Package 'isthisthat'

July 17, 2020

Type Package

Title Representation and equality of strings and numbers

Version 1.0.0

Description The package check the representation and equality of strings and numbers. All functions determine if an object this is of the form or equal to that and return a logical object of the same dimension.

License GPL-3

Copyright Research and development institute for agri-environment (IRDA)

URL https://github.com/irda-rd/isthisthat, https://www.irda.qc.ca/en/

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Imports methods, stringr

R topics documented:

`isthisthat-package` *isthisthat: Representation and equality of strings and numbers*

Description

The package check the representation and equality of strings and numbers. All functions determine if an object *this* is of the form or equal to *that* and return a `logical` object of the same dimension.

More precisely, a group of functions (`isStringAnInteger`, `isStringADecimal`, `isStringASciNumber`, `isStringANumber`) verify if an object of class `character` correspond to certain types of number, while `isStringADate` and `isStringCanBeADate` concern its representation of a date.

Similarly, the functions `isNumericADecimal` and `isNumericAnInteger` allow verifications to be made on objects of class `numeric`.

Finally, the functions `isEqual` and `isAlmostEqual` allow to compare objects with `NA` and according to some tolerance.

The name is a reference to *This Is That*, the news satire program broadcasted on CBC Radio.

Author(s)

Maintainer: Alexandre Leblanc <`alexandre.leblanc@irda.qc.ca`>

Other contributors:

- Christine Landry <`christine.landry@irda.qc.ca`> [research team head]
- Research and development institute for agri-environment (IRDA) [copyright holder]

See Also

Useful links:

- https://github.com/irda-rd/isthisthat
- https://www.irda.qc.ca/en/

`isAlmostEqual` *Approximative equality*

Description

Check approximative equality for the `numeric` class, including `NA` presence.

Usage

```
isAlmostEqual(x, y, ...)
```

Arguments

x	object 1 to compare (vector, matrix, data.frame).
y	object 2 to compare (vector, matrix, data.frame).
tol	tolerance used when considering numeric values equal.

Details

The length or dimension of x and y must be the same. The function allows to determine if the difference between two numeric is smaller than tol. It also allows to compare a numeric with character representing a number. The latter, if corresponding to a number, are converted to numeric then rounded at 15 significative numbers (the recommanded limit for doubles in the IEEE-754 standard). Comparison is made as character to avoid coercion of the initial character vector. Scientific notation is also handled. The approach allows to avoid truncating problems associated with as.character for number with several decimals, precision problems with numeric as well as notation problems. The function refer to isEqual for any other case.

Value

If x and y are vectors, a logical vector of the same length is returned; if x and y are of class matrix or data.frame, a logical matrix of the same dimension is returned.

Examples

```
#Comparison between isEqual and isAlmostEqual
##For two numeric vectors
##(4th case is not equal, but is almost equal given the tol)
x <- c(NA, 1, 1.1, 1.000000001, 123456789.123456789, 1000000)
y <- c(NA, 1, 1.1, 1.000000002, 123456789.123456789, 1000000)
isEqual(x, y)
isAlmostEqual(x, y)

#For a numeric and a character vector
##(5th case have more than 15 significative numbers, 6th case is a change in notation)
xChar <- c(NA, "1", "1.1", "1.000000001", "123456789.123456789", "1000000")
isEqual(x, xChar)
isAlmostEqual(x, xChar)

#With two data frames
df1 <- data.frame(x1 = x, x2 = x, x3 = letters[1:6], stringsAsFactors = FALSE)
df2 <- data.frame(x1 = y, x2 = xChar, x3 = letters[1:6], stringsAsFactors = FALSE)
isEqual(df1, df2)
isAlmostEqual(df1, df2)

#With two matrices
M1 <- matrix(x,3,2)
M2 <- matrix(xChar,3,2)
isEqual(M1, M2)
isAlmostEqual(M1, M2)

#With scientific notation
isEqual("1.234E-1",0.1234)
isAlmostEqual("1.234E-1",0.1234)
```

isEqual *Verify equality*

Description

Verify equality, including NA presence.

Usage

```
isEqual(x, y)
```

Arguments

x	object 1 to compare (vector, matrix, data.frame).
y	object 2 to compare (vector, matrix, data.frame).

Details

The length or dimension of x and y must be the same. The function rely on the == operator; if x or y are numeric with several significative numbers, one should use the function isAlmostEqual instead.

Value

If x and y are vectors, a logical vector of the same length is returned; if x and y are of class matrix or data.frame, a logical matrix of the same dimension is returned.

Examples

```
x <- c(1:3,NA)
y <- c("1","C",NA, NA)
isEqual(x, y)
```

isNumericADecimal *Identify if numbers are decimal numbers*

Description

Identify if elements of a numeric vector are decimal numbers.

Usage

```
isNumericADecimal(x, tol = .Machine$double.eps^0.5)
```

Arguments

x	vector of class numeric.
tol	tolerance used when considering numeric values equal.

Details

The function also return `FALSE` for `NA`. The code is in part taken from the example of the function `integer`.

Value

Return a `logical` vector indicating for each element if it correspond to a decimal number.

