Package 'tester'

February 20, 2015

Type Package

Title Tests and checks characteristics of R objects

Version 0.1.7

Date 2013-11-14

Author Gaston Sanchez

Maintainer Gaston Sanchez <gaston.stat@gmail.com>

Description tester allows you to test characteristics of common R objects.

License GPL-3

URL <http://www.gastonsanchez.com>

Depends R (>= 3.0)

Suggests testthat, knitr

VignetteBuilder knitr

Collate 'has-dimension.r' 'has_factors.r' 'has-missing.r'
'has-names.r' 'is-class.r' 'is-dataframe.r' 'is-decimal.r'
'is-integer.r' 'is-matrix.r' 'is-multiple.r' 'is-natural.r'
'is-positive-negative.r' 'is-string.r' 'is-tabular.r'
'is-triangular.r' 'is-vector.r' 'is_square_matrix.r'
'list-of-vectors.r' 'list-with-vectors.r' 'odd-even.r'
'true-false.r' 'same-class.r' 'same-dim.r' 'same-length.r'
'same-mode.r' 'same-type.r' 'is-one-dim.r' 'is-single.r'
'is-scalar.r'

NeedsCompilation no

Repository CRAN

Date/Publication 2013-11-14 09:26:49

R topics documented:

1

Index [37]

---

has_dimension *Has dimension?*

Description

has_dimension and has_dim test if an object has dimension (i.e. dim)
lacks_dimension and lacks_dim test if an object lacks dimension

Usage

```
has_dimension(x)
```

Arguments

x   an R object

Examples

```
m = matrix(1:12, 4, 3)
a = as.array(letters)
has_dim(m) # TRUE
has_dimension(a)

has_dimension(iris) # TRUE

has_dim(matrix(1:10, 10, 1)) # TRUE
has_dim(matrix(1:10, 1, 10)) # TRUE

has_dim(1) # FALSE
lacks_dim(1) # TRUE
has_dim(1:10) # FALSE
has_dimension("dimension") # FALSE
```

---

has_factors *Has factors?*

Description

Whether a data frame or list has factors

Usage

```
has_factors(x)
```

Arguments

x   an R object

Examples

```
has_factors(iris) # TRUE
has_factors(iris[,1:4]) # FALSE
has_factors(list(iris$Species, 1:150)) # TRUE
```

---

has_missing   *Has missing values, NA, NaN, Inf*

Description

has_missing and has_NA tests if there are missing values (NA)
has_infinite and has_Inf tests if there are infinite values (Inf, -Inf)
has_not_a_number and has_NaN tests if there are 'Not a Number' (NaN)
has_nas tests if there are any of the previous ones

Usage

```
has_missing(x)
```

Arguments

x   an R object

Examples

```
has_missing(1:5) # FALSE
has_missing(c(1, 2, 3, 4, NA)) # TRUE

has_infinite(c(1, 2, Inf, 1/0))
has_infinite(c(-Inf, "infinite"))

has_not_a_number(c(1, 2, 3)) # FALSE
has_not_a_number(c(1, 0/0, 3)) # TRUE
has_not_a_number(c(NaN, pi, log(1))) # TRUE
```

---

has_names *Has or lacks names?*

Description

has_names tests if an object has names
lacks_names tests if an object lacks names

Usage

```
has_names(x)
```

Arguments

x   an R object

See Also

has_rownames

Examples

```
set.seed(1)
x <- y <- runif(10)
names(x) = letters[1:10]

has_names(x) # TRUE
has_names(y) # FALSE

lacks_names(x) # FALSE
lacks_names(y) # TRUE
```

---

has_rownames *Has or lacks row/column names?*

Description

has_rownames tests if an object has row names
has_colnames tests if an object has column names
has_dimnames tests if an object has dimnames
lacks_rownames tests if an object lacks row names
lacks_colnames tests if an object lacks column names
lacks_dimnames tests if an object lacks dimnames

Usage

```
has_rownames(x)
```

Arguments

x   an R object

See Also

has_names

Examples

```
has_rownames(iris) # TRUE
has_colnames(iris) # TRUE

lacks_rownames(letters[1:10]) # TRUE
lacks_colnames(letters[1:10]) # TRUE

A = matrix(1:10)
has_dimnames(A) # FALSE
lacks_dimnames(A) # TRUE
```

