Package 'common'

July 18, 2022

Type Package

Title Solutions for Common Problems in Base R

Version 1.0.1

Maintainer David Bosak <dbosak01@gmail.com>

Description Contains functions for solving commonly encountered problems while programming in R. This package is intended to provide a lightweight supplement to Base R, and will be useful for almost any R user.

License CC0

Encoding UTF-8

Depends R (>= 3.6.0)

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Imports this.path

Config/testthat/edition 3

RoxygenNote 7.2.0

VignetteBuilder knitr

NeedsCompilation no

Author David Bosak [aut, cre],
Andrew Simmons [aut],
Duong Tran [ctb]

Repository CRAN

Date/Publication 2022-07-18 08:10:11 UTC

R topics documented:

`labels.data.frame` *Get or set labels for a data frame*

Description

The `labels` function extracts all assigned labels from a data frame, and returns them in a named list. The function also assigns labels from a named list. This function is a data frame-specific implementation of the Base R `labels` function.

Usage

```
## S3 method for class 'data.frame'
labels(object, ...)

labels(x) <- value
```

Arguments

`object`	A data frame or tibble.
`...`	Follow-on parameters. Required for generic function.
`x`	A data frame or tibble
`value`	A named list of labels. The labels must be quoted strings.

Details

If labels are assigned to the "label" attributes of the data frame columns, the `labels` function will extract those labels. The function will return the labels in a named list, where the names correspond to the name of the column that the label was assigned to. If a column does not have a label attribute assigned, that column will not be included in the list.

When used on the receiving side of an assignment, the function will assign labels to a data frame. The labels should be in a named list, where each name corresponds to the data frame column to assign the label to.

Finally, if you wish to clear out the label attributes, assign a NULL value to the `labels` function.

Value

A named list of labels. The labels must be quoted strings.

Examples

```
# Take subset of data
df1 <- mtcars[1:10, c("mpg", "cyl") ]

# Assign labels
labels(df1) <- list(mpg = "Mile Per Gallon", cyl = "Cylinders")
```

```
# Examine attributes
str(df1)
# 'data.frame': 10 obs. of  2 variables:
# $ mpg: num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2
# ..- attr(*, "label")= chr "Mile Per Gallon"
# $ cyl: num  6 6 4 6 8 6 8 4 4 6
# ..- attr(*, "label")= chr "Cylinders"

# View assigned labels
labels(df1)
# $mpg
# [1] "Mile Per Gallon"
#
# $cyl
# [1] "Cylinders"

# Clear labels
labels(df1) <- NULL

# Display Cleared Labels
labels(df1)
# list()
```

`roundup` *Rounds numbers up*

Description

A function that rounds positive numbers up when the last digit is a 5. For negative numbers ending in 5, the function actually rounds down. "Round away from zero" is the most accurate description of this function.

Usage

```
roundup(x, digits = 0)
```

Arguments

`x`	A vector of values to round.
`digits`	A number of decimal places to round to. Default is zero.

Value

The rounded data vector.

Examples

```
# Round to even
round(2.4)   # 2
round(2.5)   # 2
round(-2.5)  # -2
round(2.6)   # 3

# Round up
roundup(2.4)  # 2
roundup(2.5)  # 3
roundup(-2.5) # -3
roundup(2.6)  # 3
```

`sort.data.frame` *Sorts a data frame*

Description

An overload to the Base R `sort` function for data frames. Allows multiple columns to be sorted easily. Also allows you to control the sort direction for each column independently.

