Package ‘qdapTools’

May 10, 2023

Type Package
Title Tools for the 'qdap' Package
Version 1.3.7
Maintainer Tyler Rinker <tyler.rinker@gmail.com>
Depends R (>= 3.0.0)
Imports chron, data.table (>= 1.9.6), methods, RCurl, XML
Suggests testthat
Description A collection of tools associated with the 'qdap' package that may be useful outside of the context of text analysis.
License GPL-2
URL https://github.com/trinker/qdapTools
BugReports https://github.com/trinker/qdapTools/issues
Encoding UTF-8
RoxygenNote 7.2.3
NeedsCompilation no
Author Bryan Goodrich [ctb],
    Dason Kurkiewicz [ctb],
    Kirill Muller [ctb],
    Tyler Rinker [aut, cre]
Repository CRAN
Date/Publication 2023-05-10 07:20:02 UTC

R topics documented:

hash ................................................................. 2
hms2sec ............................................................. 4
id ................................................................. 4
list2df ............................................................. 5
loc_split ........................................................... 8
lookup ........................................................... 10
hash

Description

hash - Creates a data.table based hash table for quick hash style dictionary lookup.
hash_look - Works with a hash table such as is returned from hash, to lookup values.
%hl% - A binary operator version of hash_look.
%hl+% - A binary operator version of hash_look for when missing is assumed to be NULL.
hash_e - Creates a new environment for quick hash style dictionary lookup.

Usage

hash(x)

hash_look(terms, key, missing = NA)

terms %hl% key

terms %hl+% key

hash_e(x, mode.out = "numeric")

Arguments

x A two column dataframe.
terms A vector of terms to undergo a lookup.
key The hash key to use.
missing Value to assign to terms not found in the hash table.
mode.out The type of output (column 2) expected (e.g., "character", "numeric", etc.)
Value

hash - Creates a "hash table", a two column data.table.
hash_e - Creates a "hash table", a two column data.frame in its own environment.

Author(s)

hash_e - Bryan Goodrich and Tyler Rinker <tyler.rinker@gmail.com>.

References

https://www.talkstats.com/showthread.php/22754-Create-a-fast-dictionary

See Also

setDT, hash
environment

Examples

##===================##
data.table Hashes ##
##===================##
(DF <- aggregate(mpg~as.character(carb), mtcars, mean))
x <- sample(DF[, 1], 20, TRUE)
new.hash <- hash(DF)
x2 <- c(9, 12, x)
hash_look(x, new.hash)
x %hl% new.hash
x2 %hl% new.hash
x2 %hl+% new.hash

## Create generic functions
hfun <- function(x, ...) {
  hsh <- hash(x, ...)
  function(x, ...) hash_look(x, hsh, ...)
}
m <- hfun(DF)
m(x)

## Environment Hashes ##

new.hash2 <- hash_e(DF)
x %hl% new.hash2
x2 %hl% new.hash2
x2 %hl+% new.hash2
**hms2sec**  
*Convert h:m:s To/From Seconds*

**Description**

hms2sec - Converts a vector of h:m:s to seconds.  
sec2hms - Converts a vector of seconds to h:m:s.

**Usage**

```r
hms2sec(x)
sec2hms(x)
```

**Arguments**

*x*  
A vector of times in h:m:s (for hms2sec) or seconds (for sec2hms).

**Value**

hms2sec - Returns a vector of times in seconds.  
sec2hms - Returns a vector of times in h:m:s format.

**See Also**

*times*

**Examples**

```r
hms2sec(c("02:00:03", "04:03:01"))
hms2sec(sec2hms(c(222, 1234, 55)))
sec2hms(c(256, 3456, 56565))
```

---

**id**  
*ID By Row Number or Sequence Along*

**Description**

Generate a sequence of integers the `length/ncol` of an object.

**Usage**

```r
id(x, prefix = FALSE, pad = TRUE, ...)
```
list2df

Arguments

x  A dataframe, matrix, vector, or list object.

prefix  A character string to use as a prefix. FALSE or NULL results in no prefix being used. TRUE will utilize the prefix "X."

pad  logical. If TRUE the beginning number will be padded with zeros.

...  Other arguments passed to pad.

Value

Returns a vector of sequential integers.

