

Package ‘admiraldev’

December 1, 2022

Type Package

Title Development Tools for the Admiral Package Family

Version 0.2.0

Description Utility functions to check data, variables and conditions for functions used in 'admiral' and 'admiral' extension packages. Additional utility helper functions to assist developers with maintaining documentation, testing and general upkeep of 'admiral' and 'admiral' extension packages.

License Apache License (>= 2)

URL <https://pharmaverse.github.io/admiraldev/main/>,
<https://github.com/pharmaverse/admiraldev/>

Encoding UTF-8

Language en-US

LazyData false

RoxygenNote 7.2.1

Depends R (>= 3.5)

Imports dplyr (>= 0.8.4), hms (>= 0.5.3), lifecycle (>= 0.1.0),
lubridate (>= 1.7.4), magrittr (>= 1.5), purrr (>= 0.3.3),
rlang (>= 0.4.4), stringr (>= 1.4.0), tidyr (>= 1.0.2),
tidyselect (>= 1.0.0)

Suggests admiral.test, devtools, diffdf, lintr, pkgdown, testthat,
knitr, methods, miniUI, rmarkdown, roxygen2, rstudioapi,
spelling, styler, tibble, usethis, covr, DT

VignetteBuilder knitr

NeedsCompilation no

Author Ben Straub [aut, cre],
Stefan Bundfuss [aut],
Thomas Neitmann [aut],
Samia Kabi [aut],
Pooja Kumari [aut],
Syed Mubasheer [aut],
Ross Farrugia [aut],

Sadchla Mascary [aut],
 Zelos Zhu [aut],
 Jeffrey Dickinson [aut],
 Ania Golab [aut],
 Ondrej Slama [ctb],
 F. Hoffmann-La Roche AG [cph, fnd],
 GlaxoSmithKline LLC [cph, fnd]

Maintainer Ben Straub <ben.x.straub@gsk.com>

Repository CRAN

Date/Publication 2022-12-01 00:10:02 UTC

R topics documented:

| | |
|---------------------------------------|----|
| add_suffix_to_vars | 3 |
| anti_join | 4 |
| arg_name | 5 |
| assert_atomic_vector | 6 |
| assert_character_scalar | 7 |
| assert_character_vector | 8 |
| assert_data_frame | 9 |
| assert_date_var | 10 |
| assert_date_vector | 12 |
| assert_expr | 13 |
| assert_filter_cond | 14 |
| assert_function | 15 |
| assert_function_param | 16 |
| assert_has_variables | 17 |
| assert_integer_scalar | 18 |
| assert_list_element | 19 |
| assert_list_of | 20 |
| assert_logical_scalar | 21 |
| assert_named_exprs | 22 |
| assert_numeric_vector | 23 |
| assert_one_to_one | 24 |
| assert_order_vars | 25 |
| assert_param_does_not_exist | 26 |
| assert_s3_class | 27 |
| assert_same_type | 28 |
| assert_symbol | 29 |
| assert_unit | 30 |
| assert_vars | 31 |
| assert_varval_list | 32 |
| as_name | 34 |
| backquote | 34 |
| contains_vars | 35 |
| convert_dtm_to_dtc | 35 |
| dataset_vignette | 36 |

dquote 37

enumerate 37

expect_dfs_equal 38

extract_vars 39

filter_if 40

get_constant_vars 41

get_dataset 41

get_duplicates 42

get_new_tmp_var 42

get_source_vars 43

is_auto 44

is_named 44

is_order_vars 45

is_valid_dtc 45

quo_c 46

quo_not_missing 46

remove_tmp_vars 47

replace_symbol_in_quo 48

replace_values_by_names 49

squote 49

suppress_warning 50

valid_time_units 51

vars2chr 51

warn_if_incomplete_dtc 52

warn_if_inconsistent_list 52

warn_if_invalid_dtc 53

warn_if_vars_exist 54

what_is_it 55

%notin% 56

%or% 56

Index **58**

add_suffix_to_vars *Add a Suffix to Variables in a List of Quosures*

Description

Add a suffix to variables in a list of quosures

Usage

add_suffix_to_vars(order, vars, suffix)

Arguments

| | |
|--------|--|
| order | List of quosures <i>Permitted Values:</i> list of variables or desc(<variable>) function calls created by vars(), e.g., vars(ADT, desc(AVAL)) |
| vars | Variables to change <i>Permitted Values:</i> list of variables created by vars() |
| suffix | Suffix <i>Permitted Values:</i> A character scalar |

Value

The list of quosures where for each element the suffix (suffix) is added to every symbol specified for vars

Author(s)

Stefan Bundfuss

See Also

Helpers for working with Quosures: [quo_c\(\)](#), [quo_not_missing\(\)](#), [replace_symbol_in_quo\(\)](#), [replace_values_by_names\(\)](#)

Examples

```
library(dplyr, warn.conflicts = FALSE)

add_suffix_to_vars(vars(ADT, desc(AVAL), AVALC), vars = vars(AVAL), suffix = ".join")
```

 anti_join

Join Functions

Description

The *_join() functions from dplyr without a warning on different attributes in datasets.

Usage

```
anti_join(x, y, by = NULL, copy = FALSE, ...)

inner_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)

left_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)
```

Arguments

| | |
|--------|----------------------|
| x | data.frame |
| y | data.frame |
| by | character vector |
| copy | logical |
| ... | Additional arguments |
| suffix | character vector |

Value

data.frame

| | |
|----------|---|
| arg_name | <i>Extract Argument Name from an Expression</i> |
|----------|---|

Description

Extract Argument Name from an Expression

Usage

```
arg_name(expr)
```

Arguments

| | |
|------|---|
| expr | An expression created inside a function using <code>substitute()</code> |
|------|---|

Value

character vector

Author(s)

Thomas Neitmann, Ondrej Slama

See Also

Developer Utility Functions: [%notin%\(\)](#), [%or%\(\)](#), [as_name\(\)](#), [contains_vars\(\)](#), [convert_dtm_to_dtc\(\)](#), [extract_vars\(\)](#), [filter_if\(\)](#), [valid_time_units\(\)](#), [vars2chr\(\)](#)

assert_atomic_vector *Is an Argument an Atomic Vector?*

Description

Checks if an argument is an atomic vector

Usage

```
assert_atomic_vector(arg, optional = FALSE)
```

Arguments

| | |
|----------|--|
| arg | A function argument to be checked |
| optional | Is the checked parameter optional? If set to FALSE and arg is NULL then an error is thrown |

Value

The function throws an error if arg is not an atomic vector. Otherwise, the input is returned invisibly.

Author(s)

Ania Golab

See Also

Checks for valid input and returns warning or errors messages: [assert_character_scalar\(\)](#), [assert_character_vector\(\)](#), [assert_data_frame\(\)](#), [assert_date_vector\(\)](#), [assert_expr\(\)](#), [assert_filter_cond\(\)](#), [assert_function_param\(\)](#), [assert_function\(\)](#), [assert_has_variables\(\)](#), [assert_integer_scalar\(\)](#), [assert_list_element\(\)](#), [assert_list_of\(\)](#), [assert_logical_scalar\(\)](#), [assert_named_exprs\(\)](#), [assert_numeric_vector\(\)](#), [assert_one_to_one\(\)](#), [assert_order_vars\(\)](#), [assert_param_does_not_exist\(\)](#), [assert_s3_class\(\)](#), [assert_same_type\(\)](#), [assert_symbol\(\)](#), [assert_unit\(\)](#), [assert_vars\(\)](#), [assert_varval_list\(\)](#)

Examples

```
example_fun <- function(x) {  
  assert_atomic_vector(x)  
}  
  
example_fun(1:10)  
  
try(example_fun(list(1, 2)))
```

`assert_character_scalar`*Is an Argument a Character Scalar (String)?*

Description

Checks if an argument is a character scalar and (optionally) whether it matches one of the provided values.

Usage

```
assert_character_scalar(  
  arg,  
  values = NULL,  
  case_sensitive = TRUE,  
  optional = FALSE  
)
```

Arguments

| | |
|-----------------------------|---|
| <code>arg</code> | A function argument to be checked |
| <code>values</code> | A character vector of valid values for <code>arg</code> . Values is converted to a lower case vector if <code>case_sensitive = FALSE</code> is used. |
| <code>case_sensitive</code> | Should the argument be handled case-sensitive? If set to <code>FALSE</code> , the argument is converted to lower case for checking the permitted values and returning the argument. |
| <code>optional</code> | Is the checked parameter optional? If set to <code>FALSE</code> and <code>arg</code> is <code>NULL</code> then an error is thrown |

Value

The function throws an error if `arg` is not a character vector or if `arg` is a character vector but of length > 1 or if its value is not one of the values specified. Otherwise, the input is returned invisibly.

Author(s)

Thomas Neitmann

See Also

Checks for valid input and returns warning or errors messages: [assert_atomic_vector\(\)](#), [assert_character_vector\(\)](#), [assert_data_frame\(\)](#), [assert_date_vector\(\)](#), [assert_expr\(\)](#), [assert_filter_cond\(\)](#), [assert_function_param\(\)](#), [assert_function\(\)](#), [assert_has_variables\(\)](#), [assert_integer_scalar\(\)](#), [assert_list_element\(\)](#), [assert_list_of\(\)](#), [assert_logical_scalar\(\)](#), [assert_named_exprs\(\)](#), [assert_numeric_vector\(\)](#), [assert_one_to_one\(\)](#), [assert_order_vars\(\)](#), [assert_param_does_not_exist\(\)](#), [assert_s3_class\(\)](#), [assert_same_type\(\)](#), [assert_symbol\(\)](#), [assert_unit\(\)](#), [assert_vars\(\)](#), [assert_varval_list\(\)](#)

Examples

```
example_fun <- function(msg_type) {
  assert_character_scalar(msg_type, values = c("warning", "error"))
}

example_fun("warning")

try(example_fun("message"))

try(example_fun(TRUE))

# handling parameters case-insensitive
example_fun2 <- function(msg_type) {
  msg_type <- assert_character_scalar(
    msg_type,
    values = c("warning", "error"),
    case_sensitive = FALSE
  )
  if (msg_type == "warning") {
    print("A warning was requested.")
  }
}

example_fun2("Warning")
```

assert_character_vector

Is an Argument a Character Vector?