---

is_class   *Is class*

Description

Tests if an object is of a given class

Usage

```
is_class(x, name = NULL)
```

Arguments

x   an R object
name   string giving the class to be tested

Examples

```
is_class("test_me", "character") # TRUE
is_class(1:10, "numeric") # TRUE

y = 'hello'
class(y) = "hello"
is_class(y, 'hello')
```

---

is_dataframe   *Is data frame*

Description

is_dataframe tests if an object is a data frame

is_numeric_dataframe tests if an object is a numeric data frame

is_string_dataframe tests if an object is a string data frame

is_factor_dataframe tests if an object is a factor data frame

is_not_dataframe tests if an object is not a data frame

Arguments

x   an R object

Examples

```
is_dataframe(iris) # TRUE
is_dataframe(1:10) # FALSE

is_numeric_dataframe(iris) # FALSE
is_numeric_dataframe(iris[,1:4]) # TRUE

DF = matrix(letters[1:24], 6, 4)
DF1 = data.frame(DF)
DF2 = data.frame(DF, stringsAsFactors=FALSE)

is_string_dataframe(DF1) # FALSE
is_string_dataframe(DF2) # TRUE

is_factor_dataframe(DF1) # TRUE
is_factor_dataframe(DF2) # FALSE
```

---

is_decimal   *Is decimal*

Description

Test if is a decimal number

Usage

```
is_decimal(x)
```

Arguments

x   an R object

Details

decimal is any number in the intervals (-1,0) and (0,1)

See Also

is_integer

Examples

```
is_decimal(0.01) # TRUE
is_decimal(-0.01) # TRUE
is_decimal(0) # FALSE
is_decimal(1) # FALSE
is_decimal(runif(5))
is_decimal(rnorm(5))

M = matrix(seq(-2, 2, length.out=10), 5, 2)
is_decimal(M)
```

---

is_diagonal   *Is diagonal matrix*

Description

Test if an object is a diagonal matrix (or not) (i.e. square matrix with zeros above and below the
diagonal)

Usage

```
is_diagonal(x)
```

Arguments

x   an R object

See Also

is_matrix, is_square_matrix

Examples

```
m1 = diag(1:3, 3, 3)
m2 = matrix(1:9, 3, 3)

is_diagonal(m1) # TRUE
is_diagonal(m2) # FALSE
is_not_diagonal(m2) # TRUE
```

---

is_even   *Is even*

Description

Test if an object is an even number
is_not_even tests the opposite condition

Usage

```
is_even(x)
```

Arguments

x   an R object

See Also

[is_odd](#)

Examples

```
is_even(2)
is_even(1)
is_even(seq(-5, 5))

is_even(iris$Species)
is_even(iris)
is_even(list(1, 0, -1, iris))

set.seed(999)
M = matrix(1:12, 4, 3)
is_even(M)
```

---

is_integer   *Is integer*

Description

Test if a number is an integer
Use is_not_integer to test the opposite condition

Usage

```
is_integer(x)
```

Arguments

x   an R object

See Also

is_natural

Examples

```
is_integer(1) # TRUE
is_integer(-3) # TRUE
is_integer(pi) # FALSE
is_integer(iris$Species)

M = matrix(seq(-3, 2), 2, 3)
is_integer(M)
```

---

is_matrix   *Is matrix*

Description

is_matrix tests if an object is a matrix
is_numeric_matrix tests if an object is a numeric matrix
is_string_matrix tests if an object is a string matrix
is_logical_matrix tests if an object is a logical matrix
is_not_matrix tests if an object is not a matrix

Arguments

x   an R object

Examples

```
A = matrix(1:10, 5, 2)
B = matrix(letters[1:10], 5, 2)
C = 1:10

is_matrix(A) # TRUE
is_matrix(C) # FALSE
is_not_matrix(C) # TRUE

is_numeric_matrix(A) # TRUE
is_numeric_matrix(B) # FALSE

is_string_matrix(A) # FALSE
is_string_matrix(B) # TRUE
```

---

is_multiple   *Is multiple*

Description

Tests if x is multiple of a given number

Usage

```
is_multiple(x, of)
```

Arguments

x   a numeric object

of   a given number

Examples

```
is_multiple(5, of = 5) # TRUE
is_multiple(15, of = 5) # TRUE
is_multiple(3, of = 5) # FALSE
is_multiple(2*pi, of = pi) # TRUE
is_multiple(matrix(1:6, 2, 3), of = 2)
```

