

Package ‘chk’

September 3, 2026

Title Check User-Supplied Function Arguments

Version 0.11.0

Description For developers to check user-supplied function arguments. It is designed to be simple, fast and customizable. Error messages follow the tidyverse style guide.

License MIT + file LICENSE

URL <https://github.com/poissonconsulting/chk>,
<https://poissonconsulting.github.io/chk/>

BugReports <https://github.com/poissonconsulting/chk/issues/>

Depends R (>= 4.1)

Imports lifecycle, methods, rlang, tools

Suggests covr, knitr, R6, rmarkdown, S7, testthat (>= 3.0.0), withr

VignetteBuilder knitr

RdMacros lifecycle

Config/gha/extra-packages testthat=?ignore-before-r=4.1.0,
rmarkdown=?ignore-before-r=4.1.0

Config/Needs/website poissonconsulting/poissontemplate

Config/roxygen2/version 8.1.0

Config/testthat/edition 3

Encoding UTF-8

Language en-US

NeedsCompilation no

Author Joe Thorley [aut, cre] (ORCID: <<https://orcid.org/0000-0002-7683-4592>>),
Kirill Müller [aut] (ORCID: <<https://orcid.org/0000-0002-1416-3412>>),
Ayla Pearson [aut] (ORCID: <<https://orcid.org/0000-0001-7388-1222>>),
Florenca D'Andrea [ctb],
Nadine Hussein [ctb] (ORCID: <<https://orcid.org/0000-0003-4470-8361>>),
Evan Amies-Galonski [ctb] (ORCID:
<<https://orcid.org/0000-0003-1096-2089>>),
Stefano Mezzini [ctb] (ORCID: <<https://orcid.org/0000-0001-8551-7436>>),
Poisson Consulting [cph, fnd]

Maintainer Joe Thorley <joe@poissonconsulting.ca>

Repository CRAN

Date/Publication 2026-09-03 05:20:50 UTC

Contents

abort_chk	4
cc	5
check_data	6
check_dim	7
check_dirs	8
check_files	9
check_key	9
check_length	10
check_names	11
check_values	12
chkor	13
chkor_vld	13
chk_all	14
chk_all_equal	15
chk_all_equivalent	16
chk_all_identical	17
chk_all_na	18
chk_array	19
chk_atomic	20
chk_character	21
chk_character_or_factor	22
chk_chr	24
chk_class	25
chk_compatible_lengths	26
chk_complex	27
chk_complex_number	29
chk_count	30
chk_data	31
chk_date	32
chk_date_time	33
chk_dbl	34
chk_dir	35
chk_double	36
chk_environment	37
chk_equal	38
chk_equivalent	39
chk_ext	40
chk_factor	41
chk_false	42
chk_file	43
chk_flag	44

chk_function	45
chk_gt	46
chk_gte	47
chk_identical	48
chk_integer	49
chk_is	50
chk_join	51
chk_length	52
chk_lgl	53
chk_list	54
chk_logical	55
chk_lt	57
chk_lte	58
chk_match	59
chk_matrix	60
chk_missing	61
chk_named	62
chk_not_any_na	63
chk_not_empty	64
chk_not_missing	65
chk_not_null	66
chk_not_subset	67
chk_null	68
chk_null_or	69
chk_number	70
chk_numeric	71
chk_orderset	72
chk_r6_class	73
chk_range	74
chk_raw	75
chk_s3_class	76
chk_s3_class_strict	77
chk_s4_class	78
chk_scalar	79
chk_setequal	80
chk_sorted	81
chk_string	82
chk_subset	83
chk_superset	84
chk_true	85
chk_tz	86
chk_unique	87
chk_unused	89
chk_used	90
chk_valid_name	91
chk_vector	92
chk_whole_number	93
chk_whole_numeric	94

chk_wnum 95
deparse_backtick_chk 96
err 97
expect_chk_error 98
generate_check_data 100
message_chk 101
p 102

Index 104

abort_chk	<i>Abort Check</i>
-----------	--------------------

Description

A wrapper on [err\(\)](#) that sets the subclass to be 'chk_error'.

Usage

```
abort_chk(..., n = NULL, tidy = TRUE, call = rlang::caller_call(2))
```

Arguments

...	Multiple objects that are converted to a string using <code>paste0(..., collapse = '')</code> .
n	The value of n for converting <code>sprintf</code> -like types.
tidy	A flag specifying whether capitalize the first character and add a missing period.
call	The execution environment of a currently running function, e.g. <code>call = caller_env()</code> . The corresponding function call is retrieved and mentioned in error messages as the source of the error. You only need to supply <code>call</code> when throwing a condition from a helper function which wouldn't be relevant to mention in the message. Can also be <code>NULL</code> or a defused function call to respectively not display any call or hard-code a code to display. For more information about error calls, see Including function calls in error messages .

Details

It is exported to allow users to easily construct their own `chk_` functions.

Value

Throws an error of class 'chk_error'.

See Also

[err\(\)](#)

Examples

```
try(abort_chk("x must be NULL"))
try(abort_chk("`x` must be NULL"))
try(abort_chk("there %r %n problem value%s", n = 1))
try(abort_chk("there %r %n problem value%s", n = 1.5))
```

cc

Concatenate with Commas

Description

Concatenates object values into a string with each value separated by a comma and the last value separated by a conjunction.

Usage

```
cc(
  x,
  conj = ", ",
  sep = ", ",
  brac = if (is.character(x) || is.factor(x)) "'" else "",
  ellipsis = 10L,
  chk = TRUE
)
```

Arguments

x	The object to concatenate.
conj	A string of the conjunction to separate the last value by.
sep	A string of the separator.
brac	A string to brace the values by.
ellipsis	A numeric scalar of the maximum number of values to display before using an ellipsis.
chk	A flag specifying whether to check the other parameters.

Details

By default, if x has more than 10 values an ellipsis is used to ensure only 10 values are displayed (including the ellipsis).

Value

A string.

Examples

```
cc(1:2)
cc(1:2, conj = " or")
cc(3:1, brac = "'")
cc(1:11)
cc(as.character(1:2))
```

check_data

Check Data

Description

Checks column names, values, number of rows and key for a data.frame.

Usage

```
check_data(
  x,
  values = NULL,
  exclusive = FALSE,
  order = FALSE,
  nrow = numeric(0),
  key = character(0),
  x_name = NULL
)
```

Arguments

x	The object to check.
values	A uniquely named list of atomic vectors of the column values.
exclusive	A flag specifying whether x must only include columns named in values.
order	A flag specifying whether the order of columns in x must match names in values.
nrow	A flag or a whole numeric vector of the value, value range or possible values.
key	A character vector of the columns that represent a unique key.
x_name	A string of the name of object x or NULL.

Details

The values of each column are checked using `check_values()`. See its details for the rules including those for factor levels and missing values.

values is reduced with `unique()`, so a column whose class lacks a correct `unique()` method is checked against the wrong values. In particular `unique()` on a lubridate Period vector discards everything but the seconds, which collapses values and silently changes the check applied (see <https://github.com/tidyverse/lubridate/issues/1154>). Either define `unique.Period()` in your own code or convert the column and values to a class with a correct `unique()` method, such as the number of seconds with `as.numeric(lubridate::as.duration(x))`.