Description

Checks if an argument is a character vector

Usage

```
assert_character_vector(arg, values = NULL, optional = FALSE)
```

Arguments

| | |
|----------|--|
| arg | A function argument to be checked |
| values | A character vector of valid values for arg |
| optional | Is the checked parameter optional? If set to FALSE and arg is NULL then an error is thrown |

Value

The function throws an error if `arg` is not a character vector or if any element is not included in the list of valid values. Otherwise, the input is returned invisibly.

Author(s)

Thomas Neitmann

See Also

Checks for valid input and returns warning or errors messages: [assert_atomic_vector\(\)](#), [assert_character_scalar\(\)](#), [assert_data_frame\(\)](#), [assert_date_vector\(\)](#), [assert_expr\(\)](#), [assert_filter_cond\(\)](#), [assert_function_param\(\)](#), [assert_function\(\)](#), [assert_has_variables\(\)](#), [assert_integer_scalar\(\)](#), [assert_list_element\(\)](#), [assert_list_of\(\)](#), [assert_logical_scalar\(\)](#), [assert_named_exprs\(\)](#), [assert_numeric_vector\(\)](#), [assert_one_to_one\(\)](#), [assert_order_vars\(\)](#), [assert_param_does_not_exist\(\)](#), [assert_s3_class\(\)](#), [assert_same_type\(\)](#), [assert_symbol\(\)](#), [assert_unit\(\)](#), [assert_vars\(\)](#), [assert_varval_list\(\)](#)

Examples

```
example_fun <- function(chr) {
  assert_character_vector(chr)
}

example_fun(letters)

try(example_fun(1:10))
```

assert_data_frame *Is an Argument a Data Frame?*

Description

Checks if an argument is a data frame and (optionally) whether it contains a set of required variables

Usage

```
assert_data_frame(
  arg,
  required_vars = NULL,
  check_is_grouped = TRUE,
  optional = FALSE
)
```

Arguments

| | |
|------------------|--|
| arg | A function argument to be checked |
| required_vars | A list of variables created using <code>vars()</code> |
| check_is_grouped | Throw an error if dataset is grouped? Defaults to TRUE. |
| optional | Is the checked parameter optional? If set to FALSE and arg is NULL then an error is thrown |

Value

The function throws an error if `arg` is not a data frame or if `arg` is a data frame but misses any variable specified in `required_vars`. Otherwise, the input is returned invisibly.

Author(s)

Thomas Neitmann

See Also

Checks for valid input and returns warning or errors messages: [assert_atomic_vector\(\)](#), [assert_character_scalar\(\)](#), [assert_character_vector\(\)](#), [assert_date_vector\(\)](#), [assert_expr\(\)](#), [assert_filter_cond\(\)](#), [assert_function_param\(\)](#), [assert_function\(\)](#), [assert_has_variables\(\)](#), [assert_integer_scalar\(\)](#), [assert_list_element\(\)](#), [assert_list_of\(\)](#), [assert_logical_scalar\(\)](#), [assert_named_exprs\(\)](#), [assert_numeric_vector\(\)](#), [assert_one_to_one\(\)](#), [assert_order_vars\(\)](#), [assert_param_does_not_exist\(\)](#), [assert_s3_class\(\)](#), [assert_same_type\(\)](#), [assert_symbol\(\)](#), [assert_unit\(\)](#), [assert_vars\(\)](#), [assert_varval_list\(\)](#)

Examples

```
library(admiral.test)
library(dplyr, warn.conflicts = FALSE)
data(admiral_dm)

example_fun <- function(dataset) {
  assert_data_frame(dataset, required_vars = vars(STUDYID, USUBJID))
}

example_fun(admiral_dm)

try(example_fun(select(admiral_dm, -STUDYID)))

try(example_fun("Not a dataset"))
```

assert_date_var

Is a Variable in a Dataset a Date or Datetime Variable?

Description

Checks if a variable in a dataset is a date or datetime variable

Usage

```
assert_date_var(dataset, var, dataset_name = NULL, var_name = NULL)
```

Arguments

| | |
|--------------|---|
| dataset | The dataset where the variable is expected |
| var | The variable to check |
| dataset_name | The name of the dataset. If the argument is specified, the specified name is displayed in the error message. |
| var_name | The name of the variable. If the argument is specified, the specified name is displayed in the error message. |

Value

The function throws an error if `var` is not a date or datetime variable in `dataset` and returns the input invisibly otherwise.

Author(s)

Stefan Bundfuss

Examples

```
library(tibble)
library(lubridate)
library(rlang)

example_fun <- function(dataset, var) {
  var <- assert_symbol(enquo(var))
  assert_date_var(dataset = dataset, var = !!var)
}

my_data <- tribble(
  ~USUBJID, ~ADT,
  "1",      ymd("2020-12-06"),
  "2",      ymd("")
)

example_fun(
  dataset = my_data,
  var = ADT
)

try(example_fun(
  dataset = my_data,
  var = USUBJID
))

example_fun2 <- function(dataset, var) {
  var <- assert_symbol(enquo(var))
  assert_date_var(
    dataset = dataset,
    var = !!var,
    dataset_name = "your_data",
```

```
    var_name = "your_var"
  )
}

try(example_fun2(
  dataset = my_data,
  var = USUBJID
))
```

assert_date_vector *Is an object a date or datetime vector?*

Description

Check if an object/vector is a date or datetime variable without needing a dataset as input

Usage

```
assert_date_vector(arg, optional = TRUE)
```

Arguments

| | |
|----------|---|
| arg | The function argument to be checked |
| optional | Is the checked parameter optional? If set to FALSE and arg is NULL then the function assert_date_vector exits early and throw an error. |

Value

The function returns an error if arg is missing, or not a date or datetime variable but otherwise returns an invisible output.

Author(s)

Sadchla Mascary

See Also

Checks for valid input and returns warning or errors messages: [assert_atomic_vector\(\)](#), [assert_character_scalar\(\)](#), [assert_character_vector\(\)](#), [assert_data_frame\(\)](#), [assert_expr\(\)](#), [assert_filter_cond\(\)](#), [assert_function_param\(\)](#), [assert_function\(\)](#), [assert_has_variables\(\)](#), [assert_integer_scalar\(\)](#), [assert_list_element\(\)](#), [assert_list_of\(\)](#), [assert_logical_scalar\(\)](#), [assert_named_exprs\(\)](#), [assert_numeric_vector\(\)](#), [assert_one_to_one\(\)](#), [assert_order_vars\(\)](#), [assert_param_does_not_exist\(\)](#), [assert_s3_class\(\)](#), [assert_same_type\(\)](#), [assert_symbol\(\)](#), [assert_unit\(\)](#), [assert_vars\(\)](#), [assert_varval_list\(\)](#)

Examples

```
example_fun <- function(arg) {
  assert_date_vector(arg)
}

example_fun(
  as.Date("2022-01-30", tz = "UTC")
)
try(example_fun("1993-07-14"))
```

| | |
|-------------|---|
| assert_expr | <i>Assert Argument is an Expression</i> |
|-------------|---|

Description

Assert Argument is an Expression

Usage

```
assert_expr(arg, optional = FALSE)
```

Arguments

| | |
|----------|--|
| arg | A function argument to be checked |
| optional | Is the checked parameter optional? If set to FALSE and arg is NULL then an error is thrown |

Value

The function throws an error if arg is not an expression, i.e. either a symbol or a call, or returns the input invisibly otherwise

See Also

Checks for valid input and returns warning or errors messages: [assert_atomic_vector\(\)](#), [assert_character_scalar\(\)](#), [assert_character_vector\(\)](#), [assert_data_frame\(\)](#), [assert_date_vector\(\)](#), [assert_filter_cond\(\)](#), [assert_function_param\(\)](#), [assert_function\(\)](#), [assert_has_variables\(\)](#), [assert_integer_scalar\(\)](#), [assert_list_element\(\)](#), [assert_list_of\(\)](#), [assert_logical_scalar\(\)](#), [assert_named_exprs\(\)](#), [assert_numeric_vector\(\)](#), [assert_one_to_one\(\)](#), [assert_order_vars\(\)](#), [assert_param_does_not_exist\(\)](#), [assert_s3_class\(\)](#), [assert_same_type\(\)](#), [assert_symbol\(\)](#), [assert_unit\(\)](#), [assert_vars\(\)](#), [assert_varval_list\(\)](#)

assert_filter_cond *Is an Argument a Filter Condition?*

Description

Is an Argument a Filter Condition?

Usage

```
assert_filter_cond(arg, optional = FALSE)
```

Arguments

| | |
|----------|--|
| arg | Quosure - filtering condition. |
| optional | Logical - is the argument optional? Defaults to FALSE. |

Details

Check if arg is a suitable filtering condition to be used in functions like subset or dplyr::filter.

Value

Performs necessary checks and returns arg if all pass. Otherwise throws an informative error.

Author(s)

Ondrej Slama

See Also

Checks for valid input and returns warning or errors messages: [assert_atomic_vector\(\)](#), [assert_character_scalar\(\)](#), [assert_character_vector\(\)](#), [assert_data_frame\(\)](#), [assert_date_vector\(\)](#), [assert_expr\(\)](#), [assert_function_param\(\)](#), [assert_function\(\)](#), [assert_has_variables\(\)](#), [assert_integer_scalar\(\)](#), [assert_list_element\(\)](#), [assert_list_of\(\)](#), [assert_logical_scalar\(\)](#), [assert_named_exprs\(\)](#), [assert_numeric_vector\(\)](#), [assert_one_to_one\(\)](#), [assert_order_vars\(\)](#), [assert_param_does_not_exist\(\)](#), [assert_s3_class\(\)](#), [assert_same_type\(\)](#), [assert_symbol\(\)](#), [assert_unit\(\)](#), [assert_vars\(\)](#), [assert_varval_list\(\)](#)

Examples

```
library(admiral.test)
library(dplyr, warn.conflicts = FALSE)
library(rlang)
data(admiral_dm)

# typical usage in a function as a parameter check
example_fun <- function(dat, x) {
  x <- assert_filter_cond(enquo(x))
}
```

```
  filter(dat, !!x)
}

example_fun(admiral_dm, AGE == 64)

try(example_fun(admiral_dm, USUBJID))
```

| | |
|-----------------|--------------------------------|
| assert_function | <i>Is Argument a Function?</i> |
|-----------------|--------------------------------|

Description

Checks if the argument is a function and if all expected parameters are provided by the function.

Usage

```
assert_function(arg, params = NULL, optional = FALSE)
```

Arguments

| | |
|----------|--|
| arg | A function argument to be checked |
| params | A character vector of expected parameter names |
| optional | Is the checked parameter optional? If set to FALSE and arg is NULL then an error is thrown. |

Value

The function throws an error

- if the argument is not a function or
- if the function does not provide all parameters as specified for the params parameter.

