical scalar: if NA data and/or NA levels, then covert to NA strings and expand these as for any other factor
`verbose`	single logical value, if `TRUE` then produce verbose messages

Value

data.frame with columns of logicals

`fillMissingCombs` *fill out missing combinations of factors with NA*

Description

fill out missing combinations of factors with NA

Usage

```
fillMissingCombs(df)
```

Arguments

`df`	data frame

Details

Adapated from http://www.cookbook-r.com/Manipulating_data/Summarizing_data/#using-aggregate

---

`filterBetter` *filter data with diagnostics*

Description

applies an expression to a data frame, and gives information about the numbers of dropped rows.

Usage

```
filterBetter(x, expr, verbose = TRUE)
```

Arguments

`x`	data frame
`expr`	expression in the context of the data frame, i.e. the terms should be column names.
`verbose`	logical default is TRUE

Value

filtered data frame

flattenList *flatten a list*

Description

unlike unlist, this function returns a list of objects of different data types, but removes any depth

Usage

```
flattenList(..., na.rm = FALSE)
```

Arguments

...	list or any set of objects which will be made into a list, may include lists and nested lists
na.rm	will drop NA values if TRUE

Value

list without nested lists, objects with preserved data types

Source

https://stackoverflow.com/questions/8139677/\ how-to-flatten-a-list-to-a-list-without-coercion

getDropped *get items or numerics that would be dropped in a merge*

Description

converts both vectors to numeric. This simulates merging when one key is character (but contains integer numbers), and another key is stored as integer.

Usage

```
getDropped(x, y)
```

Arguments

x	vector or factor
y	vector or factor

Value

list of two vectors

getFactorNames *get names of the factor fields in a data frame*

Description

Get the names of those fields in a data frame which are factors.

Usage

```
getFactorNames(x, consider = names(x))

getNonFactorNames(x, consider = names(x))
```

Arguments

x	data frame
consider	character vector of field names of the data frame to test, default is to use all of them.

Value

vector

getNAFields *get NA field names from data frame*

Description

Get the names of any columns in a data frame which have NA values.

Usage

```
getNAFields(dframe)

getNonNAFields(dframe)
```

Arguments

dframe	data.frame

Value

vector of names of fields which contain any NA values, length zero if no matches

invwhich *inverse which*

Description

for a given vector of ordinals which would reference items in a vector, list etc, `invwhich` returns a logical vector with TRUE for the cited positions. If length is not provided, the maximum index is used.

Usage

```
invwhich(which, len = max(which))
```

Arguments

`which`	integer vector of indices, as would be produced by `which`
`len`	integer scalar: length of return vector, defaults to `max(which)`

Value

logical vector of length `length`

is.Date *is the object a* `Date`

Description

copied from lubridate

Usage

```
is.Date(x)
```

Arguments

`x`	object to test

Value

logical

isFlat *determine whether a list is nested*

Description

Returns TRUE if the given list is not nested.

Usage

```
isFlat(x)
```

Arguments

x	list

Value

single logical

isRowSorted *is every row sorted?*

Description

Quicky run through rows of a matrix looking for any non-ascending rows in C++

Usage

```
isRowSorted(x)
```

Arguments

x	matrix, each row containing ordered or disordered numerics

isValidTime *check if a time is valid in 24h clock*

Description

allow leading and trailing space, optional colon in middle, 2400 is not allowed. TODO: can lubridate do this better?

Usage

```
isValidTime(tms, na.rm = FALSE)
```

Arguments

tms	is a vector of characters which may represent times
na.rm	logical if true, will ignore NA values, otherwise these will test as invalid.

Value

logical vector, with NA out if NA given

listTrim *trim null or empty values from a list*

Description

delele null/empty entries in a list. Recursively looks through list if nested.

Usage

```
listTrim(x)
```

Arguments

x	list

Value

trimmed list

listTrimFlat *trim null or empty values from a list*

Description

Trim NULL or empty values from a flat list.