Usage

```
## S3 method for class 'data.frame'
sort(
  x,
  decreasing = FALSE,
  ...,
  by = NULL,
  ascending = TRUE,
  na.last = TRUE,
  index.return = FALSE
)
```

Arguments

`x`	The input data frame to sort.
`decreasing`	This parameter was added to conform to the S3 generic method signature of the `sort` function, and will be ignored here. Please use the `ascending` parameter.
`...`	This parameter is required for the generic method signature. Anything passed on it will be ignored.
`by`	A vector of column names to sort by. If this parameter is not supplied, the function will sort by all columns in order from left to right.

ascending	A vector of TRUE or FALSE values corresponding to the variables on the by parameter. These values will determine the direction to sort each column. Ascending is TRUE, and descending is FALSE. The vector will be recycled if it is short, and truncated if it is long. By default, all variables will be sorted ascending.
na.last	Whether to put NA values first or last in the sort. If TRUE, NA values will sort to the bottom. If FALSE, NA values will sort to the top. The default is TRUE.
index.return	Whether to return the sorted data frame or a vector of sorted index values. If this parameter is TRUE, the function will return sorted index values. By default, the parameter is FALSE, and will return the sorted data frame.

Value

The function returns either a sorted data frame or a sorted vector of row index values, depending on the value of the `index.return` parameter. If `index.return` is FALSE, the function will return the sorted data frame. If the `index.return` parameter is TRUE, it will return a vector of row indices.

Examples

```
# Prepare unsorted sample data
dt <- mtcars[1:10, 1:3]
dt
#                    mpg cyl  disp
# Mazda RX4         21.0   6 160.0
# Mazda RX4 Wag     21.0   6 160.0
# Datsun 710        22.8   4 108.0
# Hornet 4 Drive    21.4   6 258.0
# Hornet Sportabout 18.7   8 360.0
# Valiant           18.1   6 225.0
# Duster 360        14.3   8 360.0
# Merc 240D         24.4   4 146.7
# Merc 230          22.8   4 140.8
# Merc 280          19.2   6 167.6

# Sort by mpg ascending
dt1 <- sort(dt, by = "mpg")
dt1
#                    mpg cyl  disp
# Duster 360        14.3   8 360.0
# Valiant           18.1   6 225.0
# Hornet Sportabout 18.7   8 360.0
# Merc 280          19.2   6 167.6
# Mazda RX4         21.0   6 160.0
# Mazda RX4 Wag     21.0   6 160.0
# Hornet 4 Drive    21.4   6 258.0
# Datsun 710        22.8   4 108.0
# Merc 230          22.8   4 140.8
# Merc 240D         24.4   4 146.7

# Sort by mpg descending
dt1 <- sort(dt, by = "mpg", ascending = FALSE)
```

```
dt1
#                    mpg cyl  disp
# Merc 240D         24.4   4 146.7
# Datsun 710        22.8   4 108.0
# Merc 230          22.8   4 140.8
# Hornet 4 Drive    21.4   6 258.0
# Mazda RX4         21.0   6 160.0
# Mazda RX4 Wag     21.0   6 160.0
# Merc 280          19.2   6 167.6
# Hornet Sportabout 18.7   8 360.0
# Valiant           18.1   6 225.0
# Duster 360        14.3   8 360.0

# Sort by cyl then mpg
dt1 <- sort(dt, by = c("cyl", "mpg"))
dt1
#                    mpg cyl  disp
# Datsun 710        22.8   4 108.0
# Merc 230          22.8   4 140.8
# Merc 240D         24.4   4 146.7
# Valiant           18.1   6 225.0
# Merc 280          19.2   6 167.6
# Mazda RX4         21.0   6 160.0
# Mazda RX4 Wag     21.0   6 160.0
# Hornet 4 Drive    21.4   6 258.0
# Duster 360        14.3   8 360.0
# Hornet Sportabout 18.7   8 360.0

# Sort by cyl descending then mpg ascending
dt1 <- sort(dt, by = c("cyl", "mpg"),
            ascending = c(FALSE, TRUE))
dt1
#                    mpg cyl  disp
# Duster 360        14.3   8 360.0
# Hornet Sportabout 18.7   8 360.0
# Valiant           18.1   6 225.0
# Merc 280          19.2   6 167.6
# Mazda RX4         21.0   6 160.0
# Mazda RX4 Wag     21.0   6 160.0
# Hornet 4 Drive    21.4   6 258.0
# Datsun 710        22.8   4 108.0
# Merc 230          22.8   4 140.8
# Merc 240D         24.4   4 146.7
```

`Sys.path` *Returns the path of the current program*

Description

A function that gets the full path of the currently running program. If the function fails to retrieve the path for some reason, it will return a NULL. The function takes no parameters.