Author(s)

Stefan Bundfuss

See Also

Checks for valid input and returns warning or errors messages: [assert_atomic_vector\(\)](#), [assert_character_scalar\(\)](#), [assert_character_vector\(\)](#), [assert_data_frame\(\)](#), [assert_date_vector\(\)](#), [assert_expr\(\)](#), [assert_filter_cond\(\)](#), [assert_function_param\(\)](#), [assert_has_variables\(\)](#), [assert_integer_scalar\(\)](#), [assert_list_element\(\)](#), [assert_list_of\(\)](#), [assert_logical_scalar\(\)](#), [assert_named_exprs\(\)](#), [assert_numeric_vector\(\)](#), [assert_one_to_one\(\)](#), [assert_order_vars\(\)](#), [assert_param_does_not_exist\(\)](#), [assert_s3_class\(\)](#), [assert_same_type\(\)](#), [assert_symbol\(\)](#), [assert_unit\(\)](#), [assert_vars\(\)](#), [assert_varval_list\(\)](#)

Examples

```
example_fun <- function(fun) {  
  assert_function(fun, params = c("x"))  
}  
  
example_fun(mean)  
  
try(example_fun(1))  
  
try(example_fun(sum))
```

assert_function_param *Assert Argument is a Parameter of a Function*

Description

Assert Argument is a Parameter of a Function

Usage

```
assert_function_param(arg, params)
```

Arguments

| | |
|--------|---|
| arg | The name of a function passed as a string |
| params | A character vector of function parameters |

Value

The function throws an error if any elements of params is not a parameter of the function given by arg

See Also

Checks for valid input and returns warning or errors messages: [assert_atomic_vector\(\)](#), [assert_character_scalar\(\)](#), [assert_character_vector\(\)](#), [assert_data_frame\(\)](#), [assert_date_vector\(\)](#), [assert_expr\(\)](#), [assert_filter_cond\(\)](#), [assert_function\(\)](#), [assert_has_variables\(\)](#), [assert_integer_scalar\(\)](#), [assert_list_element\(\)](#), [assert_list_of\(\)](#), [assert_logical_scalar\(\)](#), [assert_named_exprs\(\)](#), [assert_numeric_vector\(\)](#), [assert_one_to_one\(\)](#), [assert_order_vars\(\)](#), [assert_param_does_not_exist\(\)](#), [assert_s3_class\(\)](#), [assert_same_type\(\)](#), [assert_symbol\(\)](#), [assert_unit\(\)](#), [assert_vars\(\)](#), [assert_varval_list\(\)](#)

Examples

```
hello <- function(name) {  
  print(sprintf("Hello %s", name))  
}  
  
assert_function_param("hello", "name")  
  
try(assert_function_param("hello", "surname"))
```

assert_has_variables *Does a Dataset Contain All Required Variables?*

Description

Checks if a dataset contains all required variables

Usage

```
assert_has_variables(dataset, required_vars)
```

Arguments

dataset A data.frame
required_vars A character vector of variable names

Value

The function throws an error if any of the required variables are missing in the input dataset. Otherwise, the dataset is returned invisibly.

Author(s)

Thomas Neitmann

See Also

Checks for valid input and returns warning or errors messages: [assert_atomic_vector\(\)](#), [assert_character_scalar\(\)](#), [assert_character_vector\(\)](#), [assert_data_frame\(\)](#), [assert_date_vector\(\)](#), [assert_expr\(\)](#), [assert_filter_cond\(\)](#), [assert_function_param\(\)](#), [assert_function\(\)](#), [assert_integer_scalar\(\)](#), [assert_list_element\(\)](#), [assert_list_of\(\)](#), [assert_logical_scalar\(\)](#), [assert_named_exprs\(\)](#), [assert_numeric_vector\(\)](#), [assert_one_to_one\(\)](#), [assert_order_vars\(\)](#), [assert_param_does_not_exist\(\)](#), [assert_s3_class\(\)](#), [assert_same_type\(\)](#), [assert_symbol\(\)](#), [assert_unit\(\)](#), [assert_vars\(\)](#), [assert_varval_list\(\)](#)

Examples

```
library(admiral.test)
data(admiral_dm)

assert_has_variables(admiral_dm, "STUDYID")

try(assert_has_variables(admiral_dm, "AVAL"))
```

assert_integer_scalar *Is an Argument an Integer Scalar?*

Description

Checks if an argument is an integer scalar

Usage

```
assert_integer_scalar(arg, subset = "none", optional = FALSE)
```

Arguments

| | |
|----------|---|
| arg | A function argument to be checked |
| subset | A subset of integers that arg should be part of. Should be one of "none" (the default), "positive", "non-negative" or "negative". |
| optional | Is the checked parameter optional? If set to FALSE and arg is NULL then an error is thrown |

Value

The function throws an error if arg is not an integer belonging to the specified subset. Otherwise, the input is returned invisibly.

Author(s)

Thomas Neitmann

See Also

Checks for valid input and returns warning or errors messages: [assert_atomic_vector\(\)](#), [assert_character_scalar\(\)](#), [assert_character_vector\(\)](#), [assert_data_frame\(\)](#), [assert_date_vector\(\)](#), [assert_expr\(\)](#), [assert_filter_cond\(\)](#), [assert_function_param\(\)](#), [assert_function\(\)](#), [assert_has_variables\(\)](#), [assert_list_element\(\)](#), [assert_list_of\(\)](#), [assert_logical_scalar\(\)](#), [assert_named_exprs\(\)](#), [assert_numeric_vector\(\)](#), [assert_one_to_one\(\)](#), [assert_order_vars\(\)](#), [assert_param_does_not_exist\(\)](#), [assert_s3_class\(\)](#), [assert_same_type\(\)](#), [assert_symbol\(\)](#), [assert_unit\(\)](#), [assert_vars\(\)](#), [assert_varval_list\(\)](#)

Examples

```
example_fun <- function(num1, num2) {
  assert_integer_scalar(num1, subset = "positive")
  assert_integer_scalar(num2, subset = "negative")
}

example_fun(1, -9)

try(example_fun(1.5, -9))

try(example_fun(2, 0))

try(example_fun("2", 0))
```

assert_list_element *Is an Element of a List of Lists/Classes Fulfilling a Condition?*

Description

Checks if the elements of a list of named lists/classes fulfill a certain condition. If not, an error is issued and all elements of the list not fulfilling the condition are listed.

Usage

```
assert_list_element(list, element, condition, message_text, ...)
```

Arguments

| | |
|--------------|---|
| list | A list to be checked A list of named lists or classes is expected. |
| element | The name of an element of the lists/classes A character scalar is expected. |
| condition | Condition to be fulfilled The condition is evaluated for each element of the list. The element of the lists/classes can be referred to by its name, e.g., <code>sensor == 0</code> to check the sensor field of a class. |
| message_text | Text to be displayed in the message The text should describe the condition to be fulfilled, e.g., "For events the sensor values must be zero." |
| ... | Objects required to evaluate the condition If the condition contains objects apart from the element, they have to be passed to the function. See the second example below. |

Value

An error if the condition is not meet. The input otherwise.

Author(s)

Stefan Bundfuss

See Also

Checks for valid input and returns warning or errors messages: [assert_atomic_vector\(\)](#), [assert_character_scalar\(\)](#), [assert_character_vector\(\)](#), [assert_data_frame\(\)](#), [assert_date_vector\(\)](#), [assert_expr\(\)](#), [assert_filter_cond\(\)](#), [assert_function_param\(\)](#), [assert_function\(\)](#), [assert_has_variables\(\)](#), [assert_integer_scalar\(\)](#), [assert_list_of\(\)](#), [assert_logical_scalar\(\)](#), [assert_named_exprs\(\)](#), [assert_numeric_vector\(\)](#), [assert_one_to_one\(\)](#), [assert_order_vars\(\)](#), [assert_param_does_not_exist\(\)](#), [assert_s3_class\(\)](#), [assert_same_type\(\)](#), [assert_symbol\(\)](#), [assert_unit\(\)](#), [assert_vars\(\)](#), [assert_varval_list\(\)](#)

 assert_list_of

Is an Argument a List of Objects of a Specific S3 Class?

Description

Checks if an argument is a list of objects inheriting from the S3 class specified.

Usage

```
assert_list_of(arg, class, optional = TRUE)
```

Arguments

| | |
|----------|--|
| arg | A function argument to be checked |
| class | The S3 class to check for |
| optional | Is the checked parameter optional? If set to FALSE and arg is NULL then an error is thrown |

Value

The function throws an error if arg is not a list or if arg is a list but its elements are not objects inheriting from class. Otherwise, the input is returned invisibly.

Author(s)

Thomas Neitmann

See Also

Checks for valid input and returns warning or errors messages: [assert_atomic_vector\(\)](#), [assert_character_scalar\(\)](#), [assert_character_vector\(\)](#), [assert_data_frame\(\)](#), [assert_date_vector\(\)](#), [assert_expr\(\)](#), [assert_filter_cond\(\)](#), [assert_function_param\(\)](#), [assert_function\(\)](#), [assert_has_variables\(\)](#), [assert_integer_scalar\(\)](#), [assert_list_element\(\)](#), [assert_logical_scalar\(\)](#), [assert_named_exprs\(\)](#), [assert_numeric_vector\(\)](#), [assert_one_to_one\(\)](#), [assert_order_vars\(\)](#), [assert_param_does_not_exist\(\)](#), [assert_s3_class\(\)](#), [assert_same_type\(\)](#), [assert_symbol\(\)](#), [assert_unit\(\)](#), [assert_vars\(\)](#), [assert_varval_list\(\)](#)

Examples

```
example_fun <- function(list) {
  assert_list_of(list, "data.frame")
}

example_fun(list(mtcars, iris))

try(example_fun(list(letters, 1:10)))

try(example_fun(c(TRUE, FALSE)))
```

assert_logical_scalar *Is an Argument a Logical Scalar (Boolean)?*

Description

Checks if an argument is a logical scalar

Usage

```
assert_logical_scalar(arg, optional = FALSE)
```

Arguments

| | |
|----------|--|
| arg | A function argument to be checked |
| optional | Is the checked parameter optional? If set to FALSE and arg is NULL then an error is thrown. Otherwise, NULL is considered as valid value. |

Value

The function throws an error if arg is neither TRUE or FALSE. Otherwise, the input is returned invisibly.