Value

An informative error if the test fails or an invisible copy of x.

See Also

Other check: [check_dim\(\)](#), [check_dirs\(\)](#), [check_files\(\)](#), [check_key\(\)](#), [check_length\(\)](#), [check_names\(\)](#), [check_values\(\)](#)

Examples

```
check_data(data.frame())
check_data(data.frame(x = 2), list(x = 1))
try(check_data(data.frame(x = 2), list(y = 1L)))
try(check_data(data.frame(x = 2), list(y = 1)))
try(check_data(data.frame(x = 2), nrow = 2))
```

check_dim	<i>Check Dimension</i>
-----------	------------------------

Description

Checks dimension of an object.

Usage

```
check_dim(x, dim = length, values = numeric(0), x_name = NULL, dim_name = NULL)
```

Arguments

- x The object to check.
- dim A function returning a non-negative whole number of the dimension.
- values A flag or a whole numeric vector of the value, value range or possible values.
- x_name A string of the name of object x or NULL.
- dim_name A string of the name of the dim function.

Value

An informative error if the test fails or an invisible copy of x.

See Also

Other check: [check_data\(\)](#), [check_dirs\(\)](#), [check_files\(\)](#), [check_key\(\)](#), [check_length\(\)](#), [check_names\(\)](#), [check_values\(\)](#)

Examples

```
check_dim(1)
try(check_dim(1, values = FALSE))
try(check_dim(1, values = c(10, 2)))
try(check_dim(data.frame(x = 1), dim = nrow, values = c(10, 10, 2)))
```

check_dirs

Check Directories Exist

Description

Checks if all directories exist (or if exists = FALSE do not exist as directories or files).

Usage

```
check_dirs(x, exists = TRUE, x_name = NULL)
```

Arguments

x	The object to check.
exists	A flag specifying whether the files/directories must (or must not) exist.
x_name	A string of the name of object x or NULL.

Value

An informative error if the test fails or an invisible copy of x.

See Also

Other check: [check_data\(\)](#), [check_dim\(\)](#), [check_files\(\)](#), [check_key\(\)](#), [check_length\(\)](#), [check_names\(\)](#), [check_values\(\)](#)

Examples

```
check_dirs(tempdir())
try(check_dirs(tempdir(), exists = FALSE))
```

check_files	<i>Check Files Exist</i>
-------------	--------------------------

Description

Checks if all files exist (or if exists = FALSE do not exist as files or directories).

Usage

```
check_files(x, exists = TRUE, x_name = NULL)
```

Arguments

x	The object to check.
exists	A flag specifying whether the files/directories must (or must not) exist.
x_name	A string of the name of object x or NULL.

Value

An informative error if the test fails or an invisible copy of x.

See Also

Other check: [check_data\(\)](#), [check_dim\(\)](#), [check_dirs\(\)](#), [check_key\(\)](#), [check_length\(\)](#), [check_names\(\)](#), [check_values\(\)](#)

Examples

```
check_files(tempfile("unlikely-that-exists-chk"), exists = FALSE)
try(check_files(tempfile("unlikely-that-exists-chk")))
```

check_key	<i>Check Key</i>
-----------	------------------

Description

Checks if columns have unique rows.

Usage

```
check_key(x, key = character(0), na_distinct = FALSE, x_name = NULL)
```

Arguments

x	The object to check.
key	A character vector of the columns that represent a unique key.
na_distinct	A flag specifying whether missing values should be considered distinct.
x_name	A string of the name of object x or NULL.

Value

An informative error if the test fails or an invisible copy of x.

See Also

Other check: [check_data\(\)](#), [check_dim\(\)](#), [check_dirs\(\)](#), [check_files\(\)](#), [check_length\(\)](#), [check_names\(\)](#), [check_values\(\)](#)

Examples

```
x <- data.frame(x = c(1, 2), y = c(1, 1))
check_key(x)
try(check_key(x, "y"))
```

check_length

Check Length

Description

Checks length of an object.

Usage

```
check_length(x, values = numeric(0), x_name = NULL)
```

Arguments

x	The object to check.
values	A flag or a whole numeric vector of the value, value range or possible values.
x_name	A string of the name of object x or NULL.

Value

An informative error if the test fails or an invisible copy of x.

See Also

Other check: [check_data\(\)](#), [check_dim\(\)](#), [check_dirs\(\)](#), [check_files\(\)](#), [check_key\(\)](#), [check_names\(\)](#), [check_values\(\)](#)

Examples

```
check_length(1)
try(check_length(1, values = FALSE))
try(check_length(1, values = c(10, 2)))
```

check_names

*Check Names***Description**

Checks the names of an object.

Usage

```
check_names(
  x,
  names = character(0),
  exclusive = FALSE,
  order = FALSE,
  x_name = NULL
)
```

Arguments

x	The object to check.
names	A character vector of the required names.
exclusive	A flag specifying whether x must only contain the required names.
order	A flag specifying whether the order of the required names in x must match the order in names.
x_name	A string of the name of object x or NULL.

Value

An informative error if the test fails or an invisible copy of x.

See Also

Other check: [check_data\(\)](#), [check_dim\(\)](#), [check_dirs\(\)](#), [check_files\(\)](#), [check_key\(\)](#), [check_length\(\)](#), [check_values\(\)](#)

Examples

```
x <- c(x = 1, y = 2)
check_names(x, c("y", "x"))
try(check_names(x, c("y", "x"), order = TRUE))
try(check_names(x, "x", exclusive = TRUE))
```

check_values

*Check Values and Class***Description**

Checks values and S3 class of an atomic object.

Usage

```
check_values(x, values, x_name = NULL)
```

Arguments

x	The object to check.
values	An atomic vector specifying the S3 class and possible values.
x_name	A string of the name of object x or NULL.

Details

To check the class simply pass a vector of the desired class.

To check that x does not include missing values pass a single non-missing value (of the correct class).

To allow it to include missing values include a missing value.

To check that it only includes missing values only pass a missing value (of the correct class).

To check the range of the values in x pass two non-missing values (as well as the missing value if required).

To check that x only includes specific values pass three or more non-missing values.

In the case of a factor if values has two or more levels then the levels of x must be identical, including their order, to the levels of values. A factor with less than two levels only checks that x is a factor. Thus `factor(1)` allows any factor without missing values, `factor(c(1, NA))` any factor with or without missing values and `factor(NA)` any factor with all missing values.

Value

An informative error if the test fails or an invisible copy of x.

See Also

Other check: [check_data\(\)](#), [check_dim\(\)](#), [check_dirs\(\)](#), [check_files\(\)](#), [check_key\(\)](#), [check_length\(\)](#), [check_names\(\)](#)

Examples

```
check_values(1, numeric(0))
check_values(1, 2)
try(check_values(1, 1L))
try(check_values(NA_real_, 1))
```

chkor	<i>Check OR</i>
-------	-----------------

Description

The `chkor()` function has been deprecated for the faster `chkor_vld()`.

