Package 'admiraldev'

August 26, 2022

Type Package

Title Development Tools for the Admiral Package Family

Version 0.1.0

Description Utility functions to check data, variables and conditions for functions used in 'admiral' and 'admiral' extension packages. Additional utility helper functions to 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 7.2.0

Depends R (>= 3.5)

Imports assertthat, dplyr, lubridate, magrittr, purrr, rlang, stringr, hms, tidyr, tidyselect, lifecycle

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],
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 2022-08-26 08:12:34 UTC

R topics documented:

`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 *Extract Argument Name from an Expression*

Description

Extract Argument Name from an Expression

Usage

```
arg_name(expr)
```

Arguments

expr	An expression created inside a function using substitute()

Value

character vector

Author(s)

Thomas Neitmann, Ondrej Slama

See Also

Developer Utility Functions: %notin%(), %or%(), as_name(), convert_dtm_to_dtc(), extract_vars(), filter_if(), negate_vars(), replace_values_by_names(), valid_time_units(), vars2chr()

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

`arg`	A function argument to be checked
`values`	A `character` vector of valid values for `arg`
`case_sensitive`	Should the argument be handled case-sensitive? If set to `FALSE`, the argument is converted to lower case for checking the permitted values and returning the argument.
`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 `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_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_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_character_scalar(), 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_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 is 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 vars()
check_is_grouped	Throw an error is 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_character_scalar(), assert_character_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

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

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

example_fun(admiral_dm)

try(example_fun(dplyr::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_expr` *Assert Argument is an Expression*

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_character_scalar()`, `assert_character_vector()`, `assert_data_frame()`, `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()`

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_character_scalar(), assert_character_vector(), assert_data_frame(), 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_symbol(), assert_unit(), assert_vars(), assert_varval_list()

Examples

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

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

```
example_fun(admiral_dm, AGE == 64)

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

`assert_function` *Is Argument a Function?*

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_character_scalar()`, `assert_character_vector()`, `assert_data_frame()`, `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_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_character_scalar()`, `assert_character_vector()`, `assert_data_frame()`, `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_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_character_scalar(), assert_character_vector(), assert_data_frame(), 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_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_character_scalar(), assert_character_vector(), assert_data_frame(), 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_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., `censor == 0` to check the `censor` 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 censor 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_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_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()

---

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_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_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(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_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_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(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_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_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()`

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_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_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(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_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_order_vars(), assert_param_does_not_exist(), assert_s3_class(), 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_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_param_does_not_exist(), assert_s3_class(), assert_symbol(), assert_unit(), assert_vars(), assert_varval_list()

Examples

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

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

try(example_fun(rlang::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_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_s3_class(), 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"))
```

---

assert_s3_class *Is an Argument an Object of a Specific S3 Class?*

Description

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

Usage

```
assert_s3_class(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 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_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_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_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_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_unit(), assert_vars(), assert_varval_list()

Examples

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

example_fun <- function(dat, var) {
  var <- assert_symbol(rlang::enquo(var))
  dplyr::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_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_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)
```

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 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_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_symbol(), assert_unit(), assert_varval_list()

Examples

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

example_fun(vars(USUBJID, PARAMCD))

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

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

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

---

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 it returned invisibly.

Author(s)

Stefan Bundfuss, Thomas Neitmann

See Also

Checks for valid input and returns warning or errors messages: 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_symbol(), assert_unit(), assert_vars()

Examples

```
example_fun <- function(vars) {
  assert_varval_list(vars)
}
example_fun(vars(DTHDOM = "AE", DTHSEQ = AESEQ))

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

as_name *Turn a Quosure into a String*

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(), convert_dtm_to_dtc(), extract_vars(), filter_if(), negate_vars(), replace_values_by_names(), valid_time_units(), vars2chr()

backquote *Wrap a String in Backquotes*

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()

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

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(), extract_vars(), filter_if(), negate_vars(), replace_values_by_names(), valid_time_units(), vars2chr()

dataset_vignette *Output a Dataset in a Vignette in the admiral Format*

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 theChoose 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 *Wrap a String in Double Quotes*

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 *Enumerate Multiple Strings*

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

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

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(), convert_dtm_to_dtc(), filter_if(), negate_vars(), replace_values_by_names(), 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(), convert_dtm_to_dtc(), extract_vars(), negate_vars(), replace_values_by_names(), 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). *Permitted Values:* 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* `.datasets` *environment*

Description

Retrieve a Dataset from the `.datasets` environment

Usage