Author(s)

Thomas Neitmann, Stefan Bundfuss

See Also

Checks for valid input and returns warning or errors messages: [assert_atomic_vector\(\)](#), [assert_character_scalar\(\)](#), [assert_character_vector\(\)](#), [assert_data_frame\(\)](#), [assert_date_vector\(\)](#), [assert_expr\(\)](#), [assert_filter_cond\(\)](#), [assert_function_param\(\)](#), [assert_function\(\)](#), [assert_has_variables\(\)](#), [assert_integer_scalar\(\)](#), [assert_list_element\(\)](#), [assert_list_of\(\)](#), [assert_named_exprs\(\)](#), [assert_numeric_vector\(\)](#), [assert_one_to_one\(\)](#), [assert_order_vars\(\)](#), [assert_param_does_not_exist\(\)](#), [assert_s3_class\(\)](#), [assert_same_type\(\)](#), [assert_symbol\(\)](#), [assert_unit\(\)](#), [assert_vars\(\)](#), [assert_varval_list\(\)](#)

Examples

```
example_fun <- function(flag) {
  assert_logical_scalar(flag)
}

example_fun(FALSE)

try(example_fun(NA))

try(example_fun(c(TRUE, FALSE, FALSE)))

try(example_fun(1:10))
```

assert_named_exprs *Assert Argument is a Named List of Expressions*

Description

Assert Argument is a Named List of Expressions

Usage

```
assert_named_exprs(arg, optional = FALSE)
```

Arguments

| | |
|----------|--|
| arg | A function argument to be checked |
| optional | Is the checked parameter optional? If set to FALSE and arg is NULL then an error is thrown |

Value

The function throws an error if arg is not a named list of expression or returns the input invisibly otherwise

See Also

Checks for valid input and returns warning or errors messages: [assert_atomic_vector\(\)](#), [assert_character_scalar\(\)](#), [assert_character_vector\(\)](#), [assert_data_frame\(\)](#), [assert_date_vector\(\)](#), [assert_expr\(\)](#), [assert_filter_cond\(\)](#), [assert_function_param\(\)](#), [assert_function\(\)](#), [assert_has_variables\(\)](#), [assert_integer_scalar\(\)](#), [assert_list_element\(\)](#), [assert_list_of\(\)](#), [assert_logical_scalar\(\)](#), [assert_numeric_vector\(\)](#), [assert_one_to_one\(\)](#), [assert_order_vars\(\)](#), [assert_param_does_not_exist\(\)](#), [assert_s3_class\(\)](#), [assert_same_type\(\)](#), [assert_symbol\(\)](#), [assert_unit\(\)](#), [assert_vars\(\)](#), [assert_varval_list\(\)](#)

assert_numeric_vector *Is an Argument a Numeric Vector?*

Description

Checks if an argument is a numeric vector

Usage

```
assert_numeric_vector(arg, optional = FALSE)
```

Arguments

| | |
|----------|--|
| arg | A function argument to be checked |
| optional | Is the checked parameter optional? If set to FALSE and arg is NULL then an error is thrown |

Value

The function throws an error if arg is not a numeric vector. Otherwise, the input is returned invisibly.

Author(s)

Stefan Bundfuss

See Also

Checks for valid input and returns warning or errors messages: [assert_atomic_vector\(\)](#), [assert_character_scalar\(\)](#), [assert_character_vector\(\)](#), [assert_data_frame\(\)](#), [assert_date_vector\(\)](#), [assert_expr\(\)](#), [assert_filter_cond\(\)](#), [assert_function_param\(\)](#), [assert_function\(\)](#), [assert_has_variables\(\)](#), [assert_integer_scalar\(\)](#), [assert_list_element\(\)](#), [assert_list_of\(\)](#), [assert_logical_scalar\(\)](#), [assert_named_exprs\(\)](#), [assert_one_to_one\(\)](#), [assert_order_vars\(\)](#), [assert_param_does_not_exist\(\)](#), [assert_s3_class\(\)](#), [assert_same_type\(\)](#), [assert_symbol\(\)](#), [assert_unit\(\)](#), [assert_vars\(\)](#), [assert_varval_list\(\)](#)

Examples

```
example_fun <- function(num) {  
  assert_numeric_vector(num)  
}
```

```
example_fun(1:10)
```

```
try(example_fun(letters))
```

assert_one_to_one *Is There a One to One Mapping between Variables?*

Description

Checks if there is a one to one mapping between two lists of variables.

Usage

```
assert_one_to_one(dataset, vars1, vars2)
```

Arguments

| | |
|---------|--|
| dataset | Dataset to be checked The variables specified for vars1 and vars2 are expected. |
| vars1 | First list of variables |
| vars2 | Second list of variables |

Value

An error if the condition is not meet. The input otherwise.

Author(s)

Stefan Bundfuss

See Also

Checks for valid input and returns warning or errors messages: [assert_atomic_vector\(\)](#), [assert_character_scalar\(\)](#), [assert_character_vector\(\)](#), [assert_data_frame\(\)](#), [assert_date_vector\(\)](#), [assert_expr\(\)](#), [assert_filter_cond\(\)](#), [assert_function_param\(\)](#), [assert_function\(\)](#), [assert_has_variables\(\)](#), [assert_integer_scalar\(\)](#), [assert_list_element\(\)](#), [assert_list_of\(\)](#), [assert_logical_scalar\(\)](#), [assert_named_exprs\(\)](#), [assert_numeric_vector\(\)](#), [assert_order_vars\(\)](#), [assert_param_does_not_exist\(\)](#), [assert_s3_class\(\)](#), [assert_same_type\(\)](#), [assert_symbol\(\)](#), [assert_unit\(\)](#), [assert_vars\(\)](#), [assert_varval_list\(\)](#)

assert_order_vars *Is an Argument a List of Order Variables?*

Description

Checks if an argument is a valid list of order variables created using `vars()`

Usage

```
assert_order_vars(arg, optional = FALSE)
```

Arguments

| | |
|----------|--|
| arg | A function argument to be checked |
| optional | Is the checked parameter optional? If set to FALSE and arg is NULL then an error is thrown |

Value

The function throws an error if arg is not a list of variables or `desc()` calls created using `vars()` and returns the input invisibly otherwise.

Author(s)

Stefan Bundfuss

See Also

Checks for valid input and returns warning or errors messages: [assert_atomic_vector\(\)](#), [assert_character_scalar\(\)](#), [assert_character_vector\(\)](#), [assert_data_frame\(\)](#), [assert_date_vector\(\)](#), [assert_expr\(\)](#), [assert_filter_cond\(\)](#), [assert_function_param\(\)](#), [assert_function\(\)](#), [assert_has_variables\(\)](#), [assert_integer_scalar\(\)](#), [assert_list_element\(\)](#), [assert_list_of\(\)](#), [assert_logical_scalar\(\)](#), [assert_named_exprs\(\)](#), [assert_numeric_vector\(\)](#), [assert_one_to_one\(\)](#), [assert_param_does_not_exist\(\)](#), [assert_s3_class\(\)](#), [assert_same_type\(\)](#), [assert_symbol\(\)](#), [assert_unit\(\)](#), [assert_vars\(\)](#), [assert_varval_list\(\)](#)

Examples

```
library(dplyr, warn.conflicts = FALSE)
library(rlang)

example_fun <- function(by_vars) {
  assert_order_vars(by_vars)
}

example_fun(vars(USUBJID, PARAMCD, desc(AVISITN)))

try(example_fun(exprs(USUBJID, PARAMCD)))
```

```
try(example_fun(c("USUBJID", "PARAMCD", "VISIT")))  
try(example_fun(vars(USUBJID, toupper(PARAMCD), -AVAL)))
```

assert_param_does_not_exist

Asserts That a Parameter Does Not Exist in the Dataset

Description

Checks if a parameter (PARAMCD) does not exist in a dataset.

Usage

```
assert_param_does_not_exist(dataset, param)
```

Arguments

| | |
|---------|-------------------------|
| dataset | A data.frame |
| param | Parameter code to check |

Value

The function throws an error if the parameter exists in the input dataset. Otherwise, the dataset is returned invisibly.

Author(s)

Stefan Bundfuss

See Also

Checks for valid input and returns warning or errors messages: [assert_atomic_vector\(\)](#), [assert_character_scalar\(\)](#), [assert_character_vector\(\)](#), [assert_data_frame\(\)](#), [assert_date_vector\(\)](#), [assert_expr\(\)](#), [assert_filter_cond\(\)](#), [assert_function_param\(\)](#), [assert_function\(\)](#), [assert_has_variables\(\)](#), [assert_integer_scalar\(\)](#), [assert_list_element\(\)](#), [assert_list_of\(\)](#), [assert_logical_scalar\(\)](#), [assert_named_exprs\(\)](#), [assert_numeric_vector\(\)](#), [assert_one_to_one\(\)](#), [assert_order_vars\(\)](#), [assert_s3_class\(\)](#), [assert_same_type\(\)](#), [assert_symbol\(\)](#), [assert_unit\(\)](#), [assert_vars\(\)](#), [assert_varval_list\(\)](#)

Examples

```
library(tibble)
advs <- tribble(
  ~USUBJID, ~VSTESTCD, ~VSTRESN, ~VSSTRESU, ~PARAMCD, ~AVAL,
  "P01",    "WEIGHT",    80.1, "kg",    "WEIGHT",  80.1,
  "P02",    "WEIGHT",    85.7, "kg",    "WEIGHT",  85.7
)
assert_param_does_not_exist(advs, param = "HR")
try(assert_param_does_not_exist(advs, param = "WEIGHT"))
```

| | |
|------------------------------|---|
| <code>assert_s3_class</code> | <i>Is an Argument an Object of a Specific S3 Class?</i> |
|------------------------------|---|

Description

Checks if an argument is an object inheriting from the S3 class specified.

Usage

```
assert_s3_class(arg, class, optional = TRUE)
```

Arguments

| | |
|-----------------------|---|
| <code>arg</code> | A function argument to be checked |
| <code>class</code> | The S3 class to check for |
| <code>optional</code> | Is the checked parameter optional? If set to FALSE and <code>arg</code> is NULL then an error is thrown |

Value

The function throws an error if `arg` is an object which does *not* inherit from `class`. Otherwise, the input is returned invisibly.