Usage

```
chkor(...)
```

Arguments

... Multiple `chk_` functions.

Details

[Deprecated]

Value

An informative error if the test fails.

See Also

[chk_null_or\(\)](#)

Examples

```
chkor()  
chkor(chk_flag(TRUE))  
try(chkor(chk_flag(1)))  
try(chkor(chk_flag(1), chk_flag(2)))  
chkor(chk_flag(1), chk_flag(TRUE))
```

chkor_vld	<i>Chk OR</i>
-----------	---------------

Description

Chk OR

Usage

```
chkor_vld(..., x_name = NULL)
```

Arguments

...	Multiple vld_ calls. A common mistake is to pass chk_ calls. chkor_vld() is relatively slow. If at all possible use chk_null_or() or first test using the individual vld_ functions and then call chkor_vld() to generate an informative error message.
x_name	A string of the name of the object to use in the error message or NULL. If a string it is passed to each of the chk_ calls so that the error message refers to x_name rather than the deparsed vld_ arguments.

Value

An informative error if the test fails.

See Also

[chk_null_or\(\)](#)

Examples

```
chkor_vld()
chkor_vld(vld_flag(TRUE))
try(chkor_vld(vld_flag(1)))
try(chkor_vld(vld_flag(1), vld_flag(2)))
chkor_vld(vld_flag(1), vld_flag(TRUE))
try(chkor_vld(vld_flag(1), vld_number(1), x_name = "`arg`"))
```

chk_all

Check All

Description

Checks all elements using
all(vapply(x, chk_fun, TRUE, ...))

Usage

```
chk_all(x, chk_fun, ..., x_name = NULL)

vld_all(x, vld_fun, ...)
```

Arguments

x	The object to check.
chk_fun	A chk_ function.
...	Additional arguments.
x_name	A string of the name of object x or NULL.
vld_fun	A vld_ function.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_all()`: Validate All

See Also

[all\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other `all_`checkers: [chk_all_equal\(\)](#), [chk_all_equivalent\(\)](#), [chk_all_identical\(\)](#)

Examples

```
# chk_all
chk_all(TRUE, chk_lgl)
# FIXME try(chk_all(1, chk_lgl))
chk_all(c(TRUE, NA), chk_lgl)
# vld_all
vld_all(c(TRUE, NA), vld_lgl)
```

chk_all_equal	<i>Check All Equal</i>
---------------	------------------------

Description

Checks all elements in `x` equal using

```
length(x) < 2L || all(vapply(x, vld_equal, TRUE, y = x[[1]], tolerance = tolerance))
```

Usage

```
chk_all_equal(x, tolerance = sqrt(.Machine$double.eps), x_name = NULL)
```

```
vld_all_equal(x, tolerance = sqrt(.Machine$double.eps))
```

Arguments

<code>x</code>	The object to check.
<code>tolerance</code>	A non-negative numeric scalar.
<code>x_name</code>	A string of the name of object <code>x</code> or <code>NULL</code> .

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_all_equal()`: Validate All Equal

See Also

[length\(\)](#)

[vld_equal\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other `equal_checkers`: [chk_all_equivalent\(\)](#), [chk_all_identical\(\)](#), [chk_equal\(\)](#), [chk_equivalent\(\)](#), [chk_identical\(\)](#)

Other `all_checkers`: [chk_all\(\)](#), [chk_all_equivalent\(\)](#), [chk_all_identical\(\)](#)

Examples

```
# chk_all_equal
chk_all_equal(c(1, 1.00000001))
try(chk_all_equal(c(1, 1.0000001)))
chk_all_equal(list(c(x = 1), c(x = 1)))
try(chk_all_equal(list(c(x = 1), c(y = 1))))
# vld_all_equal
vld_all_equal(c(1, 1L))
```

<code>chk_all_equivalent</code>	<i>Check All Equivalent</i>
---------------------------------	-----------------------------

Description

Checks all elements in `x` equivalent using

```
length(x) < 2L || all(vapply(x, vld_equivalent, TRUE, y = x[[1]], tolerance = tolerance))
```

Usage

```
chk_all_equivalent(x, tolerance = sqrt(.Machine$double.eps), x_name = NULL)
```

```
vld_all_equivalent(x, tolerance = sqrt(.Machine$double.eps))
```

Arguments

<code>x</code>	The object to check.
<code>tolerance</code>	A non-negative numeric scalar.
<code>x_name</code>	A string of the name of object <code>x</code> or <code>NULL</code> .

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_all_equivalent()`: Validate All Equivalent

See Also

[length\(\)](#)

[vld_equivalent\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other `equal_` checkers: [chk_all_equal\(\)](#), [chk_all_identical\(\)](#), [chk_equal\(\)](#), [chk_equivalent\(\)](#), [chk_identical\(\)](#)

Other `all_` checkers: [chk_all\(\)](#), [chk_all_equal\(\)](#), [chk_all_identical\(\)](#)

Examples

```
# chk_all_equivalent
chk_all_equivalent(c(1, 1.00000001))
try(chk_all_equivalent(c(1, 1.0000001)))
chk_all_equivalent(list(c(x = 1), c(x = 1)))
chk_all_equivalent(list(c(x = 1), c(y = 1)))
# vld_all_equivalent
vld_all_equivalent(c(x = 1, y = 1))
```

<code>chk_all_identical</code>	<i>Check All Identical</i>
--------------------------------	----------------------------

Description

Checks all elements in `x` identical using

```
length(x) < 2L || all(vapply(x, vld_identical, TRUE, y = x[[1]]))
```

Pass: `c(1, 1, 1)`, `list(1, 1)`

Fail: `c(1, 1.0000001)`, `list(1, NA)`

Usage

```
chk_all_identical(x, x_name = NULL)
```

```
vld_all_identical(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_all_identical()`: Validate All Identical

See Also

[length\(\)](#)

[vld_identical\(\)](#)

For more details about the use of this function, please read the article [vignette\("chk-families"\)](#).

Other `equal_` checkers: [chk_all_equal\(\)](#), [chk_all_equivalent\(\)](#), [chk_equal\(\)](#), [chk_equivalent\(\)](#), [chk_identical\(\)](#)

Other `all_` checkers: [chk_all\(\)](#), [chk_all_equal\(\)](#), [chk_all_equivalent\(\)](#)

Examples

```
# chk_all_identical
chk_all_identical(c(1, 1))
try(chk_all_identical(c(1, 1.1)))
# vld_all_identical
vld_all_identical(c(1, 1))
```

chk_all_na

Check All Missing Values

Description

Checks if all missing values using

`all(is.na(x))`

Pass: NA, `c(NA, NA)`, `logical(0)`.

Fail: 1, 1:2, "1", `c(1, NA)`.

Usage

```
chk_all_na(x, x_name = NULL)
```

```
vld_all_na(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_all_na()`: Validate All Missing Values

See Also

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other `misc_checkers`: [chk_join\(\)](#), [chk_not_any_na\(\)](#), [chk_not_empty\(\)](#), [chk_unique\(\)](#)