Examples

id(list(1, 4, 6))
id(matrix(1:10, ncol=1))
id(mtcars)
id(mtcars, TRUE)
id("w")
id(mtcars, prefix="id-")

## Not run:
library(qdap)
question_type(DATA.SPLIT$state, id(DATA.SPLIT, TRUE))

## End(Not run)

---

list2df  List/Matrix/Vector to Dataframe/List/Matrix

Description

list2df - Convert a named list of vectors to a dataframe.

matrix2df - Convert a matrix to a dataframe and convert the rownames to the first column.

vect2df - Convert a named vector to a dataframe.

list_df2df - Convert a list of equal numbered/named columns to a dataframe using the list names as the level two variable.

list_vect2df - Convert a list of named vectors to a hierarchical dataframe.

counts2list - Convert a count matrix to a named list of elements.

vect2list - Convert a vector to a named list.

df2matrix - Convert a dataframe to a matrix and simultaneously move a column (default is the first column) to the rownames of a matrix.

matrix2long - Convert a matrix to a long format dataframe where column names become column 1, row names, column 2 and the values become column 3.
Usage

\begin{verbatim}
list2df(list.object, col1 = "X1", col2 = "X2")

matrix2df(matrix.object, col1 = "var1")

vect2df(vector.object, col1 = "X1", col2 = "X2", order = TRUE, rev = FALSE)

list_df2df(list.df.object, col1 = "X1")

list_vect2df(
    list.vector.object,
    col1 = "X1",
    col2 = "X2",
    col3 = "X3",
    order = TRUE,
    ...
)

counts2list(mat, nm = rownames(mat))

vect2list(vector.object, use.names = TRUE, numbered.names = FALSE)

df2matrix(data.frame.object, i = 1)

matrix2long(matrix.object, col1 = "cols", col2 = "rows", col3 = "vals")
\end{verbatim}

Arguments

- **list.object** A named list of vectors.
- **col1** Name for column 1 (the vector elements if converting a list or the rownames if converting a matrix).
- **col2** Name for column 2 (the names of the vectors).
- **matrix.object** A matrix or simple_triplet_matrix object.
- **vector.object** A vector object.
- **order** logical. If TRUE the dataframe will be ordered.
- **rev** logical. If TRUE and order = TRUE the dataframe will be ordered in descending order.
- **list.df.object** A list of dataframes with equal number/named of columns.
- **list.vector.object** A list of dataframes with equal number/named of columns.
- **col3** The name of the third column (list_vect2df).
- **...** Further arguments passed to vect2df.
- **mat** A matrix of counts.
- **nm** A character vector of names to assign to the list.
use.names logical. If TRUE and the vector is named, these names will be transferred to the list names.

numbered.names logical. If TRUE padded numbers will be used as list names. If FALSE the vector elements themselves will become the list names.

data.frame.object A data.frame object.
i The column number or name to become the rownames of the matrix.

Value

list2df - Returns a dataframe with two columns.
matrix2df - Returns a dataframe.
vect2df - Returns a dataframe.
list_df2df - Returns a dataframe.
list_vect2df - Returns a dataframe.
counts2list - Returns a list of elements.
vect2list - Returns a list of named elements.
df2matrix - Returns a matrix.
matrix2long - Returns a long format dataframe.

See Also

mtabulate

Examples

lst1 <- list(x=c("foo", "bar"), y=1:5)
list2df(lst1)

lst2 <- list(a=c("hello", "everybody"), b = mtcars[1:6, 1])
list2df(lst2, "col 1", "col 2")

matrix2df(mtcars)
matrix2df(cor(mtcars))
matrix2df(matrix(1:9, ncol=3))

vect2df(1:10)
vect2df(c(table(mtcars[, "gear"])))

list_df2df(list(mtcars, mtcars))

L1 <- list(a=1:10, b=1:6, c=5:8)
list_vect2df(L1)

L2 <- list(
  months=setNames(1:12, month.abb),
  numbers=1:6,
  states=setNames(factor(state.name[1:4]), state.abb[1:4])
)
loc_split

Split Data Forms at Specified Locations

Description

Split data forms at specified integer locations.

Usage

loc_split(x, locs, names = NULL, ...)

## S3 method for class 'list'
loc_split(x, locs, names = NULL, ...)

## S3 method for class 'data.frame'
loc_split(x, locs, names = NULL, ...)
S3 method for class 'matrix'
loc_split(x, locs, names = NULL, ...)

S3 method for class 'numeric'
loc_split(x, locs, names = NULL, ...)

S3 method for class 'factor'
loc_split(x, locs, names = NULL, ...)

S3 method for class 'character'
loc_split(x, locs, names = NULL, ...)