Author(s)

Thomas Neitmann

See Also

Checks for valid input and returns warning or errors messages: [assert_atomic_vector\(\)](#), [assert_character_scalar\(\)](#), [assert_character_vector\(\)](#), [assert_data_frame\(\)](#), [assert_date_vector\(\)](#), [assert_expr\(\)](#), [assert_filter_cond\(\)](#), [assert_function_param\(\)](#), [assert_function\(\)](#), [assert_has_variables\(\)](#), [assert_integer_scalar\(\)](#), [assert_list_element\(\)](#), [assert_list_of\(\)](#), [assert_logical_scalar\(\)](#), [assert_named_exprs\(\)](#), [assert_numeric_vector\(\)](#), [assert_one_to_one\(\)](#), [assert_order_vars\(\)](#), [assert_param_does_not_exist\(\)](#), [assert_same_type\(\)](#), [assert_symbol\(\)](#), [assert_unit\(\)](#), [assert_vars\(\)](#), [assert_varval_list\(\)](#)

Examples

```
example_fun <- function(obj) {  
  assert_s3_class(obj, "factor")  
}  
  
example_fun(as.factor(letters))  
  
try(example_fun(letters))  
  
try(example_fun(1:10))
```

assert_same_type *Are All Argument of the Same Type?*

Description

Checks if all arguments are of the same type.

Usage

```
assert_same_type(...)
```

Arguments

... Arguments to be checked

Value

The function throws an error if not all arguments are of the same type.

Author(s)

Stefan Bundfuss

See Also

Checks for valid input and returns warning or errors messages: [assert_atomic_vector\(\)](#), [assert_character_scalar\(\)](#), [assert_character_vector\(\)](#), [assert_data_frame\(\)](#), [assert_date_vector\(\)](#), [assert_expr\(\)](#), [assert_filter_cond\(\)](#), [assert_function_param\(\)](#), [assert_function\(\)](#), [assert_has_variables\(\)](#), [assert_integer_scalar\(\)](#), [assert_list_element\(\)](#), [assert_list_of\(\)](#), [assert_logical_scalar\(\)](#), [assert_named_exprs\(\)](#), [assert_numeric_vector\(\)](#), [assert_one_to_one\(\)](#), [assert_order_vars\(\)](#), [assert_param_does_not_exist\(\)](#), [assert_s3_class\(\)](#), [assert_symbol\(\)](#), [assert_unit\(\)](#), [assert_vars\(\)](#), [assert_varval_list\(\)](#)

Examples

```
example_fun <- function(true_value, false_value, missing_value) {
  assert_same_type(true_value, false_value, missing_value)
}

example_fun(
  true_value = "Y",
  false_value = "N",
  missing_value = NA_character_
)

try(example_fun(
  true_value = 1,
  false_value = 0,
  missing_value = "missing"
))
```

assert_symbol

Is an Argument a Symbol?

Description

Checks if an argument is a symbol

Usage

```
assert_symbol(arg, optional = FALSE)
```

Arguments

| | |
|----------|--|
| arg | A function argument to be checked. Must be a quosure. See examples. |
| optional | Is the checked parameter optional? If set to FALSE and arg is NULL then an error is thrown |

Value

The function throws an error if `arg` is not a symbol and returns the input invisibly otherwise.

Author(s)

Thomas Neitmann

See Also

Checks for valid input and returns warning or errors messages: [assert_atomic_vector\(\)](#), [assert_character_scalar\(\)](#), [assert_character_vector\(\)](#), [assert_data_frame\(\)](#), [assert_date_vector\(\)](#), [assert_expr\(\)](#), [assert_filter_cond\(\)](#), [assert_function_param\(\)](#), [assert_function\(\)](#), [assert_has_variables\(\)](#), [assert_integer_scalar\(\)](#), [assert_list_element\(\)](#), [assert_list_of\(\)](#), [assert_logical_scalar\(\)](#), [assert_named_exprs\(\)](#), [assert_numeric_vector\(\)](#), [assert_one_to_one\(\)](#), [assert_order_vars\(\)](#), [assert_param_does_not_exist\(\)](#), [assert_s3_class\(\)](#), [assert_same_type\(\)](#), [assert_unit\(\)](#), [assert_vars\(\)](#), [assert_varval_list\(\)](#)

Examples

```
library(admiral.test)
library(dplyr, warn.conflicts = FALSE)
library(rlang)
data(admiral_dm)

example_fun <- function(dat, var) {
  var <- assert_symbol(enquo(var))
  select(dat, !!var)
}

example_fun(admiral_dm, USUBJID)

try(example_fun(admiral_dm))

try(example_fun(admiral_dm, "USUBJID"))

try(example_fun(admiral_dm, toupper(PARAMCD)))
```

 assert_unit

Asserts That a Parameter is Provided in the Expected Unit

Description

Checks if a parameter (PARAMCD) in a dataset is provided in the expected unit.

Usage

```
assert_unit(dataset, param, required_unit, get_unit_expr)
```

Arguments

| | |
|---------------|--|
| dataset | A data.frame |
| param | Parameter code of the parameter to check |
| required_unit | Expected unit |
| get_unit_expr | Expression used to provide the unit of param |

Value

The function throws an error if the unit variable differs from the unit for any observation of the parameter in the input dataset. Otherwise, the dataset is returned invisibly.

Author(s)

Stefan Bundfuss

See Also

Checks for valid input and returns warning or errors messages: [assert_atomic_vector\(\)](#), [assert_character_scalar\(\)](#), [assert_character_vector\(\)](#), [assert_data_frame\(\)](#), [assert_date_vector\(\)](#), [assert_expr\(\)](#), [assert_filter_cond\(\)](#), [assert_function_param\(\)](#), [assert_function\(\)](#), [assert_has_variables\(\)](#), [assert_integer_scalar\(\)](#), [assert_list_element\(\)](#), [assert_list_of\(\)](#), [assert_logical_scalar\(\)](#), [assert_named_exprs\(\)](#), [assert_numeric_vector\(\)](#), [assert_one_to_one\(\)](#), [assert_order_vars\(\)](#), [assert_param_does_not_exist\(\)](#), [assert_s3_class\(\)](#), [assert_same_type\(\)](#), [assert_symbol\(\)](#), [assert_vars\(\)](#), [assert_varval_list\(\)](#)

Examples

```
library(tibble)
advs <- tribble(
  ~USUBJID, ~VSTESTCD, ~VSTRESN, ~VSSTRESU, ~PARAMCD, ~AVAL,
  "P01",    "WEIGHT",    80.1, "kg",    "WEIGHT",    80.1,
  "P02",    "WEIGHT",    85.7, "kg",    "WEIGHT",    85.7
)

assert_unit(advs, param = "WEIGHT", required_unit = "kg", get_unit_expr = VSSTRESU)
```

assert_vars

Is an Argument a List of Variables?

Description

Checks if an argument is a valid list of variables created using `vars()`

Usage

```
assert_vars(arg, optional = FALSE, expect_names = FALSE)
```

Arguments

| | |
|--------------|---|
| arg | A function argument to be checked |
| optional | Is the checked parameter optional? If set to FALSE and arg is NULL then an error is thrown |
| expect_names | If the argument is set to TRUE, it is checked if all variables are named, e.g., <code>vars(APERSDT = APxxSDT, APEREDT = APxxEDT)</code> . |

Value

The function throws an error if `arg` is not a list of variables created using `vars()` and returns the input invisibly otherwise.

Author(s)

Samia Kabi

See Also

Checks for valid input and returns warning or errors messages: `assert_atomic_vector()`, `assert_character_scalar()`, `assert_character_vector()`, `assert_data_frame()`, `assert_date_vector()`, `assert_expr()`, `assert_filter_cond()`, `assert_function_param()`, `assert_function()`, `assert_has_variables()`, `assert_integer_scalar()`, `assert_list_element()`, `assert_list_of()`, `assert_logical_scalar()`, `assert_named_exprs()`, `assert_numeric_vector()`, `assert_one_to_one()`, `assert_order_vars()`, `assert_param_does_not_exist()`, `assert_s3_class()`, `assert_same_type()`, `assert_symbol()`, `assert_unit()`, `assert_varval_list()`

Examples

```
library(dplyr, warn.conflicts = FALSE)
library(rlang)

example_fun <- function(by_vars) {
  assert_vars(by_vars)
}

example_fun(vars(USUBJID, PARAMCD))

try(example_fun(exprs(USUBJID, PARAMCD)))

try(example_fun(c("USUBJID", "PARAMCD", "VISIT")))

try(example_fun(vars(USUBJID, toupper(PARAMCD), desc(AVAL))))

example_fun_name <- function(by_vars) {
  assert_vars(by_vars, expect_names = TRUE)
}

example_fun_name(vars(APERSDT = APxxSDT, APEREDT = APxxEDT))

try(example_fun_name(vars(APERSDT = APxxSDT, APxxEDT)))
```

`assert_varval_list` *Is an Argument a Variable-Value List?*

Description

Checks if the argument is a list of quosures where the expressions are variable-value pairs. The value can be a symbol, a string, a numeric, or NA. More general expression are not allowed.

Usage

```
assert_varval_list(
  arg,
  required_elements = NULL,
  accept_expr = FALSE,
  accept_var = FALSE,
  optional = FALSE
)
```

Arguments

| | |
|-------------------|---|
| arg | A function argument to be checked |
| required_elements | A character vector of names that must be present in arg |
| accept_expr | Should expressions on the right hand side be accepted? |
| accept_var | Should unnamed variable names (e.g. vars(USUBJID)) on the right hand side be accepted? |
| optional | Is the checked parameter optional? If set to FALSE and arg is NULL then an error is thrown. |

Value

The function throws an error if arg is not a list of variable-value expressions. Otherwise, the input is returned invisibly.

Author(s)

Stefan Bundfuss, Thomas Neitmann

See Also

Checks for valid input and returns warning or errors messages: [assert_atomic_vector\(\)](#), [assert_character_scalar\(\)](#), [assert_character_vector\(\)](#), [assert_data_frame\(\)](#), [assert_date_vector\(\)](#), [assert_expr\(\)](#), [assert_filter_cond\(\)](#), [assert_function_param\(\)](#), [assert_function\(\)](#), [assert_has_variables\(\)](#), [assert_integer_scalar\(\)](#), [assert_list_element\(\)](#), [assert_list_of\(\)](#), [assert_logical_scalar\(\)](#), [assert_named_exprs\(\)](#), [assert_numeric_vector\(\)](#), [assert_one_to_one\(\)](#), [assert_order_vars\(\)](#), [assert_param_does_not_exist\(\)](#), [assert_s3_class\(\)](#), [assert_same_type\(\)](#), [assert_symbol\(\)](#), [assert_unit\(\)](#), [assert_vars\(\)](#)

Examples

```
library(dplyr, warn.conflicts = FALSE)

example_fun <- function(vars) {
  assert_varval_list(vars)
}

example_fun(vars(DTHDOM = "AE", DTHSEQ = AESEQ))

try(example_fun(vars("AE", DTSEQ = AESEQ)))
```

| | |
|---------|-------------------------------------|
| as_name | <i>Turn a Quosure into a String</i> |
|---------|-------------------------------------|

Description

Turn a Quosure into a String

Usage

```
as_name(x)
```

Arguments

x A quosure

Details

This function is missing in earlier version of rlang which is why we re- implement it here.