Examples

```
# chk_all_na
try(chk_all_na(1))
try(chk_all_na(c(1, NA)))
chk_all_na(NA)
chk_all_na(c(NA, NA_character_, NA_real_))
# vld_all_na
vld_all_na(1)
vld_all_na(1:2)
vld_all_na(NA_real_)
vld_all_na(integer(0))
vld_all_na(c(NA, 1))
vld_all_na(TRUE)
vld_all_na(c(NA_real_, NA_character_))
```

chk_array	<i>Check Array</i>
-----------	--------------------

Description

Checks if is an array using
`is.array(x)`

Usage

```
chk_array(x, x_name = NULL)
```

```
vld_array(x)
```

Arguments

- | | |
|--------|---|
| x | The object to check. |
| x_name | A string of the name of object x or NULL. |

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_array()`: Validate Array

See Also

[is.array\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other `data_structure_checkers`: [chk_atomic\(\)](#), [chk_list\(\)](#), [chk_matrix\(\)](#), [chk_vector\(\)](#)

Examples

```
# chk_array
chk_array(array(1))
try(chk_array(matrix(1)))

# vld_array
vld_array(1)
vld_array(array(1))
```

chk_atomic

Check Atomic

Description

Checks if atomic using

`is.atomic(x)`

Usage

```
chk_atomic(x, x_name = NULL)
```

```
vld_atomic(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_atomic()`: Validate Atomic

See Also

[is.atomic\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other `data_structure_checkers`: [chk_array\(\)](#), [chk_list\(\)](#), [chk_matrix\(\)](#), [chk_vector\(\)](#)

Examples

```
# chk_atomic
chk_atomic(1)
try(chk_atomic(list(1)))
# vld_atomic
vld_atomic(1)
vld_atomic(matrix(1:3))
vld_atomic(character(0))
vld_atomic(list(1))
vld_atomic(NULL)
```

chk_character

Check Character

Description

Checks if character using
`is.character(x)`

Usage

```
chk_character(x, x_name = NULL)
```

```
vld_character(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_character()`: Validate Character

See Also

[is.character\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other `data_type_checkers`: [chk_character_or_factor\(\)](#), [chk_complex\(\)](#), [chk_double\(\)](#), [chk_environment\(\)](#), [chk_factor\(\)](#), [chk_integer\(\)](#), [chk_logical\(\)](#), [chk_number\(\)](#), [chk_numeric\(\)](#), [chk_raw\(\)](#)

Examples

```
# chk_character
chk_character("1")
try(chk_character(1))
# vld_character
vld_character("1")
vld_character(matrix("a"))
vld_character(character(0))
vld_character(NA_character_)
vld_character(1)
vld_character(TRUE)
vld_character(factor("text"))
```

chk_character_or_factor

Check Character or Factor

Description

Checks if character or factor using

`is.character(x) || is.factor(x)`

Usage

```
chk_character_or_factor(x, x_name = NULL)
```

```
vld_character_or_factor(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_character_or_factor()`: Validate Character or Factor

See Also

[is.character\(\)](#)

[is.factor\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other data_type_checkers: [chk_character\(\)](#), [chk_complex\(\)](#), [chk_double\(\)](#), [chk_environment\(\)](#), [chk_factor\(\)](#), [chk_integer\(\)](#), [chk_logical\(\)](#), [chk_number\(\)](#), [chk_numeric\(\)](#), [chk_raw\(\)](#)

Other factor_checkers: [chk_factor\(\)](#)

Examples

```
# chk_character_or_factor
chk_character_or_factor("1")
chk_character_or_factor(factor("1"))
try(chk_character_or_factor(1))
# vld_character_or_factor
vld_character_or_factor("1")
vld_character_or_factor(matrix("a"))
vld_character_or_factor(character(0))
vld_character_or_factor(NA_character_)
vld_character_or_factor(1)
vld_character_or_factor(TRUE)
vld_character_or_factor(factor("text"))
```

chk_chr	<i>Check Character Scalar</i>
---------	-------------------------------

Description

Checks if character scalar using
`is.character(x) && length(x) == 1L`

[Deprecated]

Usage

```
chk_chr(x, x_name = NULL)
```

```
vld_chr(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_chr()`: Validate Character Scalar
[Deprecated]

See Also

Other deprecated: [chk_dbl\(\)](#), [chk_deprecated](#), [chk_wnum\(\)](#)

Examples

```
chk_chr("a")
try(chk_chr(1))
# vld_chr
vld_chr("")
vld_chr("a")
vld_chr(NA_character_)
vld_chr(c("a", "b"))
vld_chr(1)
```

chk_class	<i>Check Type</i>
-----------	-------------------

Description

Checks inherits from class using
`inherits(x, class)`

Usage

```
chk_class(x, class, x_name = NULL)

vld_class(x, class)
```

Arguments

<code>x</code>	The object to check.
<code>class</code>	A character vector specifying the possible class values.
<code>x_name</code>	A string of the name of object <code>x</code> or <code>NULL</code> .

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_class()`: Validate Inherits from Class

See Also

[inherits\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other `id_checkers`: [chk_data\(\)](#), [chk_is\(\)](#), [chk_r6_class\(\)](#), [chk_s3_class\(\)](#), [chk_s3_class_strict\(\)](#), [chk_s4_class\(\)](#)

Examples

```
# chk_class
chk_class(1, "numeric")
try(chk_class(getClass("MethodDefinition"), "classRepresentation"))
# vld_class
vld_class(numeric(0), "numeric")
vld_class(getClass("MethodDefinition"), "classRepresentation")
```

`chk_compatible_lengths`*Check Compatible Lengths*

Description

Checks objects (including vectors) have lengths that could be 'strictly recycled'. That is to say they must all be either zero length or the same length with some of length 1.

Usage

```
chk_compatible_lengths(..., x_name = NULL)
```

```
vld_compatible_lengths(...)
```

Arguments

<code>...</code>	The objects to check for compatible lengths.
<code>x_name</code>	A string of the name of object x or NULL.

Details

This function helps to check vectors could be 'strictly recycled.' For example the function will error if you had a vector of length 2 and length 4, even though the vector of length 2 could be 'loosely recycled' to match up to the vector of length 4 when combined.

The intent of the function is to check that only strict recycling is occurring.

Value

The `chk_` function throws an informative error if the test fails.