---

is_natural   *Is natural*

Description

Test if is a natural number

Usage

```
is_natural(x)
```

Arguments

x   an R object

Details

Zero is not included in the set of natural numbers

See Also

is_negative

Examples

```
is_natural(1)
is_natural(0)
is_natural(seq(-2, 3))
is_natural(iris$Species)

M = matrix(seq(-3, 2), 2, 3)
is_natural(M)
```

---

is_negative   *Is negative*

Description

Test if an object is negative

Usage

```
is_negative(x)
```

Arguments

x   an R object

See Also

is_positive

Examples

```
is_negative(1)
is_negative(0)
is_negative(-1)
is_negative(iris$Species)
is_negative(iris)
is_negative(list(1, 0, -1, iris))

set.seed(999)
M = matrix(rnorm(12), 4, 3)
is_negative(M)
```

---

is_negative_decimal *Is negative decimal*

Description

Test if is a negative decimal

Usage

```
is_negative_decimal(x)
```

Arguments

x   an R object

Examples

```
is_negative_decimal(0.0001)
is_negative_decimal(-0.0003)
is_negative_decimal(0)
is_negative_decimal(pi)
is_negative_decimal(-exp(1))
```

---

is_negative_integer *Is negative integer*

Description

Test if is a positive integer

Usage

```
is_negative_integer(x)
```

Arguments

x   an R object

Examples

```
is_negative_integer(-1) # TRUE
is_negative_integer(1) # FALSE
is_negative_integer(0) # FALSE
is_negative_integer(pi) # FALSE
is_negative_integer(2.2) # FALSE
```

---

is_odd   *Is even*

Description

Test if an object is an even number
`is_not_odd` tests the opposite condition

Usage

```
is_odd(x)
```

Arguments

x   an R object

See Also

is_even

Examples

```
is_odd(2)
is_odd(1)
is_odd(seq(-5, 5))

is_odd(iris$Species)
is_odd(iris)
is_odd(list(1, 0, -1, iris))

set.seed(999)
M = matrix(1:12, 4, 3)
is_odd(M)
```

---

is_one_dim   *Test if an object has one-dimension*

Description

Returns `TRUE` if an object is a vector or a one-dimensional matrix, `FALSE` otherwise

Usage

```
is_one_dim(x)
```

Arguments

x   an R object

Value

whether x is one-dimensional

Examples

```
# vector
is_one_dim(1:5)  # TRUE

# factor
is_one_dim(iris$Species)  # TRUE

# one row matrix
is_one_dim(matrix(1:5, 1, 5))  # TRUE

# one column matrix
is_one_dim(matrix(1:5, 5, 1))  # TRUE

# general matrix (nrow>1, ncol>1)
is_one_dim(matrix(1:9, 3, 3))  # FALSE

# general data frame
is_one_dim(iris)  # FALSE
```

---

is_positive   *Is positive*

Description

Test if an object is positive

Usage

```
is_positive(x)
```

Arguments

x   an R object

See Also

is_negative

Examples

```
is_positive(1)
is_positive(0)
is_positive(-1)
is_positive(iris$Species)
is_positive(iris)
```

```
is_positive(list(1, 0, -1, iris))

set.seed(999)
M = matrix(rnorm(12), 4, 3)
is_positive(M)
```

is_positive_decimal   *Is positive decimal*

Description

Test if is a positive decimal

Usage

```
is_positive_decimal(x)
```

Arguments

x   an R object

Examples

```
is_positive_decimal(0.0001)
is_positive_decimal(-0.0003)
is_positive_decimal(0)
is_positive_decimal(pi)
is_positive_decimal(-exp(1))
```

is_positive_integer   *Is positive integer*

Description

Test if is a positive integer

Usage

```
is_positive_integer(x)
```

Arguments

x   an R object

Examples

```
is_positive_integer(1) # TRUE
is_positive_integer(0) # FALSE
is_positive_integer(pi) # FALSE
is_positive_integer(2.2) # FALSE
is_positive_integer(-1) # FALSE
```

---

is_scalar   *Is scalar*

Description

Tests if an object is a scalar number
`is_scalar` tests if an object is a scalar
`is_not_scalar` tests if an object is not a scalar
`is_positive_scalar` tests if an object is a positive scalar
`is_negative_scalar` tests if an object is a negative scalar

Arguments

x   an R object

See Also

[is_single_number](#)