```
get_dataset(name)
```

Arguments

name	The name of the dataset to retrieve

Value

A `data.frame`

Author(s)

Thomas Neitmann

See Also

Other datasets: `set_dataset()`

`get_duplicates` *Get Duplicates From a Vector*

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 Also

Brings 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 *Get a New Temporary Variable Name for a Dataset*

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 `datset` 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 *Checks if the argument equals the auto keyword*

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_date(), is_named(), is_order_vars(), is_timeunit(), is_valid_date_entry(), is_valid_day(), is_valid_dtc(), is_valid_hour(), is_valid_month(), is_valid_sec_min(), is_valid_time_entry()

---

is_date *Is Date/Date-time?*

Description

Checks if a date or date-time vector was specified

Usage

```
is_date(arg)
```

Arguments

arg	The argument to check

Value

TRUE if the argument is a date or date-time, FALSE otherwise

Author(s)

Stefan Bundfuss

See Also

Identifies type of Object with return of TRUE/FALSE: is_auto(), is_named(), is_order_vars(), is_timeunit(), is_valid_date_entry(), is_valid_day(), is_valid_dtc(), is_valid_hour(), is_valid_month(), is_valid_sec_min(), is_valid_time_entry()

is_named *Is a named argument*

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_date(), is_order_vars(), is_timeunit(), is_valid_date_entry(), is_valid_day(), is_valid_dtc(), is_valid_hour(), is_valid_month(), is_valid_sec_min(), is_valid_time_entry()

is_order_vars *Is order vars?*

Description

Is order vars?

Usage

```
is_order_vars(arg)
```

Arguments

arg	A<n R object

Value

A logical vector

See Also

Identifies type of Object with return of TRUE/FALSE: `is_auto()`, `is_date()`, `is_named()`, `is_timeunit()`, `is_valid_date_entry()`, `is_valid_day()`, `is_valid_dtc()`, `is_valid_hour()`, `is_valid_month()`, `is_valid_sec_min()`, `is_valid_time_entry()`

`is_timeunit` *Is Time Unit?*

Description

Checks if a string is a time unit, i.e., 'years', 'months', 'days', 'hours', 'minutes', or 'seconds'.

Usage

```
is_timeunit(arg)
```

Arguments

`arg`	The argument to check

Value

`TRUE` if the argument is a time unit, `FALSE` otherwise

Author(s)

Stefan Bundfuss

See Also

Identifies type of Object with return of TRUE/FALSE: `is_auto()`, `is_date()`, `is_named()`, `is_order_vars()`, `is_valid_date_entry()`, `is_valid_day()`, `is_valid_dtc()`, `is_valid_hour()`, `is_valid_month()`, `is_valid_sec_min()`, `is_valid_time_entry()`

`is_valid_date_entry` *Check Validity of the Date Imputation Input*

Description

Date_imputation format should be specified as "dd-mm" (e.g. "01-01") or as a keyword: "FIRST", "MID", "LAST"

Usage

```
is_valid_date_entry(arg)
```

Arguments

arg	The argument to check

Value

TRUE if the argument is a valid date_imputation input, FALSE otherwise

Author(s)

Samia Kabi

See Also

Identifies type of Object with return of TRUE/FALSE: is_auto(), is_date(), is_named(), is_order_vars(), is_timeunit(), is_valid_day(), is_valid_dtc(), is_valid_hour(), is_valid_month(), is_valid_sec_min(), is_valid_time_entry()

---

is_valid_day *Check Validity of the Day Portion in the Date Input*

Description

Days are expected to range from 1 to 31

Usage

```
is_valid_day(arg)
```

Arguments

arg	The argument to check

Value

TRUE if the argument is a day input, FALSE otherwise

Author(s)

Samia Kabi

See Also

Identifies type of Object with return of TRUE/FALSE: is_auto(), is_date(), is_named(), is_order_vars(), is_timeunit(), is_valid_date_entry(), is_valid_dtc(), is_valid_hour(), is_valid_month(), is_valid_sec_min(), is_valid_time_entry()

is_valid_dtc *Is this string a valid DTC*

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_date(), is_named(), is_order_vars(), is_timeunit(), is_valid_date_entry(), is_valid_day(), is_valid_hour(), is_valid_month(), is_valid_sec_min(), is_valid_time_entry()

is_valid_hour *Check Validity of the Hour Portion in the Time Input*

Description

Hours are expected to range from 0 to 23

Usage