Functions

- `vld_compatible_lengths()`: Validate Compatible Lengths

See Also

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other length_checkers: [chk_length\(\)](#)

Examples

```
# chk_compatible_lengths

a <- integer(0)
b <- numeric(0)
chk_compatible_lengths(a, b)
```

```
a <- 1
b <- 2
chk_compatible_lengths(a, b)

a <- 1:3
b <- 1:3
chk_compatible_lengths(a, b)

b <- 1
chk_compatible_lengths(a, b)

b <- 1:2
try(chk_compatible_lengths(a, b))

b <- 1:6
try(chk_compatible_lengths(a, b))
# vld_compatible_lengths

a <- integer(0)
b <- numeric(0)
vld_compatible_lengths(a, b)

a <- 1
b <- 2
vld_compatible_lengths(a, b)

a <- 1:3
b <- 1:3
vld_compatible_lengths(a, b)

b <- 1
vld_compatible_lengths(a, b)

b <- 1:2
vld_compatible_lengths(a, b)

b <- 1:6
vld_compatible_lengths(a, b)
```

`chk_complex`*Check Complex*

Description

Checks if complex using
`is.complex(x)`

Usage

```
chk_complex(x, x_name = NULL)
```

```
vld_complex(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_complex()`: Validate Complex

See Also

[is.complex\(\)](#)

For more details about the use of this function, please read the article [chk families](#).

Other `data_type_checkers`: [chk_character\(\)](#), [chk_character_or_factor\(\)](#), [chk_double\(\)](#), [chk_environment\(\)](#), [chk_factor\(\)](#), [chk_integer\(\)](#), [chk_logical\(\)](#), [chk_number\(\)](#), [chk_numeric\(\)](#), [chk_raw\(\)](#)

Examples

```
# chk_complex
chk_complex(1i)
try(chk_complex(1))
# vld_complex
vld_complex(1i)
vld_complex(complex())
vld_complex(NA_complex_)
vld_complex(1)
vld_complex(TRUE)
```

chk_complex_number	<i>Check Complex Number</i>
--------------------	-----------------------------

Description

Checks if non-missing complex scalar using
`is.complex(x) && length(x) == 1L && !anyNA(x)`

Usage

```
chk_complex_number(x, x_name = NULL)

vld_complex_number(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_complex_number()`: Validate Complex Number

See Also

[is.complex\(\)](#)
[length\(\)](#)

For more details about the use of this function, please read the article [chk families](#).

Other scalar_checkers: [chk_count\(\)](#), [chk_date\(\)](#), [chk_date_time\(\)](#), [chk_false\(\)](#), [chk_flag\(\)](#), [chk_lgl\(\)](#), [chk_number\(\)](#), [chk_scalar\(\)](#), [chk_string\(\)](#), [chk_true\(\)](#), [chk_tz\(\)](#), [chk_whole_number\(\)](#)

Examples

```
# chk_complex_number
chk_complex_number(as.complex(1.1))
try(chk_complex_number(1.1))
# vld_complex_number
vld_complex_number(as.complex(2))
```

chk_count	<i>Check Count</i>
-----------	--------------------

Description

Checks if non-negative whole number using

```
vld_whole_number(x) && x >= 0
```

Usage

```
chk_count(x, x_name = NULL)
```

```
vld_count(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_count()`: Validate Count

See Also

[vld_whole_number\(\)](#)

[length\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other scalar_checkers: [chk_complex_number\(\)](#), [chk_date\(\)](#), [chk_date_time\(\)](#), [chk_false\(\)](#), [chk_flag\(\)](#), [chk_lgl\(\)](#), [chk_number\(\)](#), [chk_scalar\(\)](#), [chk_string\(\)](#), [chk_true\(\)](#), [chk_tz\(\)](#), [chk_whole_number\(\)](#)

Other whole_number_checkers: [chk_whole_number\(\)](#), [chk_whole_numeric\(\)](#)

Examples

```
# chk_count
chk_count(1)
try(chk_count(1.5))
# vld_count
vld_count(1)
vld_count(0L)
vld_count(-1)
vld_count(0.5)
```

chk_data

*Check Data***Description**

Checks data.frame using

```
inherits(x, "data.frame")
```

Note that there is a similar function, [check_data\(\)](#), which checks the column names, values, number of rows, and keys of a data.frame.

Usage

```
chk_data(x, x_name = NULL)
```

```
vld_data(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_data()`: Validate Data

See Also

[inherits\(\)](#)

For more details about the use of this function, please read the article vignette("chk-families").

Other `id_checkers`: [chk_class\(\)](#), [chk_is\(\)](#), [chk_r6_class\(\)](#), [chk_s3_class\(\)](#), [chk_s3_class_strict\(\)](#), [chk_s4_class\(\)](#)

Examples

```
# chk_data
chk_data(data.frame(x = 1))
try(chk_data(1))
# vld_data
vld_data(data.frame())
vld_data(data.frame(x = 1))
vld_data(c(x = 1))
```

chk_date

*Check Date***Description**

Checks non-missing Date scalar using

```
inherits(x, "Date") && length(x) == 1L && !anyNA(x)
```

Usage

```
chk_date(x, x_name = NULL)

vld_date(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_date()`: Validate Date

See Also

[inherits\(\)](#)
[length\(\)](#)

For more details about the use of this function, please read the article vignette("chk-families").

Other scalar_checkers: [chk_complex_number\(\)](#), [chk_count\(\)](#), [chk_date_time\(\)](#), [chk_false\(\)](#), [chk_flag\(\)](#), [chk_lgl\(\)](#), [chk_number\(\)](#), [chk_scalar\(\)](#), [chk_string\(\)](#), [chk_true\(\)](#), [chk_tz\(\)](#), [chk_whole_number\(\)](#)

Other datetime_checkers: [chk_date_time\(\)](#)

Examples

```
# chk_date
chk_date(Sys.Date())
try(chk_date(1))
# vld_date
vld_date(Sys.Date())
vld_date(Sys.time())
vld_date(1)
```

chk_date_time	<i>Check Date Time</i>
---------------	------------------------

Description

Checks if non-missing POSIXct scalar using
`inherits(x, "POSIXct") && length(x) == 1L && !anyNA(x)`

Usage

```
chk_date_time(x, x_name = NULL)

chk_datetime(x, x_name = NULL)

vld_date_time(x)

vld_datetime(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `chk_datetime()`: Check Date Time (Deprecated)
[Deprecated]
- `vld_date_time()`: Validate Date Time
- `vld_datetime()`: Validate Date Time (Deprecated)
[Deprecated]

See Also

[inherits\(\)](#), [length\(\)](#)

For more details about the use of this function, please read the article [vignette\("chk-families"\)](#).

Other scalar_checkers: [chk_complex_number\(\)](#), [chk_count\(\)](#), [chk_date\(\)](#), [chk_false\(\)](#), [chk_flag\(\)](#), [chk_lgl\(\)](#), [chk_number\(\)](#), [chk_scalar\(\)](#), [chk_string\(\)](#), [chk_true\(\)](#), [chk_tz\(\)](#), [chk_whole_number\(\)](#)

Other datetime_checkers: [chk_date\(\)](#)

Examples

```
# chk_date_time
chk_date_time(as.POSIXct("2001-01-02"))
try(chk_date_time(1))
# vld_date_time
vld_date_time(as.POSIXct("2001-01-02"))
vld_date_time(Sys.time())
vld_date_time(1)
vld_date_time("2001-01-02")
vld_date_time(c(Sys.time(), Sys.time()))
```

chk_dbl

Check Double Scalar

Description

Checks if double scalar using

```
is.double(x) && length(x) == 1L
```

[Deprecated]

Usage

```
chk_dbl(x, x_name = NULL)
```

```
vld_dbl(x)
```

Arguments

x The object to check.

x_name A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_dbl()`: Validate Double
[Deprecated]

See Also

Other deprecated: [chk_chr\(\)](#), [chk_deprecated](#), [chk_wnum\(\)](#)

Examples

```
# chk_dbl
chk_dbl(1)
try(chk_dbl(1L))
# vld_dbl
vld_dbl(1)
vld_dbl(double(0))
vld_dbl(NA_real_)
vld_dbl(c(1, 1))
vld_dbl(1L)
```

chk_dir	<i>Check Directory Exists</i>
---------	-------------------------------

Description

Checks if directory exists using
`vld_string(x) && dir.exists(x)`

Usage

```
chk_dir(x, x_name = NULL)

vld_dir(x)
```

Arguments

<code>x</code>	The object to check.
<code>x_name</code>	A string of the name of object <code>x</code> or <code>NULL</code> .