Usage

```
listTrimFlat(x)
```

Arguments

x	list

Value

trimmed list

logicalToBinary *encode TRUE as 1, and FALSE as 0 (integers)*

Description

when saving data as text files for distribution, printing large amounts of text containing TRUE and FALSE is inefficient. Convert to binary takes more R memory, but allows more compact output TODO: test

Usage

```
logicalToBinary(x)
```

Arguments

x	dataframe which may contain logical fields

Value

data frame without logical fields

lsf *list all functions in a package*

Description

List functions in a package

Usage

```
lsf(pkg)
```

Arguments

pkg	character string containing package name

Value

character vector of functions in given package

lsos *show largest objects*

Description

https://gist.github.com/1187166.git Taken from http://stackoverflow.com/questions/1358003/\ tricks-to-manage-the-available-memory-in-an-r-session

Usage

```
lsos(..., n = 10)
```

Arguments

...	arguments passed on to `.ls.objects`
n	scalar integer, number of objects to show

lsp *list all items in a package*

Description

default to including (?private) functions beginning with '.'

Usage

```
lsp(package, all.names = TRUE, pattern)
```

Arguments

package	is the (unquoted) name of the package
all.names	= TRUE, set to FALSE to ignore items beginning with a period
pattern	= optional pattern to match

Value

character vector of package contents

mergeBetter *merge better*

Description

apply built-in R merge but with additional features:

Usage

```
mergeBetter(x, y, by.x, by.y, all.x = FALSE, all.y = FALSE, affix = NULL,
  renameConflict = c("suffix", "prefix"), renameAll = c("no", "suffix",
  "prefix"), convert_factors = TRUE, verbose = FALSE)
```

Arguments

x	data frame
y	data frame
by.x	field in x to merge on. Unlike merge, this is compulsory.
by.y	field in y to merge on. Unlike merge, this is compulsory.
all.x	outer join to keep all x values
all.y	outer join to keep all y values
affix	either prefix or suffix to disambiguate files. By default, this is the name of the table specified in y. In all other respects in this function, x and y are symmetric.

`renameConflict` - determines whether prefix or suffix is added to disambiguate conflicting column names. Value can be "suffix", "prefix". Suffix is the default.

`renameAll` - regardless of column name clashes, "prefix" or "suffix" with every field with original table name, or "no" for neither

`convert_factors`
Default is TRUE which causes factors to be converted to character before merge. This is almost certainly safer.

`verbose` logical or numbers 0, 1 or 2. 1 or TRUE will give moderate verbosity, 2 will give full verbosity. 0 or FALSE turns off all messages.

Value

merged data frame

`mergeLists` *merge lists by names*

Description

merge lists by vector combining all the vector elements of the list items with the matching names. Unnamed vectors in the list will be dropped silently.

Usage

```
mergeLists(x, y)
```

Arguments

`x` unnested list with named elements, each of which is a vector

`y` unnested list with named elements, each of which is a vector

Value

list of vectors

numbers_to_long_and_float

convert numbers to long and float types

Description

intended for generating values for stress testing functions

Usage

```
numbers_to_long_and_float(..., na.rm = TRUE)
```

Arguments

...	list of values to convert to long and double
na.rm	logical, defaults to TRUE, so output contains only long and float values.

Value

list of long and double versions of convertable values from the input

opt_binary_brute

optimizes a function for all combinations of all subsets

Description

takes a data frame and optimization function

Usage

```
opt_binary_brute(x, fun = opt_binary_fun, verbose = TRUE)
```

Arguments

x	data frame
fun	function which takes parameters x = data.frame, n = columns
verbose	single logical value, if TRUE then produce verbose messages

permute *generate all permutations of input*

Description

systematically permute the input vector or list, which is very slow for long x. Am amazed something this simple isn't either in base R, or in a straightforward form in a package.