Value

A character vector

See Also

Developer Utility Functions: [%notin%\(\)](#), [%or%\(\)](#), [arg_name\(\)](#), [contains_vars\(\)](#), [convert_dtm_to_dtc\(\)](#), [extract_vars\(\)](#), [filter_if\(\)](#), [valid_time_units\(\)](#), [vars2chr\(\)](#)

| | |
|-----------|------------------------------------|
| backquote | <i>Wrap a String in Backquotes</i> |
|-----------|------------------------------------|

Description

Wrap a String in Backquotes

Usage

```
backquote(x)
```

Arguments

x A character vector

Value

A character vector

Author(s)

Thomas Neitmann

See Also

Helpers for working with Quotes and Quoting: [dquote\(\)](#), [enumerate\(\)](#), [squote\(\)](#)

| | |
|---------------|---|
| contains_vars | <i>check that argument contains valid variable(s) created with vars() or Source Variables from a List of Quosures</i> |
|---------------|---|

Description

check that argument contains valid variable(s) created with vars() or Source Variables from a List of Quosures

Usage

```
contains_vars(arg)
```

Arguments

arg A function argument to be checked

Value

A TRUE if variables were valid variable

See Also

Developer Utility Functions: [%notin%](#)(), [%or%](#)(), [arg_name\(\)](#), [as_name\(\)](#), [convert_dtm_to_dtc\(\)](#), [extract_vars\(\)](#), [filter_if\(\)](#), [valid_time_units\(\)](#), [vars2chr\(\)](#)

| | |
|--------------------|---|
| convert_dtm_to_dtc | <i>Helper Function to Convert Date (or Date-time) Objects to Characters of dtc Format (-DTC type of variable)</i> |
|--------------------|---|

Description

Helper Function to Convert Date (or Date-time) Objects to Characters of dtc Format (-DTC type of variable)

Usage

```
convert_dtm_to_dtc(dtm)
```

Arguments

dtm date or date-time

Value

character vector

Author(s)

Ondrej Slama

See Also

Developer Utility Functions: [%notin%\(\)](#), [%or%\(\)](#), [arg_name\(\)](#), [as_name\(\)](#), [contains_vars\(\)](#), [extract_vars\(\)](#), [filter_if\(\)](#), [valid_time_units\(\)](#), [vars2chr\(\)](#)

| | |
|------------------|---|
| dataset_vignette | <i>Output a Dataset in a Vignette in the admiral Format</i> |
|------------------|---|

Description

Output a dataset in a vignette with the pre-specified admiral format.

Usage

```
dataset_vignette(dataset, display_vars = NULL, filter = NULL)
```

Arguments

| | |
|--------------|--|
| dataset | Dataset to output in the vignette |
| display_vars | Variables selected to demonstrate the outcome of the derivation Permitted Values: list of variables Default is NULL If display_vars is not NULL, only the selected variables are visible in the vignette while the other variables are hidden. They can be made visible by clicking the Choose the columns to display button. |
| filter | Filter condition The specified condition is applied to the dataset before it is displayed. Permitted Values: a condition |

Value

A HTML table

| | |
|--------|---------------------------------------|
| dquote | <i>Wrap a String in Double Quotes</i> |
|--------|---------------------------------------|

Description

Wrap a string in double quotes, e.g., for displaying character values in messages.

Usage

```
dquote(x)
```

Arguments

x A character vector

Value

If the input is NULL, the text "NULL" is returned. Otherwise, the input in double quotes is returned.

Author(s)

Stefan Bundfuss

See Also

Helpers for working with Quotes and Quoting: [backquote\(\)](#), [enumerate\(\)](#), [squote\(\)](#)

| | |
|-----------|-----------------------------------|
| enumerate | <i>Enumerate Multiple Strings</i> |
|-----------|-----------------------------------|

Description

Enumerate Multiple Strings

Usage

```
enumerate(x, quote_fun = backquote, conjunction = "and")
```

Arguments

x A character vector
quote_fun Quoting function, defaults to backquote.
conjunction Character to be used in the message, defaults to "and".

Value

A character vector

Author(s)

Thomas Neitmann

See Also

Helpers for working with Quotes and Quoting: [backquote\(\)](#), [dquote\(\)](#), [squote\(\)](#)

expect_dfs_equal

Expectation: Are Two Datasets Equal?

Description

Uses [diffdf::diffdf\(\)](#) to compares 2 datasets for any differences. This function can be thought of as an R-equivalent of SAS proc compare and a useful tool for unit testing as well.

Usage

```
expect_dfs_equal(base, compare, keys, ...)
```

Arguments

| | |
|---------|---|
| base | Input dataset |
| compare | Comparison dataset |
| keys | character vector of variables that define a unique row in the base and compare datasets |
| ... | Additional arguments passed onto diffdf::diffdf() |

Value

An error if base and compare do not match or NULL invisibly if they do

Author(s)

Thomas Neitmann

Examples

```

library(dplyr, warn.conflicts = FALSE)
library(tibble)

tbl1 <- tribble(
  ~USUBJID, ~AGE, ~SEX,
  "1001", 18, "M",
  "1002", 19, "F",
  "1003", 20, "M",
  "1004", 18, "F"
)

tbl2 <- tribble(
  ~USUBJID, ~AGE, ~SEX,
  "1001", 18, "M",
  "1002", 18.9, "F",
  "1003", 20, NA
)

try(expect_dfs_equal(tbl1, tbl2, keys = "USUBJID"))

tbl3 <- tribble(
  ~USUBJID, ~AGE, ~SEX,
  "1004", 18, "F",
  "1003", 20, "M",
  "1002", 19, "F",
  "1001", 18, "M",
)

# Note the sorting order of the keys is not required
expect_dfs_equal(tbl1, tbl3, keys = "USUBJID")

```

extract_vars

Extract All Symbols from a List of Quosures

Description

Extract All Symbols from a List of Quosures

Usage

```
extract_vars(x, side = "lhs")
```

Arguments

| | |
|------|-------------------------------------|
| x | An R object |
| side | One of "lhs" (the default) or "rhs" |

Value

A list of quosures

Author(s)

Thomas Neitmann

See Also

Developer Utility Functions: [%notin%\(\)](#), [%or%\(\)](#), [arg_name\(\)](#), [as_name\(\)](#), [contains_vars\(\)](#), [convert_dtm_to_dtc\(\)](#), [filter_if\(\)](#), [valid_time_units\(\)](#), [vars2chr\(\)](#)

filter_if

Optional Filter

Description

Filters the input dataset if the provided expression is not NULL

Usage

```
filter_if(dataset, filter)
```

Arguments

| | |
|---------|--|
| dataset | Input dataset |
| filter | A filter condition. Must be a quosure. |

Value

A data.frame containing all rows in dataset matching filter or just dataset if filter is NULL

Author(s)

Thomas Neitmann

See Also

Developer Utility Functions: [%notin%\(\)](#), [%or%\(\)](#), [arg_name\(\)](#), [as_name\(\)](#), [contains_vars\(\)](#), [convert_dtm_to_dtc\(\)](#), [extract_vars\(\)](#), [valid_time_units\(\)](#), [vars2chr\(\)](#)

get_constant_vars *Get Constant Variables*

Description

Get Constant Variables

Usage

```
get_constant_vars(dataset, by_vars, ignore_vars = NULL)
```

Arguments

| | |
|-------------|--|
| dataset | A data frame. |
| by_vars | By variables The groups defined by the by variables are considered separately. I.e., if a variable is constant within each by group, it is returned. |
| ignore_vars | Variables to ignore The specified variables are not considered, i.e., they are not returned even if they are constant (unless they are included in the by variables). <i>Permitted Values:</i> A list of variable names or selector function calls like starts_with("EX") |

Value

Variable vector.

See Also

Brings something to you!?: [get_duplicates\(\)](#), [get_source_vars\(\)](#)

get_dataset *Retrieve a Dataset from the admiraldev_environment environment*

Description

Retrieve a Dataset from the admiraldev_environment environment

Usage

```
get_dataset(name)
```

Arguments

| | |
|------|-------------------------------------|
| name | The name of the dataset to retrieve |
|------|-------------------------------------|

Value

A data.frame

Author(s)

Thomas Neitmann

| | |
|----------------|-------------------------------------|
| get_duplicates | <i>Get Duplicates From a Vector</i> |
|----------------|-------------------------------------|

Description

Get Duplicates From a Vector

Usage

```
get_duplicates(x)
```

Arguments

| | |
|---|------------------|
| x | An atomic vector |
|---|------------------|

Value

A vector of the same type as x contain duplicate values

See AlsoBrings something to you!?: [get_constant_vars\(\)](#), [get_source_vars\(\)](#)**Examples**

```
get_duplicates(1:10)

get_duplicates(c("a", "a", "b", "c", "d", "d"))
```

| | |
|-----------------|--|
| get_new_tmp_var | <i>Get a New Temporary Variable Name for a Dataset</i> |
|-----------------|--|

Description

Get a New Temporary Variable Name for a Dataset

Usage

```
get_new_tmp_var(dataset, prefix = "tmp_var")
```

Arguments

| | |
|---------|---|
| dataset | The input dataset |
| prefix | The prefix of the new temporary variable name to create |

Details

The function returns a new unique temporary variable name to be used inside dataset. The temporary variable names have the structure `prefix_n` where `n` is an integer, e.g. `tmp_var_1`. If there is already a variable inside dataset with a given prefix then the suffix is increased by 1, e.g. if `tmp_var_1` already exists then `get_new_tmp_var()` will return `tmp_var_2`.