Default S3 method:
loc_split(x, locs, names = NULL, ...)

Arguments

x A data form (list, vector, data.frame, matrix).
locs A vector of integer locations to split at. If locs contains the index 1, it will be silently dropped.
names Optional vector of names to give to the list elements.
... Ignored.

Value

Returns of list of data forms broken at the locs.

Note

Two dimensional object will retain dimension (i.e., drop = FALSE is used).

See Also

run_split, split_vector https://github.com/trinker/loc_split_example for practical usage.

Examples

## character
loc_split(LETTERS, c(4, 10, 16))
loc_split(LETTERS, c(4, 10, 16), c("dog", "cat", "chicken", "rabbit"))

## numeric
loc_split(1:100, c(33, 66))

## factor
(p_chng <- head(1 + cumsum(rle(as.character(CO2[["Plant"]]))[[1]]), -1))
loc_split(CO2[["Plant"]], p_chng)
## list
loc_split(as.list(LETTERS), c(4, 10, 16))

## data.frame
(vs_change <- head(1 + cumsum(rle(as.character(mtcars["vs"]))[[1]]), -1))
loc_split(mtcars, vs_change)

## matrix
(mat <- matrix(1:50, nrow = 10))
loc_split(mat, c(3, 6, 10))

lookup

**Hash Table/Dictionary Lookup**

**lookup** - Rhrefhttp://datatable.rforge.r-project.org/data.table based hash table useful for large vector lookups.

### Description

%l% - A binary operator version of `lookup` for when `key.match` is a data.frame or named list.

%l+% - A binary operator version of `lookup` for when `key.match` is a data.frame or named list and `missing` is assumed to be `NULL`.

%lc% - A binary operator version of `lookup` for when `key.match` is a data.frame or named list and all arguments are converted to character.

%lc+% - A binary operator version of `lookup` for when `key.match` is a data.frame or named list, `missing` is assumed to be `NULL`, and all arguments are converted to character.

### Usage

lookup(terms, key.match, key.reassign = NULL, missing = NA)

## S3 method for class 'list'
lookup(terms, key.match, key.reassign = NULL, missing = NA)

## S3 method for class 'data.frame'
lookup(terms, key.match, key.reassign = NULL, missing = NA)

## S3 method for class 'matrix'
lookup(terms, key.match, key.reassign = NULL, missing = NA)

## S3 method for class 'numeric'
lookup(terms, key.match, key.reassign, missing = NA)

## S3 method for class 'factor'
lookup(terms, key.match, key.reassign, missing = NA)

## S3 method for class 'character'
lookup(terms, key.match, key.reassign, missing = NA)
lookup

terms %1% key.match

terms %l+% key.match

terms %lc% key.match

terms %lc+% key.match

Arguments

terms       A vector of terms to undergo a lookup.
key.match   Takes one of the following: (1) a two column data.frame of a match key and
            reassignment column, (2) a named list of vectors (Note: if data.frame or named
            list supplied no key reassign needed) or (3) a single vector match key.
key.reassign A single reassignment vector supplied if key.match is not a two column data.frame/named
              list.
missing     Value to assign to terms not matching the key.match. If set to NULL the original
            values in terms corresponding to the missing elements are retained.

Value

Outputs A new vector with reassigned values.

See Also

setDT, hash

Examples

## Supply a dataframe to key.match
lookup(1:5, data.frame(1:4, 11:14))

## Retain original values for missing
lookup(1:5, data.frame(1:4, 11:14), missing=NULL)
lookup(LETTERS[1:5], data.frame(LETTERS[1:5], 100:104))
lookup(LETTERS[1:5], factor(LETTERS[1:5]), 100:104)

## Supply a named list of vectors to key.match

codes <- list(
    A = c(1, 2, 4),
    B = c(3, 5),
    C = 7,
    D = c(6, 8:10)
)

lookup(1:10, codes)
## Supply a single vector to key.match and key.reassign

```r
lookup(mtcars$carb, sort(unique(mtcars$carb)),
      c("one", "two", "three", "four", "six", "eight"))
```

```r
lookup(mtcars$carb, sort(unique(mtcars$carb)),
      seq(10, 60, by=10))
```

## %l%, a binary operator version of lookup
```r
1:5 %l% data.frame(1:4, 11:14)
1:10 %l% codes
```
```
1:12 %l% codes
1:12 %l+% codes
```
```
(key <- data.frame(a=1:3, b=factor(paste0("l", 1:3))))
1:3 %l% key
```