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_dir()`: Validate Directory Exists

See Also[vld_string\(\)](#)[dir.exists\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other file_checkers: [chk_ext\(\)](#), [chk_file\(\)](#)

Examples

```
# chk_dir
chk_dir(tempdir())
try(chk_dir(tempfile()))
# vld_dir
vld_dir(1)
vld_dir(tempdir())
vld_dir(tempfile())
```

chk_double

Check Double

Description

Checks if double using
[is.double\(x\)](#)

Usage

```
chk_double(x, x_name = NULL)
```

```
vld_double(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_double()`: Validate Double

See Also[is.double\(\)](#)

For more details about the use of this function, please read the article [vignette\("chk-families"\)](#).

Other `data_type_checkers`: [chk_character\(\)](#), [chk_character_or_factor\(\)](#), [chk_complex\(\)](#), [chk_environment\(\)](#), [chk_factor\(\)](#), [chk_integer\(\)](#), [chk_logical\(\)](#), [chk_number\(\)](#), [chk_numeric\(\)](#), [chk_raw\(\)](#)

Examples

```
# chk_double
chk_double(1)
try(chk_double(1L))
# vld_double
vld_double(1)
vld_double(matrix(c(1, 2, 3, 4), nrow = 2L))
vld_double(double(0))
vld_double(numeric(0))
vld_double(NA_real_)
vld_double(1L)
vld_double(TRUE)
```

chk_environment

Check Environment

Description

Checks if environment using
[is.environment\(x\)](#)

Usage

```
chk_environment(x, x_name = NULL)
```

```
vld_environment(x)
```

Arguments

<code>x</code>	The object to check.
<code>x_name</code>	A string of the name of object <code>x</code> or <code>NULL</code> .

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_environment()`: Validate Environment

See Also

[is.environment\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other `data_type_checkers`: [chk_character\(\)](#), [chk_character_or_factor\(\)](#), [chk_complex\(\)](#), [chk_double\(\)](#), [chk_factor\(\)](#), [chk_integer\(\)](#), [chk_logical\(\)](#), [chk_number\(\)](#), [chk_numeric\(\)](#), [chk_raw\(\)](#)

Examples

```
# chk_environment
chk_environment(.GlobalEnv)
try(chk_environment(1))
# vld_environment
vld_environment(1)
vld_environment(list(1))
vld_environment(.GlobalEnv)
vld_environment(environment())
```

chk_equal

Check Equal

Description

Checks if is equal (identical within tolerance) to `y` using

`vld_true(all.equal(x, y, tolerance))`

Usage

```
chk_equal(x, y, tolerance = sqrt(.Machine$double.eps), x_name = NULL)
```

```
vld_equal(x, y, tolerance = sqrt(.Machine$double.eps))
```

Arguments

<code>x</code>	The object to check.
<code>y</code>	An object to check against.
<code>tolerance</code>	A non-negative numeric scalar.
<code>x_name</code>	A string of the name of object <code>x</code> or <code>NULL</code> .

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_equal()`: Validate Equal

See Also

`vld_true()`
[all.equal\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other `equal_checkers`: [chk_all_equal\(\)](#), [chk_all_equivalent\(\)](#), [chk_all_identical\(\)](#), [chk_equivalent\(\)](#), [chk_identical\(\)](#)

Examples

```
# chk_equal
chk_equal(1, 1.00000001)
try(chk_equal(1, 1.0000001))
chk_equal(1, 1L)
chk_equal(c(x = 1), c(x = 1L))
try(chk_equal(c(x = 1), c(y = 1L)))
vld_equal(1, 1.00000001)
```

<code>chk_equivalent</code>	<i>Check Equivalent</i>
-----------------------------	-------------------------

Description

Checks if is equivalent (equal ignoring attributes) to `y` using

```
vld_true(all.equal(x, y, tolerance, check.attributes = FALSE))
```

Usage

```
chk_equivalent(x, y, tolerance = sqrt(.Machine$double.eps), x_name = NULL)
```

```
vld_equivalent(x, y, tolerance = sqrt(.Machine$double.eps))
```

Arguments

<code>x</code>	The object to check.
<code>y</code>	An object to check against.
<code>tolerance</code>	A non-negative numeric scalar.
<code>x_name</code>	A string of the name of object <code>x</code> or <code>NULL</code> .

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_equivalent()`: Validate Equivalent

See Also

`vld_true()`
`all.equal()`

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other `equal_` checkers: `chk_all_equal()`, `chk_all_equivalent()`, `chk_all_identical()`, `chk_equal()`, `chk_identical()`

Examples

```
# chk_equivalent
chk_equivalent(1, 1.00000001)
try(chk_equivalent(1, 1.0000001))
chk_equivalent(1, 1L)
chk_equivalent(c(x = 1), c(y = 1))
vld_equivalent(c(x = 1), c(y = 1L))
```

chk_ext

Check File Extension

Description

Checks extension using

```
vld_string(x) && vld_subset(tools::file_ext(x), ext)
```

The user may want to use `toupper()` or `tolower()` to ensure the case matches.

Usage

```
chk_ext(x, ext, x_name = NULL)
```

```
vld_ext(x, ext)
```

Arguments

<code>x</code>	The object to check.
<code>ext</code>	A character vector of the permitted file extensions (without the <code>.</code>).
<code>x_name</code>	A string of the name of object <code>x</code> or <code>NULL</code> .

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_ext()`: Validate File Extension

See Also

[vld_string\(\)](#)

[vld_subset\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other file-checkers: [chk_dir\(\)](#), [chk_file\(\)](#)

Examples

```
# chk_ext
try(chk_ext("file1.pdf", "png"))
# vld_ext
vld_ext("oeu.pdf", "pdf")
vld_ext(toupper("oeu.pdf"), "PDF")
```

chk_factor

Check Factor

Description

Checks if factor using

`is.factor(x)`

Usage

```
chk_factor(x, x_name = NULL)
```

```
vld_factor(x)
```

Arguments

`x` The object to check.

`x_name` A string of the name of object `x` or `NULL`.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_factor()`: Validate Factor

See Also

[is.factor\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other `factor_checkers`: [chk_character_or_factor\(\)](#)

Other `data_type_checkers`: [chk_character\(\)](#), [chk_character_or_factor\(\)](#), [chk_complex\(\)](#), [chk_double\(\)](#), [chk_environment\(\)](#), [chk_integer\(\)](#), [chk_logical\(\)](#), [chk_number\(\)](#), [chk_numeric\(\)](#), [chk_raw\(\)](#)

Examples