Value

The name of a new temporary variable as a symbol

See Also

[remove_tmp_vars\(\)](#)

Examples

```
library(dplyr, warn.conflicts = FALSE)
library(admiral.test)
data(admiral_dm)

tmp_var <- get_new_tmp_var(admiral_dm)
mutate(admiral_dm, !!tmp_var := NA)
```

get_source_vars

Get Source Variables from a List of Quosures

Description

Get Source Variables from a List of Quosures

Usage

```
get_source_vars(quosures)
```

Arguments

`quosures` A list of quosures

Value

A list of quosures

Author(s)

Stefan Bundfuss

See Also

Brings something to you!?: [get_constant_vars\(\)](#), [get_duplicates\(\)](#)

| | |
|---------|---|
| is_auto | <i>Checks if the argument equals the auto keyword</i> |
|---------|---|

Description

Checks if the argument equals the auto keyword

Usage

```
is_auto(arg)
```

Arguments

arg argument to check

Value

TRUE if the argument equals the auto keyword, i.e., it is a quosure of a symbol named auto.

Author(s)

Stefan Bundfuss

See Also

Identifies type of Object with return of TRUE/FALSE: [is_named\(\)](#), [is_order_vars\(\)](#), [is_valid_dtc\(\)](#)

| | |
|----------|----------------------------|
| is_named | <i>Is a named argument</i> |
|----------|----------------------------|

Description

Is a named argument

Usage

```
is_named(x)
```

Arguments

x Any R object

Value

TRUE if the argument is named, FALSE otherwise

See Also

Identifies type of Object with return of TRUE/FALSE: [is_auto\(\)](#), [is_order_vars\(\)](#), [is_valid_dtc\(\)](#)

| | |
|---------------|-----------------------|
| is_order_vars | <i>Is order vars?</i> |
|---------------|-----------------------|

Description

Check if inputs are created using `vars()` or calls involving `desc()`

Usage

```
is_order_vars(arg)
```

Arguments

arg An R object

Value

FALSE if the argument is not a list of order vars

See Also

Identifies type of Object with return of TRUE/FALSE: [is_auto\(\)](#), [is_named\(\)](#), [is_valid_dtc\(\)](#)

| | |
|--------------|-----------------------------------|
| is_valid_dtc | <i>Is this string a valid DTC</i> |
|--------------|-----------------------------------|

Description

Is this string a valid DTC

Usage

```
is_valid_dtc(arg)
```

Arguments

arg A character vector

Value

TRUE if the argument is a valid --DTC string, FALSE otherwise

See Also

Identifies type of Object with return of TRUE/FALSE: [is_auto\(\)](#), [is_named\(\)](#), [is_order_vars\(\)](#)

| | |
|-------|---|
| quo_c | <i>Concatenate One or More Quosure(s)</i> |
|-------|---|

Description

Concatenate One or More Quosure(s)

Usage

```
quo_c(...)
```

Arguments

... One or more objects of class quosure or quosures

Value

An object of class quosures

Author(s)

Thomas Neitmann

See Also

Helpers for working with Quosures: [add_suffix_to_vars\(\)](#), [quo_not_missing\(\)](#), [replace_symbol_in_quo\(\)](#), [replace_values_by_names\(\)](#)

| | |
|-----------------|---|
| quo_not_missing | <i>Check Whether an Argument Is Not a Quosure of a Missing Argument</i> |
|-----------------|---|

Description

Check Whether an Argument Is Not a Quosure of a Missing Argument

Usage

```
quo_not_missing(x)
```

Arguments

x Test object

Value

TRUE or error.

Author(s)

Thomas Neitmann, Ondrej Slama

See Also

Helpers for working with Quosures: [add_suffix_to_vars\(\)](#), [quo_c\(\)](#), [replace_symbol_in_quo\(\)](#), [replace_values_by_names\(\)](#)

| | |
|-----------------|---|
| remove_tmp_vars | <i>Remove All Temporary Variables Created Within the Current Function Environment</i> |
|-----------------|---|

Description

Remove All Temporary Variables Created Within the Current Function Environment

Usage

```
remove_tmp_vars(dataset)
```

Arguments

| | |
|---------|-------------------|
| dataset | The input dataset |
|---------|-------------------|

Value

The input dataset with temporary variables removed

See Also

[get_new_tmp_var\(\)](#)

Examples

```
library(dplyr, warn.conflicts = FALSE)
library(admiral.test)
data(admiral_dm)
dm <- select(admiral_dm, USUBJID)
tmp_var <- get_new_tmp_var(dm)
dm <- mutate(dm, !!tmp_var := NA)

## This function creates two new temporary variables which are removed when calling
## `remove_tmp_vars()`. Note that any temporary variable created outside this
## function is not removed
do_something <- function(dataset) {
  tmp_var_1 <- get_new_tmp_var(dm)
  tmp_var_2 <- get_new_tmp_var(dm)
  dm %>%
    mutate(!!tmp_var_1 := NA, !!tmp_var_2 := NA) %>%
```

```
    print() %>%  
    remove_tmp_vars()  
  }  
  
do_something(dm)
```

replace_symbol_in_quo *Replace Symbols in a Quosure*

Description

Replace symbols in a quosure

Usage

```
replace_symbol_in_quo(quosure, target, replace)
```

Arguments

| | |
|---------|------------------|
| quosure | Quosure |
| target | Target symbol |
| replace | Replacing symbol |

Value

The quosure where every occurrence of the symbol `target` is replaced by `replace`

Author(s)

Stefan Bundfuss

See Also

Helpers for working with Quosures: [add_suffix_to_vars\(\)](#), [quo_c\(\)](#), [quo_not_missing\(\)](#), [replace_values_by_names\(\)](#)

Examples

```
library(rlang)  
  
replace_symbol_in_quo(quo(AVAL), target = AVAL, replace = AVAL.join)  
replace_symbol_in_quo(quo(AVALC), target = AVAL, replace = AVAL.join)  
replace_symbol_in_quo(quo(desc(AVAL)), target = AVAL, replace = AVAL.join)
```

`replace_values_by_names`*Replace Quosure Value with Name*

Description

Replace Quosure Value with Name

Usage

```
replace_values_by_names(quosures)
```

Arguments

`quosures` A list of quosures

Value

A list of quosures

Author(s)

Thomas Neitmann

See Also

Helpers for working with Quosures: [add_suffix_to_vars\(\)](#), [quo_c\(\)](#), [quo_not_missing\(\)](#), [replace_symbol_in_quo\(\)](#)

`squote`*Wrap a String in Single Quotes*

Description

Wrap a String in Single Quotes

Usage

```
squote(x)
```

Arguments

`x` A character vector

Value

A character vector

Author(s)

Thomas Neitmann

See Also

Helpers for working with Quotes and Quoting: [backquote\(\)](#), [dquote\(\)](#), [enumerate\(\)](#)

suppress_warning *Suppress Specific Warnings*

Description

Suppress certain warnings issued by an expression.

Usage

```
suppress_warning(expr, regexpr)
```

Arguments

| | |
|---------|--|
| expr | Expression to be executed |
| regexpr | Regular expression matching warnings to suppress |

Details

All warnings which are issued by the expression and match the regular expression are suppressed.

Value

Return value of the expression

Author(s)

- Thomas Neitmann
- Stefan Bundfuss

See Also

Function that provide users with custom warnings [warn_if_incomplete_dtc\(\)](#), [warn_if_inconsistent_list\(\)](#), [warn_if_invalid_dtc\(\)](#), [warn_if_vars_exist\(\)](#)

| | |
|------------------|-------------------------|
| valid_time_units | <i>Valid Time Units</i> |
|------------------|-------------------------|

Description

Contains the acceptable character vector of valid time units

Usage

```
valid_time_units()
```

Value

A character vector of valid time units

See Also

Developer Utility Functions: [%notin%\(\)](#), [%or%\(\)](#), [arg_name\(\)](#), [as_name\(\)](#), [contains_vars\(\)](#), [convert_dtm_to_dtc\(\)](#), [extract_vars\(\)](#), [filter_if\(\)](#), [vars2chr\(\)](#)

| | |
|----------|--|
| vars2chr | <i>Turn a List of Quosures into a Character Vector</i> |
|----------|--|

Description

Turn a List of Quosures into a Character Vector

Usage

```
vars2chr(quosures)
```

Arguments

quosures A list of quosures created using [vars\(\)](#)

Value

A character vector

Author(s)

Thomas Neitmann

See Also

Developer Utility Functions: [%notin%\(\)](#), [%or%\(\)](#), [arg_name\(\)](#), [as_name\(\)](#), [contains_vars\(\)](#), [convert_dtm_to_dtc\(\)](#), [extract_vars\(\)](#), [filter_if\(\)](#), [valid_time_units\(\)](#)

Examples

```
library(dplyr, warn.conflicts = FALSE)

vars2chr(vars(USUBJID, AVAL))
```

```
warn_if_incomplete_dtc
```

Warn if incomplete dtc

Description

Warn if incomplete dtc

Usage

```
warn_if_incomplete_dtc(dtc, n)
```

Arguments

| | |
|-----|---|
| dtc | A character vector of date-times in ISO 8601 format |
| n | A non-negative integer |

Value

A warning if dtc contains any partial dates

See Also

Function that provide users with custom warnings [suppress_warning\(\)](#), [warn_if_inconsistent_list\(\)](#), [warn_if_invalid_dtc\(\)](#), [warn_if_vars_exist\(\)](#)

```
warn_if_inconsistent_list
```

Warn If Two Lists are Inconsistent

Description

Checks if two list inputs have the same names and same number of elements and issues a warning otherwise.