## Larger Examples
```r
key <- data.frame(x=1:2, y=c("A", "B"))
big.vec <- sample(1:2, 3000000, TRUE)
out <- lookup(big.vec, key)
out[1:20]
```

## A big string to recode with variation
## means a bigger dictionary
```r
recode.me <- sample(1:(length(LETTERS)*10), 1000000, TRUE)
```

## Time it
```r
tic <- Sys.time()
output <- recode.me %l% split(1:(length(LETTERS)*10), LETTERS)
difftime(Sys.time(), tic)
```
```
## view it
sample(output, 100)
```

---

**lookup_e**

**Hash Table/Dictionary Lookup**

`lookup_e - Environment based hash table useful for large vector lookups.`

**Description**

- `%le%` - A binary operator version of `lookup_e` for when `key.match` is a data.frame or named list.
- `%le+%` - A binary operator version of `lookup_e` for when `key.match` is a data.frame or named list and `missing` is assumed to be `NULL`. 
### Usage

```r
lookup_e(terms, key.match, key.reassign = NULL, missing = NA)
```

#### S3 method for class 'matrix'

```r
lookup_e(terms, key.match, key.reassign = NULL, missing = NA)
```

#### S3 method for class 'data.frame'

```r
lookup_e(terms, key.match, key.reassign = NULL, missing = NA)
```

#### S3 method for class 'list'

```r
lookup_e(terms, key.match, key.reassign = NULL, missing = NA)
```

#### S3 method for class 'numeric'

```r
lookup_e(terms, key.match, key.reassign = NULL, missing = NA)
```

#### S3 method for class 'factor'

```r
lookup_e(terms, key.match, key.reassign = NULL, missing = NA)
```

#### S3 method for class 'character'

```r
lookup_e(terms, key.match, key.reassign = NULL, missing = NA)
```

```r
terms %le% key.match
```

```r
terms %le+% key.match
```

### Arguments

- **terms**: A vector of terms to undergo a `lookup_e`.
- **key.match**: Takes one of the following: (1) a two column data.frame of a match key and reassignment column, (2) a named list of vectors (Note: if data.frame or named list supplied no key reassign needed) or (3) a single vector match key.
- **key.reassign**: A single reassignment vector supplied if key.match is not a two column data.frame/named list.
- **missing**: Value to assign to terms not matching the key.match. If set to `NULL` the original values in terms corresponding to the missing elements are retained.

### Value

Outputs a new vector with reassigned values.

### See Also

- `new.env`, `lookup`

### Examples

```r
lookup_e(1:5, data.frame(1:4, 11:14))
```
## Retain original values for missing
lookup_e(1:5, data.frame(1:4, 11:14), missing=NULL)

lookup_e(LETTERS[1:5], data.frame(LETTERS[1:5], 100:104))
lookup_e(LETTERS[1:5], factor(LETTERS[1:5]), 100:104)

## Supply a named list of vectors to key.match

codes <- list(
    A = c(1, 2, 4),
    B = c(3, 5),
    C = 7,
    D = c(6, 8:10)
  )

lookup_e(1:10, codes)

## Supply a single vector to key.match and key.reassign

lookup_e(mtcars$carb, sort(unique(mtcars$carb)),
    c("one", "two", "three", "four", "six", "eight"))
lookup_e(mtcars$carb, sort(unique(mtcars$carb)),
    seq(10, 60, by=10))

## %le%, a binary operator version of lookup
1:5 %le% data.frame(1:4, 11:14)
1:10 %le% codes

1:12 %le% codes
1:12 %le+% codes

---

**mtabulate**  
Tabulate Frequency Counts for Multiple Vectors

### Description
Similar to `tabulate` that works on multiple vectors.

### Usage

mtabulate(vects)

### Arguments

vects  
A vector, list, or data.frame of named/unnamed vectors.
pad

Value

Returns a data.frame with columns equal to number of unique elements and the number of rows equal to the the original length of the vector, list, or data.frame (length equals ncols in data.frame). If list of vectors is named these will be the rownames of the dataframe.

Author(s)

Joran Elias and Tyler Rinker <tyler.rinker@gmail.com>.