```
# chk_factor
chk_factor(factor("1"))
try(chk_factor("1"))
# vld_factor
vld_factor(factor("1"))
vld_factor(factor(0))
vld_factor("1")
vld_factor(1L)
```

chk_false

Check FALSE

Description

Check if FALSE using

```
is.logical(x) && length(x) == 1L && !anyNA(x) && !x
```

Usage

```
chk_false(x, x_name = NULL)
```

```
vld_false(x)
```

Arguments

<code>x</code>	The object to check.
<code>x_name</code>	A string of the name of object <code>x</code> or <code>NULL</code> .

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_false()`: Validate FALSE

See Also

[is.logical\(\)](#)

[length\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other logical_checkers: [chk_flag\(\)](#), [chk_lgl\(\)](#), [chk_logical\(\)](#), [chk_true\(\)](#)

Other scalar_checkers: [chk_complex_number\(\)](#), [chk_count\(\)](#), [chk_date\(\)](#), [chk_date_time\(\)](#), [chk_flag\(\)](#), [chk_lgl\(\)](#), [chk_number\(\)](#), [chk_scalar\(\)](#), [chk_string\(\)](#), [chk_true\(\)](#), [chk_tz\(\)](#), [chk_whole_number\(\)](#)

Examples

```
# chk_false
chk_false(FALSE)
try(chk_false(0))
# vld_false
vld_false(TRUE)
vld_false(FALSE)
vld_false(NA)
vld_false(0)
vld_false(c(FALSE, FALSE))
```

chk_file

Check File Exists

Description

Checks if file exists using

```
vld_string(x) && file.exists(x) && !dir.exists(x)
```

Usage

```
chk_file(x, x_name = NULL)
```

```
vld_file(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_file()`: Validate File Exists

See Also

[vld_string\(\)](#)
[file.exists\(\)](#)
[dir.exists\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other file_checkers: [chk_dir\(\)](#), [chk_ext\(\)](#)

Examples

```
# chk_file
try(chk_file(tempfile()))
# vld_file
vld_file(tempfile())
```

chk_flag

Check Flag

Description

Checks if non-missing logical scalar using

```
is.logical(x) && length(x) == 1L && !anyNA(x)
```

Pass: TRUE, FALSE.

Fail: `logical(0)`, `c(TRUE, TRUE)`, `"TRUE"`, 1, NA.

Do not confuse this function with [chk_lgl\(\)](#), which also checks for logical scalars of `length(x) == 1` but can include NAs.

Usage

```
chk_flag(x, x_name = NULL)
```

```
vld_flag(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_flag()`: Validate Flag

See Also

[is.logical\(\)](#) [length\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other logical_checkers: [chk_false\(\)](#), [chk_lgl\(\)](#), [chk_logical\(\)](#), [chk_true\(\)](#)

Other scalar_checkers: [chk_complex_number\(\)](#), [chk_count\(\)](#), [chk_date\(\)](#), [chk_date_time\(\)](#), [chk_false\(\)](#), [chk_lgl\(\)](#), [chk_number\(\)](#), [chk_scalar\(\)](#), [chk_string\(\)](#), [chk_true\(\)](#), [chk_tz\(\)](#), [chk_whole_number\(\)](#)

Examples

```
# chk_flag
chk_flag(TRUE)
try(vld_flag(1))
# vld_flag
vld_flag(TRUE)
vld_flag(1)
```

chk_function

Check Function

Description

Checks if is a function using

```
is.function(x) && (is.null(formals) || length(formals(x)) == formals)
```

Usage

```
chk_function(x, formals = NULL, x_name = NULL)
```

```
vld_function(x, formals = NULL)
```

Arguments

x	The object to check.
formals	A count of the number of formal arguments.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_function()`: Validate Function

See Also

[is.function\(\)](#) [formals\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other ellipsis_checkers: [chk_unused\(\)](#), [chk_used\(\)](#)

Other missing_checkers: [chk_missing\(\)](#), [chk_not_missing\(\)](#)

Examples

```
# chk_function
chk_function(mean)
try(chk_function(1))
# vld_function
vld_function(mean)
vld_function(function(x) x)
vld_function(1)
vld_function(list(1))
```

chk_gt

Check Greater Than

Description

Checks if all non-missing values are greater than value using

```
all(x[!is.na(x)] > value)
```

Usage

```
chk_gt(x, value = 0, x_name = NULL)
```

```
vld_gt(x, value = 0)
```

Arguments

x	The object to check.
value	A non-missing scalar of a value.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_gt()`: Validate Greater Than

See Also

[all\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other range_checkers: [chk_gte\(\)](#), [chk_lt\(\)](#), [chk_lte\(\)](#), [chk_range\(\)](#)

Examples

```
# chk_gt
chk_gt(0.1)
try(chk_gt(c(0.1, -0.2)))
# vld_gt
vld_gt(numeric(0))
vld_gt(0)
vld_gt(0.1)
vld_gt(c(0.1, 0.2, NA))
vld_gt(c(0.1, -0.2))
vld_gt(c(-0.1, 0.2), value = -1)
vld_gt("b", value = "a")
```

chk_gte

Check Greater Than or Equal To

Description

Checks if all non-missing values are greater than or equal to y using `all(x[!is.na(x)] >= value)`

Usage

```
chk_gte(x, value = 0, x_name = NULL)
```

```
vld_gte(x, value = 0)
```

Arguments

x	The object to check.
value	A non-missing scalar of a value.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_gte()`: Validate Greater Than or Equal To

See Also

[all\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other range_checkers: [chk_gt\(\)](#), [chk_lt\(\)](#), [chk_lte\(\)](#), [chk_range\(\)](#)

Examples

```
# chk_gte
chk_gte(0)
try(chk_gte(-0.1))
# vld_gte
vld_gte(numeric(0))
vld_gte(0)
vld_gte(-0.1)
vld_gte(c(0.1, 0.2, NA))
vld_gte(c(0.1, 0.2, NA), value = 1)
```

chk_identical	<i>Check Identical</i>
---------------	------------------------

Description

Checks if is identical to y using
`identical(x, y)`

Usage

```
chk_identical(x, y, x_name = NULL)

vld_identical(x, y)
```

Arguments

x	The object to check.
y	An object to check against.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_identical()`: Validate Identical

See Also

[identical\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other `equal_` checkers: [chk_all_equal\(\)](#), [chk_all_equivalent\(\)](#), [chk_all_identical\(\)](#), [chk_equal\(\)](#), [chk_equivalent\(\)](#)

Examples

```
# chk_identical
chk_identical(1, 1)
try(chk_identical(1, 1L))
chk_identical(c(1, 1), c(1, 1))
try(chk_identical(1, c(1, 1)))
vld_identical(1, 1)
```

chk_integer

Check Integer

Description

Checks if integer using
`is.integer(x)`

Usage

```
chk_integer(x, x_name = NULL)

vld_integer(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_integer()`: Validate Integer