Usage

```
warn_if_inconsistent_list(base, compare, list_name, i = 2)
```

Arguments

| | |
|-----------|-------------------------------------|
| base | A named list |
| compare | A named list |
| list_name | A string the name of the list |
| i | the index id to compare the 2 lists |

Value

a warning if the 2 lists have different names or length

Author(s)

Samia Kabi

See Also

Function that provide users with custom warnings [suppress_warning\(\)](#), [warn_if_incomplete_dtc\(\)](#), [warn_if_invalid_dtc\(\)](#), [warn_if_vars_exist\(\)](#)

Examples

```
library(dplyr, warn.conflicts = FALSE)

# no warning
warn_if_inconsistent_list(
  base = vars(DTHDOM = "DM", DTHSEQ = DMSEQ),
  compare = vars(DTHDOM = "DM", DTHSEQ = DMSEQ),
  list_name = "Test"
)
# warning
warn_if_inconsistent_list(
  base = vars(DTHDOM = "DM", DTHSEQ = DMSEQ, DTHVAR = "text"),
  compare = vars(DTHDOM = "DM", DTHSEQ = DMSEQ),
  list_name = "Test"
)
```

warn_if_invalid_dtc *Warn If a Vector Contains Unknown Datetime Format*

Description

Warn if the vector contains unknown datetime format such as "2003-12-15T-15:18", "2003-12-15T13:-:19", "-12-15", "—T07:15"

Usage

```
warn_if_invalid_dtc(dtc, is_valid = is_valid_dtc(dtc))
```

Arguments

`dtc` a character vector containing the dates
`is_valid` a logical vector indicating whether elements in `dtc` are valid

Value

No return value, called for side effects

Author(s)

Samia Kabi

See Also

Function that provide users with custom warnings [suppress_warning\(\)](#), [warn_if_incomplete_dtc\(\)](#), [warn_if_inconsistent_list\(\)](#), [warn_if_vars_exist\(\)](#)

Examples

```
## No warning as `dtc` is a valid date format
warn_if_invalid_dtc(dtc = "2021-04-06")

## Issues a warning
warn_if_invalid_dtc(dtc = "2021-04-06T-:30:30")
```

`warn_if_vars_exist` *Warn If a Variable Already Exists*

Description

Warn if a variable already exists inside a dataset

Usage

```
warn_if_vars_exist(dataset, vars)
```

Arguments

`dataset` A data.frame
`vars` character vector of columns to check for in dataset

Value

No return value, called for side effects

Author(s)

Thomas Neitmann

See Also

Function that provide users with custom warnings `suppress_warning()`, `warn_if_incomplete_dtc()`, `warn_if_inconsistent_list()`, `warn_if_invalid_dtc()`

Examples

```
library(admiral.test)
data(admiral_dm)

## No warning as `AAGE` doesn't exist in `dm`
warn_if_vars_exist(admiral_dm, "AAGE")

## Issues a warning
warn_if_vars_exist(admiral_dm, "ARM")
```

what_is_it

What Kind of Object is This?

Description

Returns a string describing what kind of object the input is.

Usage

```
what_is_it(x)
```

Arguments

x Any R object

Value

A character description of the type of x

Author(s)

Thomas Neitmann

Examples

```
what_is_it("abc")
what_is_it(1L)
what_is_it(1:10)
what_is_it(mtcars)
```

`%notin%`*Negated Value Matching*

Description

Returns a logical vector indicating if there is *no* match of the left operand in the right operand.

Usage

```
x %notin% table
```

Arguments

| | |
|-------|----------------------------------|
| x | The values to be matched |
| table | The values to be matched against |

Value

A logical vector

Author(s)

Thomas Neitmann

See Also

Developer Utility Functions: [%or%](#)(), [arg_name\(\)](#), [as_name\(\)](#), [contains_vars\(\)](#), [convert_dtm_to_dtc\(\)](#), [extract_vars\(\)](#), [filter_if\(\)](#), [valid_time_units\(\)](#), [vars2chr\(\)](#)

`%or%`*Or*

Description

Or

Usage

```
lhs %or% rhs
```

Arguments

| | |
|-----|------------------------|
| lhs | Any valid R expression |
| rhs | Any valid R expression |

Details

The function evaluates the expression lhs and if this expression results in an error, it catches that error and proceeds with evaluating the expression rhs and returns that result.

Value

Either the result of evaluating lhs, rhs or an error

See Also

Developer Utility Functions: [%notin%](#)(), [arg_name\(\)](#), [as_name\(\)](#), [contains_vars\(\)](#), [convert_dtm_to_dtc\(\)](#), [extract_vars\(\)](#), [filter_if\(\)](#), [valid_time_units\(\)](#), [vars2chr\(\)](#)

Index

- * **assertion**
 - assert_atomic_vector, 6
 - assert_character_scalar, 7
 - assert_character_vector, 8
 - assert_data_frame, 9
 - assert_date_var, 10
 - assert_date_vector, 12
 - assert_expr, 13
 - assert_filter_cond, 14
 - assert_function, 15
 - assert_function_param, 16
 - assert_has_variables, 17
 - assert_integer_scalar, 18
 - assert_list_element, 19
 - assert_list_of, 20
 - assert_logical_scalar, 21
 - assert_named_exprs, 22
 - assert_numeric_vector, 23
 - assert_one_to_one, 24
 - assert_order_vars, 25
 - assert_param_does_not_exist, 26
 - assert_s3_class, 27
 - assert_same_type, 28
 - assert_symbol, 29
 - assert_unit, 30
 - assert_vars, 31
 - assert_varval_list, 32
- * **datasets**
 - get_dataset, 41
- * **dev_utility**
 - %notin%, 56
 - %or%, 56
 - arg_name, 5
 - as_name, 34
 - contains_vars, 35
 - convert_dtm_to_dtc, 35
 - dataset_vignette, 36
 - extract_vars, 39
 - filter_if, 40
 - valid_time_units, 51
 - vars2chr, 51
- * **get**
 - get_constant_vars, 41
 - get_duplicates, 42
 - get_source_vars, 43
- * **is**
 - is_auto, 44
 - is_named, 44
 - is_order_vars, 45
 - is_valid_dtc, 45
- * **joins**
 - anti_join, 4
- * **quote**
 - backquote, 34
 - dquote, 37
 - enumerate, 37
 - squote, 49
- * **quo**
 - add_suffix_to_vars, 3
 - quo_c, 46
 - quo_not_missing, 46
 - replace_symbol_in_quo, 48
 - replace_values_by_names, 49
- * **test_helper**
 - expect_dfs_equal, 38
- * **tmp_vars**
 - get_new_tmp_var, 42
 - remove_tmp_vars, 47
- * **warnings**
 - suppress_warning, 50
 - warn_if_incomplete_dtc, 52
 - warn_if_inconsistent_list, 52
 - warn_if_invalid_dtc, 53
 - warn_if_vars_exist, 54
- * **what**
 - what_is_it, 55
 - %notin%, 5, 34–36, 40, 51, 56, 57
 - %or%, 5, 34–36, 40, 51, 56, 56

- add_suffix_to_vars, 3, 46–49
- anti_join, 4
- arg_name, 5, 34–36, 40, 51, 56, 57
- as_name, 5, 34, 35, 36, 40, 51, 56, 57
- assert_atomic_vector, 6, 7, 9, 10, 12–18, 20–28, 30–33
- assert_character_scalar, 6, 7, 9, 10, 12–18, 20–28, 30–33
- assert_character_vector, 6, 7, 8, 10, 12–18, 20–28, 30–33
- assert_data_frame, 6, 7, 9, 9, 12–18, 20–28, 30–33
- assert_date_var, 10
- assert_date_vector, 6, 7, 9, 10, 12, 13–18, 20–28, 30–33
- assert_expr, 6, 7, 9, 10, 12, 13, 14–18, 20–28, 30–33
- assert_filter_cond, 6, 7, 9, 10, 12, 13, 14, 15–18, 20–28, 30–33
- assert_function, 6, 7, 9, 10, 12–14, 15, 16–18, 20–28, 30–33
- assert_function_param, 6, 7, 9, 10, 12–15, 16, 17, 18, 20–28, 30–33
- assert_has_variables, 6, 7, 9, 10, 12–16, 17, 18, 20–28, 30–33
- assert_integer_scalar, 6, 7, 9, 10, 12–17, 18, 20–28, 30–33
- assert_list_element, 6, 7, 9, 10, 12–18, 19, 20–28, 30–33
- assert_list_of, 6, 7, 9, 10, 12–18, 20, 20, 21–28, 30–33
- assert_logical_scalar, 6, 7, 9, 10, 12–18, 20, 21, 22–28, 30–33
- assert_named_exprs, 6, 7, 9, 10, 12–18, 20, 21, 22, 23–28, 30–33
- assert_numeric_vector, 6, 7, 9, 10, 12–18, 20–22, 23, 24–28, 30–33
- assert_one_to_one, 6, 7, 9, 10, 12–18, 20–23, 24, 25–28, 30–33
- assert_order_vars, 6, 7, 9, 10, 12–18, 20–24, 25, 26–28, 30–33
- assert_param_does_not_exist, 6, 7, 9, 10, 12–18, 20–25, 26, 27, 28, 30–33
- assert_s3_class, 6, 7, 9, 10, 12–18, 20–26, 27, 28, 30–33
- assert_same_type, 6, 7, 9, 10, 12–18, 20–27, 28, 30–33
- assert_symbol, 6, 7, 9, 10, 12–18, 20–28, 29, 31–33
- assert_unit, 6, 7, 9, 10, 12–18, 20–28, 30, 30, 32, 33
- assert_vars, 6, 7, 9, 10, 12–18, 20–28, 30, 31, 31, 33
- assert_varval_list, 6, 7, 9, 10, 12–18, 20–28, 30–32, 32
- backquote, 34, 37, 38, 50
- contains_vars, 5, 34, 35, 36, 40, 51, 56, 57
- convert_dtm_to_dtc, 5, 34, 35, 35, 40, 51, 56, 57
- dataset_vignette, 36
- diffdf::diffdf(), 38
- dquote, 35, 37, 38, 50
- enumerate, 35, 37, 37, 50
- expect_dfs_equal, 38
- extract_vars, 5, 34–36, 39, 40, 51, 56, 57
- filter_if, 5, 34–36, 40, 40, 51, 56, 57
- get_constant_vars, 41, 42, 43
- get_dataset, 41
- get_duplicates, 41, 42, 43
- get_new_tmp_var, 42
- get_new_tmp_var(), 47
- get_source_vars, 41, 42, 43
- inner_join(anti_join), 4
- is_auto, 44, 44, 45
- is_named, 44, 44, 45
- is_order_vars, 44, 45, 45
- is_valid_dtc, 44, 45, 45
- left_join(anti_join), 4
- quo_c, 4, 46, 47–49
- quo_not_missing, 4, 46, 46, 48, 49
- remove_tmp_vars, 47
- remove_tmp_vars(), 43
- replace_symbol_in_quo, 4, 46, 47, 48, 49
- replace_values_by_names, 4, 46–48, 49
- squote, 35, 37, 38, 49
- suppress_warning, 50, 52–55

`valid_time_units`, [5](#), [34–36](#), [40](#), [51](#), [51](#), [56](#),
[57](#)
`vars()`, [51](#)
`vars2chr`, [5](#), [34–36](#), [40](#), [51](#), [51](#), [56](#), [57](#)
`warn_if_incomplete_dtc`, [50](#), [52](#), [53–55](#)
`warn_if_inconsistent_list`, [50](#), [52](#), [52](#), [54](#),
[55](#)
`warn_if_invalid_dtc`, [50](#), [52](#), [53](#), [53](#), [55](#)
`warn_if_vars_exist`, [50](#), [52–54](#), [54](#)
`what_is_it`, [55](#)