References

https://stackoverflow.com/a/9961324/1000343

See Also
tabulate, counts2list

Examples

mtabulate(list(w=letters[1:10], x=letters[1:5], z=letters))
mtabulate(list(mtcars$cyl[1:10]))

# Dummy coding
mtabulate(mtcars$cyl[1:10])
mtabulate(CO2[, "Plant"])

dat <- data.frame(matrix(sample(c("A", "B"), 30, TRUE), ncol=3))
mtabulate(dat)
t(mtabulate(dat))
counts2list(mtabulate(dat))

---

pad

Pad Strings

Description

A convenience wrapper for sprintf that pads the front end of strings with spaces or 0s. Useful for creating multiple uniform directories that will maintain correct order.

Usage

pad(x, padding = max(nchar(as.character(x))), sort = TRUE, type = "detect")
Arguments

x A character, factor, numeric vector.

padding Number of characters to pad. Default makes all elements of a string the number of characters of the element with the maximum characters.

sort logical. If TRUE the outcome is sorted.

type A character string of "detect", "numeric", "character", "d" or "s". If numeric zeros are padded. If character spaces are padded. The detect attempts to determine if x is numeric (d) or not (s).

Value

Returns a character vector every element padded with 0/spaces.

Note

pad is a wrapper for the sprintf function. pad may behave differently on various platforms in accordance with the documentation for sprintf: "actual implementation will follow the C99 standard and fine details (especially the behaviour under user error) may depend on the platform." See sprintf for more information.

See Also

sprintf

Examples

pad(sample(1:10, 10))
pad(sample(1:10, 10), sort=FALSE)
pad(as.character(sample(1:10, 10)))
pad(as.character(sample(1:10, 10)), sort=FALSE)
pad(as.character(sample(1:10, 10)), 4)
pad(month.name)

print.v_outer

Description

Prints a v_outer Object.

Usage

## S3 method for class 'v_outer'
print(x, digits = 3, ...)
Arguments

- The `v_outer` object
- `digits` Number of decimal places to print.
- ... ignored

---

**read_docx**  
*Read in .docx Content*

### Description
Read in the content from a .docx file.

### Usage
```r
read_docx(file, skip = 0)
```

### Arguments
- `file` The path to the .docx file.
- `skip` The number of lines to skip.

### Value
Returns a character vector.

### Author(s)
Bryan Goodrich and Tyler Rinker <tyler.rinker@gmail.com>.

### Examples
```r
# Not run:
# Mining Citation
dl <- dl("http://umlreading.weebly.com/uploads/2/5/2/5/25253346/whole_language_timeline-updated.docx")

(txt <- read_docx("whole_language_timeline-updated.docx"))

library(qdapTools); library(ggplot2); library(qdap)
txt <- rm_non_ascii(txt)

parts <- split_vector(txt, split = "References", include = TRUE, regex=TRUE)

parts[[1]]

rm_citation(unbag(parts[[1]]), extract=TRUE)[[1]]

# By line
rm_citation(parts[[1]], extract=TRUE)
```
## Frequency

```r
left_just(cites <- list2df(sort(table(rm_citation(unbag(parts[[1]])),
    extract=TRUE)), T), "freq", "citation")[[2:1]]
```

## Distribution of citations (find locations and then plot)
```r
cite_locs <- do.call(rbind, lapply(cites[[1]], function(x){
    m <- gregexpr(x, unbag(parts[[1]]), fixed=TRUE)
    data.frame(
        citation=x,
        start = m[[1]] -5,
        end = m[[1]] + 5 + attributes(m[[1]])["match.length"]
    )
}))
```

```r
ggplot(cite_locs) +
    geom_segment(aes(x=start, xend=end, y=citation, yend=citation), size=3,
        color="yellow") +
    xlab("Duration") +
    scale_x_continuous(expand = c(0,0),
        limits = c(0, nchar(unbag(parts[[1]])) + 25)) +
    theme_grey() +
    theme(
        panel.grid.major=element_line(color="grey20"),
        panel.grid.minor=element_line(color="grey20"),
        plot.background = element_rect(fill="black"),
        panel.background = element_rect(fill="black"),
        panel.border = element_rect(colour = "grey50", fill=NA, size=1),
        axis.text=element_text(color="grey50"),
        axis.title=element_text(color="grey50")
    )
```

## End(Not run)

---

repo2github

**Upload a Local Repo to GitHub**

### Description

 Allows uploading a local repository to GitHub without first creating the repository in the clouds. 
**repo2github** is designed for the initial push to GitHub. Future pushes can be handled via RStudio or other Git interface.