Examples

```
is_scalar(1)  # TRUE
is_scalar(pi)  # TRUE
is_scalar(1:5)  # FALSE
is_scalar(matrix(runif(4), 2, 2))  # FALSE

is_not_scalar(1:5)  # TRUE
is_not_scalar(NULL)  # TRUE
is_not_scalar(matrix(runif(4), 2, 2))  # TRUE

is_positive_scalar(1.0)  # TRUE
is_positive_scalar(0)  # FALSE
is_positive_scalar(-10)  # FALSE
is_positive_scalar("hoskdflksfd")  # FALSE
is_positive_scalar(NA)  # FALSE

is_negative_scalar(-1)  # TRUE
is_negative_scalar(0)  # FALSE
is_negative_scalar(10)  # FALSE
is_negative_scalar("hoskdflksfd")  # FALSE
is_negative_scalar(NA)  # FALSE
```

---

is_single   *Is single*

Description

Tests if an object is single (i.e. of length 1)

Usage

```
is_single(x)
```

Arguments

x   an R object

See Also

[is_single_number](#), [is_single_string](#), [is_single_logical](#)

Examples

```
is_single("hoskdflksfd")  # TRUE
is_single("1.0")  # TRUE
is_single(1:5)  # FALSE
is_single(matrix(runif(4), 2, 2))  # FALSE
```

---

is_single_decimal   *Is single decimal*

Description

Tests if an object is a single decimal number

Usage

```
is_single_decimal(x)
```

Arguments

x   an R object

See Also

[is_single](#)

Examples

```
is_single_decimal(0.01)  # TRUE
is_single_decimal(-3/4)  # TRUE
is_single_decimal("hoskdflksfd")  # FALSE
is_single_decimal("1.0")  # FALSE
is_single_decimal(1:5)  # FALSE
```

---

is_single_even   *Is single even*

Description

Tests if an object is a single even number

Usage

```
is_single_even(x)
```

Arguments

x   an R object

See Also

[is_single](), [is_single_odd]()

Examples

```
is_single_even(2)  # TRUE
is_single_even(5)  # FALSE
is_single_even(c(1.0,2))  # FALSE
is_single_even(-1.0)  # FALSE
is_single_even(0)  # TRUE
is_single_even(NA)  # FALSE
```

---

is_single_false   *Is single false*

Description

Tests if an object is a single FALSE

Usage

```
is_single_false(x)
```

Arguments

x   an R object

See Also

[is_single](), [is_single_true]()

Examples

```
is_single_false(FALSE)  # TRUE
is_single_false(TRUE)  # FALSE
is_single_false(c(TRUE, FALSE))  # FALSE
is_single_false(-1.0)  # FALSE
is_single_false(0)  # FALSE
is_single_false(NA)  # FALSE
```

---

is_single_logical   *Is single logical*

Description

Tests if an object is a single logical

Usage

```
is_single_logical(x)
```

Arguments

x   an R object

See Also

[is_single](), [is_single_true](), [is_single_false]()

Examples

```
is_single_logical(TRUE)  # TRUE
is_single_logical(FALSE)  # TRUE
is_single_logical(c(TRUE, FALSE))  # FALSE
is_single_logical(-1.0)  # FALSE
is_single_logical(0)  # FALSE
is_single_logical(NA)  # FALSE
```

is_single_negative *Is single negative number*

Description

Tests if an object is a single negative number

Usage

```
is_single_negative(x)
```

Arguments

x   an R object

See Also

[is_single](), [is_single_positive]()

Examples

```
is_single_negative(1.0)  # FALSE
is_single_negative(-1.0)  # TRUE
is_single_negative(c(-1.0,-2))  # FALSE
is_single_negative(0)  # FALSE
is_single_negative(NA)  # FALSE
```

is_single_negative_decimal

Is single negative decimal

Description

Tests if an object is a single positive decimal

Usage

```
is_single_negative_decimal(x)
```

Arguments

x   an R object

See Also

[is_single](), [is_single_negative](), [is_single_positive_decimal]()

Examples

```
is_single_negative_decimal(-3/4)  # TRUE
is_single_negative_decimal(0.01)  # FALSE
is_single_negative_decimal("hoskdflksfd")  # FALSE
is_single_negative_decimal("1.0")  # FALSE
is_single_negative_decimal(1:5)  # FALSE
```

---

is_single_negative_integer

Is single negative integer

Description

Tests if an object is a single negative integer

Usage

```
is_single_negative_integer(x)
```

Arguments

x   an R object

See Also

[is_single](#), [is_single_positive_integer](#)