Credit for this function goes to Andrew Simmons and the this.path package.

Usage

```
Sys.path()
```

Value

The full path of the currently running program, or a NULL.

Examples

```
# Get current path
pth <- Sys.path()
pth
# [1] "C:/programs/myprogram.R"
```

---

v *Combine unquoted values*

Description

A function to quote and combine unquoted values. The function will return a vector of quoted values. This function allows you to use non-standard evaluation for any parameter that accepts a string or vector of strings.

Usage

```
v(...)
```

Arguments

...	One or more unquoted values.

Value

A vector of quoted values.

Examples

```
# Combine unquoted values
v(var1, var2, var3)
# [1] "var1" "var2" "var3"

# Data frame subset
dat <- mtcars[1:5, v(mpg, cyl, disp)]
dat
#                   mpg cyl disp
# Mazda RX4         21.0   6  160
```

```
# Mazda RX4 Wag     21.0   6  160
# Datsun 710        22.8   4  108
# Hornet 4 Drive    21.4   6  258
# Hornet Sportabout 18.7   8  360

# Data frame sort
dat2 <- sort(dat, by = v(cyl, mpg))
dat2
#                    mpg cyl disp
# Datsun 710        22.8   4  108
# Mazda RX4         21.0   6  160
# Mazda RX4 Wag     21.0   6  160
# Hornet 4 Drive    21.4   6  258
# Hornet Sportabout 18.7   8  360
```

`%eq%` *Check equality of two objects*

Description

The goal of the `%eq%` operator is to return a TRUE or FALSE value when any two objects are compared. The function provides a simple, reliable equality check that allows comparing of NULLs, NA values, and atomic data types without error.

The function also allows comparing of data frames. It will return TRUE if all values in the data frames are equal, and ignores differences in attributes.

Usage

```
x1 %eq% x2
```

Arguments

`x1`	The first object to compare
`x2`	The second object to compare

Value

A single TRUE or FALSE value depending on whether the objects are equal.

Examples

```
# Comparing of NULLs and NA
NULL %eq% NULL        # TRUE
NULL %eq% NA          # FALSE
NA %eq% NA            # TRUE
1 %eq% NULL           # FALSE
1 %eq% NA             # FALSE

# Comparing of atomic values
```

```
1 %eq% 1              # TRUE
"one" %eq% "one"      # TRUE
1 %eq% "one"          # FALSE
1 %eq% Sys.Date()     # FALSE

# Comparing of vectors
v1 <- c("A", "B", "C")
v2 <- c("A", "B", "D")
v1 %eq% v1            # TRUE
v1 %eq% v2            # FALSE

# Comparing of data frames
mtcars %eq% mtcars    # TRUE
mtcars %eq% iris      # FALSE
iris %eq% iris[1:50,] # FALSE

# Mixing it up
mtcars %eq% NULL      # FALSE
v1 %eq% NA            # FALSE
1 %eq% v1             # FALSE
```

`%p%` *An infix operator for* `paste0()`

Description

This function provides an infix operator for the `paste0` function to concatenate strings. The operator will concatenate a vector of one or more values. The functionality is identical to `paste0()`, but more convenient to use in some situations.

Usage

```
x %p% y
```

Arguments

`x`	A value for the left side of the paste infix operator.
`y`	A value for the right side of the paste infix operator.

Value

The concatenated or pasted value. No spaces will be inserted in between the values to paste. If a vector of values is supplied, a vector of pasted values will be returned.

Examples

```
# Paste together two strings
str <- "Hello" %p% "World"
str
# [1] "HelloWorld"

# Paste together number and string
str <- 100 %p% " Kittens"
str
# [1] "100 Kittens"

# Paste together two vectors
v1 <- c("A", "B", "C")
v2 <- c(1, 2, 3)
str <- v1 %p% v2
str
# [1] "A1" "B2" "C3"
```

Index