### Usage

```r
repo2github(
    password,
    project.dir = getwd(),
    repo = basename(getwd()),
    github.user = getOption("github.user"),
```
repo2github


gitpath = NULL,
readme = TRUE

Arguments

password  GitHub user password (character string). If this is not supplied the user will be prompted to enter a password.
project.dir  The path to the root directory of the report/presentation.
repo  A character string naming the repo; default attempts to use the report project directory name.
github.user  GitHub user name (character string).
gitpath  Path to the location of Git. If NULL repo2github will attempt to locate the path if necessary.
readme  logical. If TRUE repo initializes with a README.md file.

Details

The arguments project.dir and repo use getwd. This assumes is the current working directory is the root directory and is done for convenience. The user should ensure that either their working directory is the root directory or supply the correct root directory/name to these arguments.

Value

Creates GitHub repository.

Warning

For Windows users this function creates a temporary _netrc file in the home directory and attempts to delete this file. The _netrc contains username and password information for GitHub. repo2github attempts to delete this file but care should be taken. The file is created in:

file.path(Sys.getenv()"HOME"), "DELETE_ME_REPORTS_PACKAGE/_nectrc")

Suggestion

The user may want to set options for github.user in the user's primary .Rprofile.

Note

The user will need to have a GitHub account established.

Author(s)

Simon O’Hanlon, Daniel Chaffiol, and Tyler Rinker 

References

https://stackoverflow.com/a/15047013/1000343
https://stackoverflow.com/a/18692400/1000343
Examples

## Not run:
repo2github()

## End(Not run)

---

run_split  Split a String Into Run Chunks

Description

Splits a string into a vector of runs.

Usage

run_split(x)

Arguments

x  
A string.

Value

Returns a list of vectors.

Author(s)

Robert Reed and Tyler Rinker <tyler.rinker@gmail.com>.

References

https://stackoverflow.com/a/24319217/1000343

See Also

loc_split, split_vector

Examples

run_split(c("1233444455555666666", NA, "abcccdddddeeeeffff"))
shift  

*Shift Vector Left/Right*

**Description**

Shift a vector left or right n spaces.

**Usage**

```r
shift(x, n, direction = "right")
shift_right(x, n)
shift_left(x, n)
```

**Arguments**

- `x`: A vector.
- `n`: The number of moves left or right to shift.
- `direction`: A direction to shift; must be either "left" or "right". Use explicit directional shift functions `shift_right` and `shift_left` for better performance.

**Value**

Returns a shifted vector.

**Examples**

```r
lapply(0:9, function(i) shift(1:10, i))
lapply(0:9, function(i) shift(1:10, i, "left"))

## Explicit, faster shifting
lapply(0:9, function(i) shift_right(1:10, i))
lapply(0:9, function(i) shift_left(1:10, i))
lapply(0:25, function(i) shift_left(LETTERS, i))
```

---

split_vector  

*Split a Vector By Split Points*

**Description**

Splits a vector into a list of vectors based on split points.

**Usage**

```r
split_vector(x, split = "", include = FALSE, regex = FALSE, ...)
```
Arguments

x  A vector with split points.
split  A vector of places (elements) to split on or a regular expression if regex argument is TRUE.
include  An integer of 1 (split character(s) are not included in the output), 2 (split character(s) are included at the beginning of the output), or 3 (split character(s) are included at the end of the output).
regex  logical. If TRUE regular expressions will be enabled for split argument.
...  other arguments passed to grep and grepl.

Value

Returns a list of vectors.

Author(s)

Matthew Flickinger and Tyler Rinker <tyler.rinker@gmail.com>.

References

https://stackoverflow.com/a/24319217/1000343

See Also

loc_split, run_split

Examples

set.seed(15)
x <- sample(c(“”, LETTERS[1:10]), 25, TRUE, prob=c(.2, rep(.08, 10)))

split_vector(x)
split_vector(x, “C”)
split_vector(x, c(“”, “C”))

split_vector(x, include = 0)
split_vector(x, include = 1)
split_vector(x, include = 2)

set.seed(15)
x <- sample(1:11, 25, TRUE, prob=c(.2, rep(.08, 10)))
split_vector(x, 1)

## relationship to ‘loc_split’
all.equal(
  split_vector(x, include = 1),
  loc_split(x, which(x == “”), names=1:6)
)
start_end

Get Location of Start/End Points

Description

Get the locations of start/end places for the ones in a binary vector.

Usage

start_end(x)

Arguments

x

A vector of 1 and 0 or logical.

Value

Returns a two column data.frame of start and end locations for ones.