Examples

```
is_single_negative_integer(-1.0)  # TRUE
is_single_negative_integer(1.0)  # FALSE
is_single_negative_integer(c(1.0,2))  # FALSE
is_single_negative_integer(0)  # FALSE
is_single_negative_integer(NA)  # FALSE
```

---

is_single_number   *Is single number*

Description

Tests if an object is a single number

Usage

```
is_single_number(x)
```

Arguments

x   an R object

See Also

is_single

Examples

```
is_single_number(1.0)  # TRUE
is_single_number("hoskdflksfd")  # FALSE
is_single_number("1.0")  # FALSE
is_single_number(1:5)  # FALSE
```

---

is_single_odd   *Is single odd*

Description

Tests if an object is a single odd number

Usage

```
is_single_odd(x)
```

Arguments

x   an R object

See Also

is_single, is_single_even

Examples

```
is_single_odd(1.0)  # TRUE
is_single_odd(2)  # FALSE
is_single_odd(c(1.0,2))  # FALSE
is_single_odd(2)  # FALSE
is_single_odd(0)  # FALSE
is_single_odd(NA)  # FALSE
```

`is_single_positive`   *Is single positive number*

Description

Tests if an object is a single positive number

Usage

```
is_single_positive(x)
```

Arguments

x   an R object

See Also

is_single, is_single_negative

Examples

```
is_single_positive(1.0)  # TRUE
is_single_positive(c(1.0,2))  # FALSE
is_single_positive(-1.0)  # FALSE
is_single_positive(0)  # FALSE
is_single_positive(NA)  # FALSE
```

`is_single_positive_decimal`

Is single positive decimal

Description

Tests if an object is a single positive decimal

Usage

```
is_single_positive_decimal(x)
```

Arguments

x   an R object

See Also

is_single, is_single_positive, is_single_negative_decimal

Examples

```
is_single_positive_decimal(0.01)  # TRUE
is_single_positive_decimal(-3/4)  # FALSE
is_single_positive_decimal("hoskdflksfd")  # FALSE
is_single_positive_decimal("1.0")  # FALSE
is_single_positive_decimal(1:5)  # FALSE
```

---

is_single_positive_integer

Is single positive integer

Description

Tests if an object is a single positive integer

Usage

```
is_single_positive_integer(x)
```

Arguments

x   an R object

See Also

[is_single](#), [is_single_negative_integer](#)

Examples

```
is_single_positive_integer(1.0)  # TRUE
is_single_positive_integer(c(1.0,2))  # FALSE
is_single_positive_integer(-1.0)  # FALSE
is_single_positive_integer(0)  # FALSE
is_single_positive_integer(NA)  # FALSE
```

---

is_single_string   *Is single string*

Description

Tests if an object is a single string

Usage

```
is_single_string(x)
```

Arguments

x   an R object

See Also

is_single

Examples

```
is_single_string(1.0)  # FALSE
is_single_string("hoskdflksfd")  # TRUE
is_single_string(c("1.0", "sd"))  # FALSE
```

---

is_single_true   *Is single true*

Description

Tests if an object is a single TRUE

Usage

```
is_single_true(x)
```

Arguments

x   an R object

See Also

is_single, is_single_false

Examples

```
is_single_true(TRUE)  # TRUE
is_single_true(FALSE)  # FALSE
is_single_true(c(TRUE, FALSE))  # FALSE
is_single_true(-1.0)  # FALSE
is_single_true(0)  # FALSE
is_single_true(NA)  # FALSE
```

`is_square_matrix`   *Is square matrix*

Description

Test if an object is a square matrix (or not) (i.e. same number of rows as number of columns)

Usage

```
is_square_matrix(x)
```

Arguments

x   an R object

See Also

is_matrix

Examples

```
m1 = matrix(1:9, 3, 3)
m2 = matrix(1:12, 4, 3)

is_square_matrix(m1) # TRUE
is_square_matrix(m2) # FALSE
is_not_square_matrix(m2) # TRUE
```

`is_square_numeric_matrix`
                    Is square numeric matrix

Description

Test if an object is a square numeric matrix (or not) (i.e. same number of rows as number of columns)