Examples

```
x <- c(1, 1.1, 0.1, 1.0, 1.00000001, NA)
isNumericADecimal(x)
```

`isNumericAnInteger` *Identify if numbers are integers*

Description

Identify if elements of a `numeric` vector are integers.

Usage

```
isNumericAnInteger(x, tol = .Machine$double.eps^0.5)
```

Arguments

`x`	vector of class `numeric`.
`tol`	tolerance used when considering numeric values equal.

Details

The function also return `FALSE` for `NA`. Code in part taken from the example of the function `integer`.

Value

Return a `logical` vector indicating for each element if it correspond to an integer.

Examples

```
x <- c(1, 1.1, 0.1, 1.0, 1.00000001, NA)
isNumericAnInteger(x)
```

isStringADate *Identify valid dates*

Description

Identify if elements of a character vector represent valid dates in the designated format.

Usage

```
isStringADate(x, format = "%Y-%m-%d")
```

Arguments

x	character vector.
format	character representing a date (see as.Date). The only implemented format is "%Y-%m-%d".

Value

Return a logical vector identifying, for each element, if the string correspond to a valid date in the given format.

Examples

```
x <- c("2019-06-20", "2019-06-32", "43636", NA, "2019/06/20")
isStringADate(x)
```

isStringADecimal *Identify if a string is a decimal number*

Description

Check if elements of a character vector are decimal numbers (i.e. separated by a decimal symbol).

Usage

```
isStringADecimal(x, decimalSeparator = ".")
```

Arguments

x	character vector.
decimalSeparator	character used as a decimal separator (”,” or ”.”); decimalSeparator = "." by default.

Details

The function does not identify number in their scientific notation. Use the function isStringSciNumber for this purpose.

Value

Return a `logical` vector indicating, for each element of `x`, if the string correspond to a decimal number.

Examples

```
x <- c("12.34", "12.00", "-1.1", "12,34", "-1.1E-2", "1234", "-1", "1E-2", "123A", "12.00.00", ".1")
isStringADecimal(x,".")
isStringADecimal(x,",")
```

---

isStringAnInteger *Identify if a string is an integer*

Description

Check if elements of a `character` vector are integers.

Usage

```
isStringAnInteger(x)
```

Arguments

`x`	`character` vector.

Details

Use the function `isStringSciNumber` to identify numbers in their scientific notation.

Value

Return a `logical` vector indicating, for each element of `x`, if the string correspond to an integer.

Examples

```
x <- c("1234", "-1","+1", "12.34","12,34", "123A","-1.1","1E+1")
isStringAnInteger(x)
```

isStringANumber *Identify if a string is a number*

Description

Check if elements of a `character` vector are numbers, in their scientific notation or not.

Usage

```
isStringANumber(x, decimalSeparator = ".")
```

Arguments

`x`	`character` vector.
`decimalSeparator`	`character` used as a decimal separator (”,” or ”.”); `decimalSeparator = "."` by default.

Value

Return a `logical` vector indicating, for each element of `x`, if the string correspond to a number.

Examples

```
x <- c("1234", "12.34","1.1E+10", "12,34", "1,1E+10", "123A")
isStringANumber(x,".")
isStringANumber(x,",")
```

isStringCanBeADate *Identify if a string can be converted into a date*

Description

Check if elements of a `character` vector can be converted into a `Date`. The function check both string corresponding to a date in the specified format or a number, for a representation given an origin.

Usage

```
isStringCanBeADate(x, format = ("%Y-%m-%d"), warning = FALSE)
```

Arguments

`x`	`character` vector.
`warning`	`logical` indicating if a warning must be sent if any element cannot be converted (`FALSE` by default).

Details

String representing valid dates, according to the specified format, numbers and `NA` (but not ”NA”) are considered to be convertible into a `Date`.

Value

Return a `logical` vector indicating, for each element of `x`, if the string can be converted into a `Date`.

Examples

```
#Examples of valid cases
x <- c("2019-06-20", 43636, "43636.1", "43636,1", NA)
isStringCanBeADate(x)

#Examples of non-valid cases
y<- c("AA", "2012/05/02", "06-05-2012", "2019-06-32", "NA")
isStringCanBeADate(y)
```

`isStringSciNumber` *Identify if a string is a scientific number*

Description

Check if elements of a `character` vector are numbers expressed in scientific notation.

Usage

```
isStringSciNumber(x, decimalSeparator = ".")
```

Arguments

`x`	`character` vector.
`decimalSeparator`	`character` used as a decimal separator (”,” or ”.”); `decimalSeparator = "."` by default.

Details

The function can identify positive or negative numbers, integer or decimals, followed by `E` (or `e`), another symbol (`+`, `-` or nothing) and digits representing the exponent.

Value

Return a `logical` vector indicating, for each element of `x`, if the string correspond to a number in scientific notation.

Examples

```
x <- c("1234", "-1","+1", "12.34","12,34", "123A","-1.1","E+1")
y <- c("1E+1","-1e-1", "+1.1e-1","1.1E10","1,1e+10")
isStringSciNumber(x)
isStringSciNumber(y)
isStringSciNumber(y, decimalSeparator = ",")
```

Index