```
is_valid_hour(arg)
```

Arguments

arg	The argument to check

Value

TRUE if the argument is a valid hour input, FALSE otherwise

Author(s)

Samia Kabi

See Also

Identifies type of Object with return of TRUE/FALSE: is_auto(), is_date(), is_named(), is_order_vars(), is_timeunit(), is_valid_date_entry(), is_valid_day(), is_valid_dtc(), is_valid_month(), is_valid_sec_min(), is_valid_time_entry()

---

is_valid_month *Check Validity of the Month Portion in the Date Input*

Description

Days are expected to range from 1 to 12

Usage

```
is_valid_month(arg)
```

Arguments

arg	The argument to check

Value

TRUE if the argument is a month input, FALSE otherwise

Author(s)

Samia Kabi

See Also

Identifies type of Object with return of TRUE/FALSE: is_auto(), is_date(), is_named(), is_order_vars(), is_timeunit(), is_valid_date_entry(), is_valid_day(), is_valid_dtc(), is_valid_hour(), is_valid_sec_min(), is_valid_time_entry()

---

is_valid_sec_min *Check Validity of the Minute/Second Portion of the Time Input*

Description

Minutes and seconds are expected to range from 0 to 59

Usage

```
is_valid_sec_min(arg)
```

Arguments

arg	The argument to check

Value

TRUE if the argument is a valid min/sec input, FALSE otherwise

Author(s)

Samia Kabi

See Also

Identifies type of Object with return of TRUE/FALSE: is_auto(), is_date(), is_named(), is_order_vars(), is_timeunit(), is_valid_date_entry(), is_valid_day(), is_valid_dtc(), is_valid_hour(), is_valid_month(), is_valid_time_entry()

---

is_valid_time_entry *Check Validity of the Time Imputation Input*

Description

Time_imputation format should be specified as "hh:mm:ss" (e.g. "00:00:00") or as a keyword: "FIRST", "LAST"

Usage

```
is_valid_time_entry(arg)
```

Arguments

arg	The argument to check

Value

TRUE if the argument is a valid time_imputation input, FALSE otherwise

Author(s)

Samia Kabi

See Also

Identifies type of Object with return of TRUE/FALSE: is_auto(), is_date(), is_named(), is_order_vars(), is_timeunit(), is_valid_date_entry(), is_valid_day(), is_valid_dtc(), is_valid_hour(), is_valid_month(), is_valid_sec_min()

negate_vars *Negate List of Variables*

Description

The function adds a minus sign as prefix to each variable.

Usage

```
negate_vars(vars = NULL)
```

Arguments

vars	List of variables created by vars()

Details

This is useful if a list of variables should be removed from a dataset, e.g., `select(!!!negate_vars(by_vars))` removes all by variables.

Value

A list of quosures

Author(s)

Stefan Bundfuss

See Also

Developer Utility Functions: %notin%(), %or%(), arg_name(), as_name(), convert_dtm_to_dtc(), extract_vars(), filter_if(), replace_values_by_names(), valid_time_units(), vars2chr()

Examples

```
negate_vars(vars(USUBJID, STUDYID))
```

quo_c *Concatenate One or More Quosure(s)*

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: quo_not_missing()

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

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: quo_c()

`remove_tmp_vars` *Remove All Temporary Variables Created Within the Current Function Environment*

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)
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_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

Developer Utility Functions: `%notin%()`, `%or%()`, `arg_name()`, `as_name()`, `convert_dtm_to_dtc()`, `extract_vars()`, `filter_if()`, `negate_vars()`, `valid_time_units()`, `vars2chr()`

`set_dataset` *Set a Dataset in the* `.datasets` *environment*

Description

Set a Dataset in the `.datasets` environment

Usage

```
set_dataset(dataset, name)
```

Arguments

dataset	A data.frame
name	A name for dataset

Details

The object passed to the dataset argument will be assigned to name in the .datasets environment. It can be retrieved later on using get_dataset()

Value

No return value, called for side effects

Author(s)

Thomas Neitmann

See Also

Other datasets: get_dataset()

---

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 *Valid Time Units*

Description

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()`, `convert_dtm_to_dtc()`, `extract_vars()`, `filter_if()`, `negate_vars()`, `replace_values_by_names()`, `vars2chr()`

`vars2chr` *Turn a List of Quosures into a Character Vector*

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()`, `convert_dtm_to_dtc()`, `extract_vars()`, `filter_if()`, `negate_vars()`, `replace_values_by_names()`, `valid_time_units()`

Examples

```
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

```
# 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()`, `convert_dtm_to_dtc()`, `extract_vars()`, `filter_if()`, `negate_vars()`, `replace_values_by_names()`, `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()`, `convert_dtm_to_dtc()`, `extract_vars()`, `filter_if()`, `negate_vars()`, `replace_values_by_names()`, `valid_time_units()`, `vars2chr()`

Index