Usage

```
is_square_numeric_matrix(x)
```

Arguments

x   an R object

See Also

is_matrix, is_square_matrix

Examples

```
# numeric matrices
m1 = matrix(1:9, 3, 3)
m2 = matrix(1:12, 4, 3)

is_square_numeric_matrix(m1)  # TRUE
is_square_numeric_matrix(m2)  # FALSE
is_not_square_numeric_matrix(m2)  # TRUE

# non-numeric matrices
str_mat = matrix(letters[1:9], 3, 3)
log_mat = matrix(rep_len(c(TRUE, FALSE), 9), 3, 3)

is_square_numeric_matrix(str_mat)  # FALSE
is_square_numeric_matrix(log_mat)  # FALSE
is_not_square_numeric_matrix(str_mat)  # TRUE
```

---

is_string   *Is string*

Description

Tests if an object is a character string
is_not_string() tests the opposite condition

Usage

```
is_string(x)
```

Arguments

x   an R object

Examples

```
is_string("test_me") # TRUE

is_string(1:10) # FALSE
```

---

is_tabular *Is tabular*

Description

is_tabular tests if an object has a tabular format (i.e. a matrix or data frame)

is_not_tabular tests if an object doesn't have a tabular format (i.e. not matrix nor data frame)

is_numeric_tabular tests if an object is a numeric table (i.e. a numeric matrix or data frame)

is_string_tabular tests if an object is a string table

Arguments

x   an R object

Examples

```
A = matrix(1:10, 5, 2)
B = matrix(letters[1:10], 5, 2)
C = 1:10

is_tabular(A) # TRUE
is_tabular(iris) # TRUE

is_numeric_tabular(A) # TRUE
is_numeric_tabular(iris) # FALSE
is_numeric_dataframe(iris[,1:4]) # TRUE
```

---

is_triangular_matrix *Is triangular matrix*

Description

is_lower_triangular tests if a matrix is lower triangular

is_upper_triangular tests if a matrix is upper triangular

is_triangular_matrix tests if a matrix is triangular (both lower or upper triangular)

Arguments

x   a matrix

diag   should the diagonal be included? (FALSE by default)

Examples

```
some_matrix = matrix(1:9, 3, 3)
lower_matrix <- upper_matrix <- some_matrix
lower_matrix[upper.tri(some_matrix)] <- 0
upper_matrix[lower.tri(some_matrix)] <- 0

is_triangular_matrix(some_matrix) # TRUE
is_triangular_matrix(lower_matrix) # TRUE
is_triangular_matrix(upper_matrix) # TRUE

is_lower_triangular(some_matrix) # FALSE
is_lower_triangular(lower_matrix) # FALSE
is_lower_triangular(upper_matrix) # FALSE

is_upper_triangular(some_matrix) # FALSE
is_upper_triangular(lower_matrix) # FALSE
is_upper_triangular(upper_matrix) # FALSE
```

---

is_TRUE   *If TRUE or FALSE*

Description

is_TRUE and is_true tests if x is TRUE
is_FALSE and is_false tests if x is FALSE
true_or_false returns whether the condition is true or false

Arguments

x   an R object

Examples

```
is_true(TRUE)
is_true(FALSE)
is_false(TRUE)
is_false(FALSE)
true_or_false(TRUE)
true_or_false(FALSE)

is_true(1) # FLASE
is_false("FALSE") # FALSE
```

---

is_vector   *Is vector*

Description

is_vector tests if an object is a vector

is_numeric_vector tests if an object is a numeric vector

is_string_vector tests if an object is a string vector

is_logical_vector tests if an object is a logical vector

is_not_vector tests if an object is not a vector

Arguments

x   an R object

Examples

```
a = 1:10
b = letters[1:10]
d = matrix(1:10, 5, 2)

is_vector(a) # TRUE
is_vector(b) # TRUE
is_vector(d) # FALSE
is_not_vector(d) # TRUE

is_numeric_vector(a) # TRUE
is_numeric_vector(b) # FALSE

is_string_vector(a) # FALSE
is_string_vector(b) # TRUE
```

---

list_of_vectors   *List of vectors*

Description

list_of_vectors checks if an object is a list of vectors

list_of_numeric_vectors checks if an object is a list of numeric vectors

list_of_string_vectors checks if an object is a list of string vectors list_of_logical_vectors
checks if an object is a list of logical vectors