Author(s)

Roland (https://stackoverflow.com/users/1412059/roland) and Tyler Rinker <tyler.rinker@gmail.com>.

References

https://stackoverflow.com/a/29184841/1000343

Examples

set.seed(10); (x <- sample(0:1, 50, TRUE, c(.35, .65)))
start_end(x)
(y <- sample(c(TRUE, FALSE), 50, TRUE, c(.35, .65)))
start_end(y)

text2color

Map Words to Colors

Description

A dictionary lookup that maps words to colors.

Usage

text2color(words, recode.words, colors)
Arguments

words A vector of words.
recode.words A vector of unique words or a list of unique word vectors that will be matched against corresponding colors.
colors A vector of colors of equal in length to recode.words +1 (the +1 is for unmatched words).

Value

Returns a vector of mapped colors equal in length to the words vector.

See Also

lookup

Examples

x <- structure(list(X1 = structure(c(3L, 1L, 8L, 4L, 7L, 2L, 2L, 2L, 4L, 8L, 4L, 3L, 5L, 3L, 1L, 8L, 7L, 2L, 1L, 6L), .Label = c("a", "and", "in", "is", "of", "that", "the", "to"), class = "factor"), .Names = "X1", row.names = c(NA, -20L), class = "data.frame")
#blue was recycled
text2color(x$X1, c("the", "and", "is"), c("red", "green", "blue"))
text2color(x$X1, c("the", "and", "is"), c("red", "green", "blue", "white"))
x$X2 <- text2color(x$X1, list(c("the", "and", "is"), "that"), c("red", "green", "white"))
x

url_dl

Download Instructional Documents

Description

This function enables downloading documents for future instructional training.

Usage

url_dl(..., url = 61803503)

Arguments

url The download url or Dropbox key.
... Document names to download. Quoted strings (complete urls) can also be supplied (if so no url argument is supplied).
Value

Places a copy of the downloaded document in the users working directory.

Examples

```r
## Not run:
## Example 1 (download from Dropbox)
# download transcript of the debate to working directory
library(qdap)
url_dl(pres.deb1.docx, pres.deb2.docx, pres.deb3.docx)

# load multiple files with read transcript and assign to working directory
dat1 <- read.transcript("pres.deb1.docx", c("person", "dialogue"))
dat2 <- read.transcript("pres.deb2.docx", c("person", "dialogue"))
dat3 <- read.transcript("pres.deb3.docx", c("person", "dialogue"))

docs <- qcv(pres.deb1.docx, pres.deb2.docx, pres.deb3.docx)
dir() %in% docs
library(reports); delete(docs)  #remove the documents
dir() %in% docs

## Example 2 (quoted string urls)
   "http://www.cran.r-project.org/doc/manuals/R-intro.pdf")
dir() %in% docs

## End(Not run)
```

---

v_outer  
Vectorized Version of outer

Description

Vectorized outer.

Usage

```r
v_outer(x, FUN, ...)
```

```r
## S3 method for class 'list'
v_outer(x, FUN, ...)
```

```r
## S3 method for class 'data.frame'
v_outer(x, FUN, ...)
```

```r
## S3 method for class 'matrix'
v_outer(x, FUN, ...)
```
Arguments

x
A matrix, dataframe or equal length list of vectors.

FUN
A vectorized function.

... Other arguments passed to the function supplied to FUN.

Value

Returns a matrix with the vectorized outer function.

Author(s)

Vincent Zoonekynd, eddi of stackoverflow.com, and Tyler Rinker <tyler.rinker@gmail.com>.

References

https://stackoverflow.com/a/9917425/1000343
https://stackoverflow.com/q/23817341/1000343

See Also

outer, cor

Examples

```r
#|------------------------------------------------------|
#| SETTING UP VARIOUS FUNCTIONS THAT WILL BE USED |
#|------------------------------------------------------|
pooled_sd <- function(x, y) {
  n1 <- length(x)
  n2 <- length(y)
  s1 <- sd(x)
  s2 <- sd(y)
  sqrt(((n1-1)*s1 + (n2-1)*s2)/((n1-1) + (n2-1)))
}

## Effect Size: Cohen's d
cohens_d <- function(x, y) {
  (mean(y) - mean(x))/pooled_sd(x, y)
}

## Euclidean Distance
euc_dist <- function(x,y) sqrt(sum((x - y) ^ 2))

## Cosine similarity
cos_sim <- function(x, y) x %*% y / sqrt(x%*%x * y%*%y)

sum2 <- function(x, y) sum(x, y)
arbitrary <- function(x, y) round(sqrt(sum(x)) - sum(y), digits=1)
```
## A data.frame

```r
v_outer(mtcars, cor)
v_outer(mtcars, pooled_sd)
v_outer(mtcars[, 1:7], euc_dist)
v_outer(mtcars[, 1:7], sum2)
v_outer(mtcars[, 1:7], arbitrary)
```