TODO: limit to a certain cut-off, after which we randomly sample

Usage

```
permute(x)
```

Arguments

x	list or vector

Value

data frame, each row being one permutation

permuteWithRepeats *generate all permutations of input, reusing values in each result row*

Description

systematically permute the input vector or list

Usage

```
permuteWithRepeats(x)
```

Arguments

x	list or vector

Value

data frame, each row being one permutation

platformIsLinux *Are we running on Linux, Mac or Windows?*

Description

Are we running on Linux, Mac or Windows?

Usage

```
platformIsLinux()

platformIsWindows()

platformIsMac()
```

Value

logical

propIsNa *Proportion of NA values in a vector*

Description

get fraction of NA in a vector

Usage

```
propIsNa(x)
```

Arguments

x	is a vector which may have NA values

Value

numeric proportion of NAs in the supplied vector

propNaPerField *return proportion of NA values per field*

Description

Return proportion of values which are NA in each field of the given data frame.

Usage

```
propNaPerField(dframe)
```

Arguments

dframe	is a data frame

Value

numeric vector

propRowSorted *proportion of non-descending rows in matrix*

Description

first performs isRowSorted to get a logical vector, then sums TRUE values and takes fraction of total

Usage

```
propRowSorted(x)
```

Arguments

x	matrix, each row containing ordered or disordered numerics

Value

double, the proportion from 0 to 1

random_test_dates *generate random Dates or POSIXlt test datetimes*

Description

generate random Dates and POSIXlt test datetimes

Usage

```
random_test_dates(n = nums_in_tests, origin = as.Date("2000-01-01"),
  dayspread = 365 * 150)

random_test_posixlt_datetimes(n = nums_in_tests,
  origin = as.Date("2000-01-01"), dayspread = 365 * 150)
```

Arguments

n	integer number to generate
origin	Date defaults to Jan 1, 2000.
dayspread	integer number of days either side of origin to pick random dates from, defaults to 150 years.

Value

vector of POSIXlt datetimes or Dates

random_test_numbers *create extreme random numbers*

Description

create random Dates, POSIX dates, letters and numbers. The numbers explore limits of R precision and floating point and integer ranges. Zero, negatives, positives.

Usage

```
random_test_numbers(n = nums_in_tests, min = NULL, max = NULL,
  hole = NULL)

random_test_integers(n = nums_in_tests, min = -.Machine$integer.max,
  max = .Machine$integer.max, hole = NULL)

random_test_letters(n = nums_in_tests, max_str_len = 257)
```

Arguments

n	integer number of each group to generate
min	optional minimum number
max	optional maximum number
hole	is a closed range of numbers not to include, e.g. c(1,2) would discard 1, 1.1 pi/2 and 2
max_str_len	integer scalar, maximum length of possible strings created, as distinct from number of strings given by n

Value

vector length 5n+1 containing variety of difficult numbers for testing purposes

---

read.zip.url *read file from zip at URL*

Description

downloads zip file, and opens named file filename, or the single file in zip if filename is not specified. FUN is a function, with additional arguments to FUN given by .... @details TODO: update from icd package

Usage

```
read.zip.url(url, filename = NULL, FUN = readLines, ...)
```

Arguments

url	character vector of length one containing URL of zip file.
filename	character vector of length one containing name of file to extract from zip. If not specified, and the zip contains a single file, then this single file will be used.
FUN	function used to process the file in the zip, defaults to readLines. The first argument to FUN will be the path of the extracted filename
...	further arguments to FUN

read_xlsx_linux *read* .xlsx *file, interpret as CSV, and return a data frame*

Description

currently relies on Linux xlsx2csv command, but could potentially be done with VB script in Windows. This offers a different backend to other Excel parsing functions in R,

Usage

```
read_xlsx_linux(file)
```

Arguments

file	is the path to the .xlsx file

Value

data frame

See Also

readxl package by Hadley Wickham

rm_r *recursive remove*

Description

search through environments until the variables in the list x are all gone. This doesn't delete functions. No barrier to infinite recursion, but rm should be able to delete anything that exists can see.