Arguments

x   an R object

See Also

is_vector, list_with_vectors

Examples

```
a = list(1:3, letters[1:3], c(exp(1), pi), NA)
b = list(1:3, c(exp(1), pi))
d = list(letters[1:3], 'bonjour a tous')
e = list(matrix(1:6, 2, 3), a, b)

list_of_vectors(a) # TRUE
list_of_vectors(b) # TRUE
list_of_vectors(d) # TRUE
list_of_vectors(e) # FALSE

list_of_numeric_vectors(a) # FALSE
list_of_numeric_vectors(b) # TRUE

list_of_string_vectors(a) # FALSE
list_of_string_vectors(d) # TRUE

list_of_logical_vectors(a) # FALSE
list_of_logical_vectors(d) # TRUE
```

---

list_with_vectors   *List with vectors*

Description

list_with_vectors checks if an object is a list with vectors
list_with_numeric_vectors checks if an object is a list with numeric vectors
list_with_string_vectors checks if an object is a list with string vectors

Arguments

x   an R object

See Also

is_vector, list_of_vectors

Examples

```
a = list(1:3, letters[1:3], c(exp(1), pi), NA)
b = list(1:3, c(exp(1), pi))
d = list(letters[1:3], 'bonjour a tous')
e = list(matrix(1:6, 2, 3), a, b)

list_with_vectors(1:10) # FALSE
```

```
list_with_vectors(b) # TRUE
list_with_vectors(d) # TRUE

list_with_numeric_vectors(a) # TRUE
list_with_numeric_vectors(b) # TRUE
list_with_string_vectors(d) # FALSE

list_with_string_vectors(a) # TRUE
list_with_string_vectors(d) # TRUE
list_with_string_vectors(b) # FALSE
```

---

same_class   *Same Class*

Description

same_class() tests if two objects have the same class
different_class() tests if two objects have different class

Usage

```
same_class(x, y)
```

Arguments

x   an R object

y   an R object

Examples

```
same_class(letters[1:3], "class") # TRUE
same_class(1:3, "class") # FALSE
```

---

same_dim   *Same Dimension*

Description

same_dim() tests if two matrices have same dimension
different_dim() tests if two matrices have different dimension

Usage

```
same_dim(x, y)
```

Arguments

x	a matrix
y	a matrix

See Also

same_nrow

Examples

```
a = matrix(1:15, 5, 3)

same_dim(a, a) # TRUE
same_dim(a, t(a)) # FALSE

different_dim(a, a) # FALSE
different_dim(a, t(a)) # TRUE
```

---

same_length	*Same Length*

Description

same_length() tests if two objects have same length
different_length() tests if two objects have different length

Usage

```
same_length(x, y)
```

Arguments

x	a matrix
y	a matrix

Examples

```
same_length(1:10, letters[11:20]) # TRUE
same_length(1:10, letters[11:19]) # FALSE

a = matrix(1:15, 5, 3)
same_length(a, a) # TRUE
same_length(a, t(a)) # TRUE

different_length(t(a), a) # FALSE
different_length(1:10, a) # TRUE
different_length(a, "a") # TRUE
```

---

same_mode   *Same Mode*

Description

same_mode() tests if two objects have the same mode
different_mode() tests if two objects have different mode

Usage

```
same_mode(x, y)
```

Arguments

x	an R object
y	an R object

Examples

```
same_mode(letters[1:3], "class") # TRUE
same_mode(1:3, "class") # FALSE
```

---

same_nrow   *Same Number of Rows / Columns*

Description

same_nrow() tests if two matrices have same number of rows
different_nrow() tests if two matrices have different number of rows
same_ncol() tests if two matrices have same number of columns
different_ncol() tests if two matrices have different number of columns

Usage

```
same_nrow(x, y)
```

Arguments

x	a matrix
y	a matrix

See Also

[same_dim](#)

Examples

```
a = matrix(1:15, 5, 3)

same_nrow(a, a) # TRUE
same_nrow(a, t(a)) # FALSE
same_ncol(a, a) # TRUE
same_ncol(a, t(a)) # FALSE

different_nrow(a, a) # FALSE
different_nrow(a, t(a)) # TRUE
different_ncol(a, a) # FALSE
different_ncol(a, t(a)) # TRUE
```

---

same_type   *Same Type*

Description

same_type() tests if two objects have the same type
different_type() tests if two objects have different type

Usage

```
same_type(x, y)
```

Arguments

x	an R object
y	an R object

Examples

```
same_type(letters[1:3], "class") # TRUE
same_type(1:3, "class") # FALSE

different_type(1, 1L) # TRUE
different_type(1, 1.0) # FALSE
```

Index