## mtcars as a list

```r
mtcars2 <- lapply(mtcars[, 1:7], "[")
v_outer(mtcars2, cor)
v_outer(mtcars2, method = "spearman")
v_outer(mtcars2, pooled_sd)
v_outer(split(mtcars["mpg"], mtcars["carb"]), cohens_d)
v_outer(split(CO2["uptake"], CO2["Plant"]), cohens_d)
print(v_outer(mtcars[, 1:7], pooled_sd), digits = 1)
print(v_outer(mtcars[, 1:7], pooled_sd), digits = NULL)
v_outer(mtcars2, euc_dist)
v_outer(mtcars2, sum2)
v_outer(mtcars2, arbitrary)
```

## A matrix

```r
mat <- matrix(rbinom(500, 0:1, .45), ncol=10)
v_outer(mat, cos_sim)
v_outer(mat, euc_dist)
v_outer(mat, arbitrary)
```

## Not run:

```r
library(qdap)
wc3 <- function(x, y) sum(sapply(list(x, y), wc, byrow = FALSE))
L1 <- word_list(DATA$state, DATA$person)$cwl
(x <- v_outer(L1, wc3))
diag(x) <- (sapply(L1, length))
x
```

```r
v_outer(with(DATA, wfm(state, person)), cos_sim)
with(DATA, Dissimilarity(state, person))
```

## End(Not run)

---

### Hash/Dictionary Lookup

**Description**

- A deprecated binary operator version of `lookup`. This will be removed in a subsequent version of `qdapTools`. Use `%l%` instead.
- A deprecated binary operator version of `hash_look`. This will be removed in a subsequent version of `qdapTools`. Use `%hl%` instead.
Usage

```r
terms %l*% key.match
terms %ha% key
```

Arguments

- `terms` A vector of terms to undergo a lookup.
- `key.match` Takes one of the following: (1) a two column data.frame of a match key and reassignment column, (2) a named list of vectors (Note: if data.frame or named list supplied no key reassign needed) or (3) a single vector match key.
- `key` The hash key to use.
Index

* begin
  start_end, 23
* collapse
  list2df, 5
* docx
  read_docx, 17
* end
  start_end, 23
* frequency
  mtabulate, 14
* id
  id, 4
* list
  list2df, 5
* shift
  shift, 21
* start
  start_end, 23
* stop
  start_end, 23
* tabulate
  mtabulate, 14
  %ha% (%l*%), 27
  %hl%+ (hash), 2
  %hl% (hash), 2
  %l% (lookup), 10
  %l% (lookup), 10
  %lc% (lookup), 10
  %l% (lookup), 10
  %lle% (lookup_e), 12
  %le% (lookup_e), 12
  %l% (hash), 27

cor, 26
counts2list, 15
counts2list (list2df), 5
data.frame, 14, 15, 23
df2matrix (list2df), 5

environment, 3
getwd, 19
grep, 22
grep1, 22
hash, 2, 3, 11
hash_e (hash), 2
hash_look (hash), 2
hms2sec, 4
id, 4
length, 4
list, 6, 14, 15
list2df, 5
list_df2df (list2df), 5
list_vect2df (list2df), 5
loc_split, 8, 20, 22
logical, 23
lookup, 10, 13, 24
lookup_e, 12

matrix2df (list2df), 5
matrix2long (list2df), 5
mtabulate, 7, 14
ncol, 4
new.env, 13
options, 19
outer, 25, 26
pad, 5, 15
print.v_outer, 16

read_docx, 17
repo2github, 18, 18, 19
run_split, 9, 20, 22
sec2hms (hms2sec), 4
setDT, 3, 11
shift, 21
shift_left (shift), 21
shift_right (shift), 21
split_vector, 9, 20, 21
sprintf, 15, 16
start_end, 23

tabulate, 14, 15
text2color, 23
times, 4

url_dl, 24

v_outer, 25
vect2df (list2df), 5
vect2list (list2df), 5
vector, 14, 15