Usage

```
rm_r(x, envir = parent.frame())
```

Arguments

x	variables to annihilate
envir	environment to start at, defaults to calling frame.

set_attr_in_place *Set attribute on any SEXP in place*

Description

Set attribute on any SEXP in place

Usage

```
set_attr_in_place(x, name, value)
```

Arguments

x	Any R object for which an attribute can be set
name	The name of the attribute, as length one character vector
value	Value to be assigned to that attribute

Examples

```
## Not run:
# benchmark to see whether setting an attribute on a function argument and returning it does a copy.
f <- function(x) { attr(x, "a") <- FALSE; x }
p <- generate_random_short_icd9(1)
q <- generate_random_short_icd9(1e6)
times <- 1e5
microbenchmark::microbenchmark(f(p), set_attr_in_place(p, "a", FALSE),
                               f(q), set_attr_in_place(q, "a", FALSE), times = times)
p2 <- p3 <- generate_random_short_icd9(1)
set_attr_in_place(p2, "a", FALSE)
stopifnot(identical(f(p3), p2))
# oh dear, \code{f} does copy.

# see if we can return without deep copy using Rcpp:

p4 <- p
q4 <- q
setDecimalCodeInPlace(p4)
stopifnot(identical(p4, setDecimalCode(p4)))
microbenchmark::microbenchmark(setDecimalCodeInPlace(p), setDecimalCode(p), times = times)
microbenchmark::microbenchmark(setDecimalCodeInPlace(q), setDecimalCode(q), times = times)



## End(Not run)
```

shuffle *shuffle*

Description

randomly shuffle the order of a vector or list. This is to improve quality of bad data to throw at functions when testing.

Usage

```
shuffle(x)
```

Arguments

x	list or vector

Value

list or vector of same length as input, (probably) in a different order

source_purl *extract code from knitr vignette and source it*

Description

extract code from knitr vignette and source it

Usage

```
source_purl(input, output = file.path(tempdir(), paste0(basename(input),
  ".R")), documentation = 1L, ...)
```

Arguments

input	path to file as single character string
output	output file path, defaults to a file in a temporary name based on input
documentation	single integer value passed on to knitr::purl. An integer specifying the level of documentation to go the tangled script: 0 means pure code (discard all text chunks); 1 (default) means add the chunk headers to code; 2 means add all text chunks to code as roxygen comments
...	further parameters passed to source

strip *strip all whitespace*

Description

could do this with regular expression, but slow, and this function is called frequently. My only use case works with removal of all space character whitespace, and I don't expect <TAB>. This uses non-unicode aware matching for speed. This can be changed by setting useBytes to FALSE.

Usage

```
strip(x, pattern = " ", useBytes = TRUE)
```

Arguments

x	is a character vector to strip
pattern	is the non-regex of the character to strip, default " "
useBytes	logical scalar. Unlike gsub, this will default to TRUE here, therefore breaking unicode.

Value

character vector

stripForFormula *strip a string so that it can be used as a variable name in a formula.*

Description

This excludes many symbols, so just strip all symbols leaving alphanumeric, and no whitespace.

Usage

```
stripForFormula(x)
```

Arguments

x	character vector of potential formula variables

Value

character vector of length x

strMultiMatch *return the actual matches from a bracketed regex*

Description

Be careful: this may throw funny results for exotic regex, but so far, it seems okay. it also drops the first result which always seems to be a duplicate or whole-string match

`strPaitMatch` differs in that there should only be two pairs of parenthesis, then the first (by default) becomes the name, and the second the value.

Usage

```
strMultiMatch(pattern, text, dropEmpty = FALSE, ...)

strPairMatch(pattern, text, swap = FALSE, dropEmpty = FALSE, ...)
```

Arguments

`pattern`	regular expression: if it has bracketed sections, these submatches are returned
`text`	is the string to match against. This vector should be the same length as the pattern vector, or the patern vector should be length one.
`dropEmpty`	logical whether to drop rows with no matches
`...`	are additional parameters passed to regexec and regmatches. I haven't tried this: it may need two separate variables containing lists of params, since this will send everything to both functions.
`swap`	logical scalar, whether to swap the names and values. Default is not to swap, so the first match becomes the name.

Value

list of character vectors, list length being the length of the inptu text vector.

trim *strip whitespace from ends of each string in given character vector*

Description

slower than `strip`.

Usage

```
trim(x)
```

Arguments

x	is a character vector to trim

Value

character vector

---

zeroes *zeroes*

Description

long and float types

Usage

```
zeroes
```

Format

An object of class `list` of length 2.

---

zero_na *zero NA values in a data.frame*

Description

Zero NA values in a data.frame, including `cols` and exluding `ignore`. Also does not replace `Date` or `POSIXt` fields.

Usage

```
zero_na(df, cols = names(df), ignore = character(), verbose = FALSE)
```

Arguments

df	data.frame
cols	names of columns to work on, default is all columns
ignore	cahracter vector of columns names to ignore
verbose	TRUE or FALSE

`%nin%` *inverse of %in%*

Description

borrowed from Hmisc. See nomatch = 0L) > 0L

Usage

```
x %nin% table
```

Arguments

`x`	is the vector of values to be matched
`table`	is actually a vector, to be matched against

Value

logical vector of length of x

Index

∗Topic **manip**
    logicalToBinary, 22
∗Topic **sysdata**
    bad_input, 7
    extreme_numbers, 15
    zeroes, 37
%nin%, 38

add_time_to_date, 3
affixFields, 4
allIsInteger, 5
allIsNumeric, 5
areIntegers *(*asNumericNoWarn*)*, 7
areNumeric, 6
asCharacterNoWarn, 6
asIntegerNoWarn *(*asNumericNoWarn*)*, 7
asNumericNoWarn, 7

bad_input, 7
binary_col_names, 8
binary_cols *(*binary_col_names*)*, 8
buildLinearFormula, 8

combn_subset, 9
countIsNa, 9
countNonNaCumulative, 10
countNonNaPairs, 10
countNotNumeric, 11
countNumeric, 11

dput_expect_equal, 12
dropDuplicateFields, 12
dropRowsWithNAField, 13

expandFactors, 13
expect_that_combine_all_args, 14
expect_that_combine_first_arg
        *(*expect_that_combine_all_args*)*,
        14
extreme_numbers, 15

factorToDataframeLogical, 15
fillMissingCombs, 16
filterBetter, 16
flattenList, 17
floor, *7*

getDropped, 17
getFactorNames, 18
getNAFields, 18
getNonFactorNames *(*getFactorNames*)*, 18
getNonNAFields *(*getNAFields*)*, 18

invwhich, 19
is.Date, 19
isFlat, 20
isRowSorted, 20
isValidTime, 21

jwutil *(*jwutil-package*)*, 3
jwutil-package, 3

listTrim, 21
listTrimFlat, 22
logicalToBinary, 22
lsf, 23
lsos, 23
lsp, 24

mergeBetter, 24
mergeLists, 25

numbers_to_long_and_float, 26
numeric_col_names *(*binary_col_names*)*, 8
numeric_cols *(*binary_col_names*)*, 8

opt_binary_brute, 26

permute, 27
permuteWithRepeats, 27
platformIsLinux, 28
platformIsMac *(*platformIsLinux*)*, 28

platformIsWindows *(*platformIsLinux*)*, 28
propIsNa, 28
propNaPerField, 29
propRowSorted, 29

random_test_dates, 30
random_test_integers
*(*random_test_numbers*)*, 30
random_test_letters
*(*random_test_numbers*)*, 30
random_test_numbers, 30
random_test_posixlt_datetimes
*(*random_test_dates*)*, 30
read.zip.url, 31
read_xlsx_linux, 32
rm_r, 32

set_attr_in_place, 33
shuffle, 34
source_purl, 34
strip, 35
stripForFormula, 35
strMultiMatch, 36
strPairMatch *(*strMultiMatch*)*, 36

trim, 36

zero_na, 37
zeroes, 37