See Also

[is.integer\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other `data_type_checkers`: [chk_character\(\)](#), [chk_character_or_factor\(\)](#), [chk_complex\(\)](#), [chk_double\(\)](#), [chk_environment\(\)](#), [chk_factor\(\)](#), [chk_logical\(\)](#), [chk_number\(\)](#), [chk_numeric\(\)](#), [chk_raw\(\)](#)

Examples

```
# chk_integer
chk_integer(1L)
try(chk_integer(1))
# vld_integer
vld_integer(1L)
vld_integer(matrix(1:4, nrow = 2L))
vld_integer(integer(0))
vld_integer(NA_integer_)
vld_integer(1)
vld_integer(TRUE)
```

chk_is	<i>Check Class</i>
--------	--------------------

Description

Checks inherits from class using
`inherits(x, class)`

Usage

```
chk_is(x, class, x_name = NULL)

vld_is(x, class)
```

Arguments

x	The object to check.
class	A character vector specifying the possible class values.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_is()`: Validate Inherits from Class

See Also

[inherits\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other `id_checkers`: [chk_class\(\)](#), [chk_data\(\)](#), [chk_r6_class\(\)](#), [chk_s3_class\(\)](#), [chk_s3_class_strict\(\)](#), [chk_s4_class\(\)](#)

Examples

```
chk_is(1, "numeric")
try(chk_is(1L, "double"))

# vld_is
vld_is(numeric(0), "numeric")
vld_is(1L, "double")
```

chk_join

Check Join

Description

Checks if all rows in x match at least one in y.

Usage

```
chk_join(x, y, by, x_name = NULL)

vld_join(x, y, by)
```

Arguments

x	The object to check.
y	A data.frame with columns in by.
by	A character vector specifying the column names to join x and y on. If named the names are the corresponding columns in x.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_join()`: Validate Join

See Also

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other misc_checkers: [chk_all_na\(\)](#), [chk_not_any_na\(\)](#), [chk_not_empty\(\)](#), [chk_unique\(\)](#)

Examples

```
# chk_join
chk_join(data.frame(z = 1), data.frame(z = 1:2), by = "z")
try(chk_join(data.frame(z = 1), data.frame(z = 2), by = "z"))
# vld_join
vld_join(data.frame(z = 1), data.frame(z = 1:2), by = "z")
vld_join(data.frame(z = 1), data.frame(z = 2), by = "z")
vld_join(data.frame(z = 1), data.frame(a = 1:2), by = c(z = "a"))
vld_join(data.frame(z = 1), data.frame(a = 2), by = c(z = "a"))
```

chk_length

Check Length

Description

Checks length is a particular value or range using
`length(x) >= length && length(x) <= upper`

Usage

```
chk_length(x, length = 1L, upper = length, x_name = NULL)
```

```
vld_length(x, length = 1L, upper = length)
```

Arguments

x	The object to check.
length	A count of the length.
upper	A count of the max length.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_length()`: Validate Length

See Also

[length\(\)](#), [check_length\(\)](#), [check_dim\(\)](#)

For more details about the use of this function, please read the article vignette("chk-families").

Other length-checkers: [chk_compatible_lengths\(\)](#)

Examples

```
# chk_length
chk_length("text")
try(vld_length("text", length = 2))
# vld_length
vld_length(2:1, 2)
vld_length(2:1, 1)
```

chk_lgl

Check Logical Scalar

Description

Checks if logical scalar using

```
is.logical(x) && length(x) == 1L
```

If you only want to check the data type (not whether `length(x) == 1`), you should use the [chk_logical\(\)](#) function.

Usage

```
chk_lgl(x, x_name = NULL)
```

```
vld_lgl(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_lgl()`: Validate Logical Scalar

See Also

[is.logical\(\)](#)

[length\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other logical_checkers: [chk_false\(\)](#), [chk_flag\(\)](#), [chk_logical\(\)](#), [chk_true\(\)](#)

Other scalar_checkers: [chk_complex_number\(\)](#), [chk_count\(\)](#), [chk_date\(\)](#), [chk_date_time\(\)](#), [chk_false\(\)](#), [chk_flag\(\)](#), [chk_number\(\)](#), [chk_scalar\(\)](#), [chk_string\(\)](#), [chk_true\(\)](#), [chk_tz\(\)](#), [chk_whole_number\(\)](#)

Examples

```
# chk_lgl
chk_lgl(NA)
try(chk_lgl(1))
# vld_lgl
vld_lgl(TRUE)
vld_lgl(FALSE)
vld_lgl(NA)
vld_lgl(1)
vld_lgl(c(TRUE, TRUE))
```

chk_list

Check List

Description

Checks if is a list using

`is.list(x)`

Usage

```
chk_list(x, x_name = NULL)
```

```
vld_list(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_list()`: Validate List

See Also

[is.list\(\)](#)

For more details about the use of this function, please read the article vignette("chk-families").

Other data_structure_checkers: [chk_array\(\)](#), [chk_atomic\(\)](#), [chk_matrix\(\)](#), [chk_vector\(\)](#)

Examples

```
# chk_list
chk_list(list())
try(chk_list(1))
# vld_list
vld_list(list())
vld_list(list(x = 1))
vld_list(mtcars)
vld_list(1)
vld_list(NULL)
```

chk_logical

Check Logical

Description

Checks if logical using

```
is.logical(x)
```

If you want to check if it is a scalar, meaning that in addition to being of logical type, it has `length(x) == 1`, you should use [chk_lgl\(\)](#)

Usage

```
chk_logical(x, x_name = NULL)
```

```
vld_logical(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_logical()`: Validate Logical

See Also

[is.logical\(\)](#)

For more details about the use of this function, please read the article vignette("chk-families").

Other logical_checkers: [chk_false\(\)](#), [chk_flag\(\)](#), [chk_lgl\(\)](#), [chk_true\(\)](#)

Other data_type_checkers: [chk_character\(\)](#), [chk_character_or_factor\(\)](#), [chk_complex\(\)](#), [chk_double\(\)](#), [chk_environment\(\)](#), [chk_factor\(\)](#), [chk_integer\(\)](#), [chk_number\(\)](#), [chk_numeric\(\)](#), [chk_raw\(\)](#)

Examples

```
# chk_logical
chk_logical(TRUE)
try(chk_logical(1))
# vld_logical
vld_logical(TRUE)
vld_logical(matrix(TRUE))
vld_logical(logical(0))
vld_logical(NA)
vld_logical(1)
vld_logical("TRUE")
```

chk_lt	<i>Check Less Than</i>
--------	------------------------

Description

Checks if all non-missing values are less than value using
`all(x[!is.na(x)] < value)`

Usage

```
chk_lt(x, value = 0, x_name = NULL)

vld_lt(x, value = 0)
```

Arguments

x	The object to check.
value	A non-missing scalar of a value.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_lt()`: Validate Less Than

See Also

[all\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other range_checkers: [chk_gt\(\)](#), [chk_gte\(\)](#), [chk_lte\(\)](#), [chk_range\(\)](#)

Examples

```
# chk_lt
chk_lt(-0.1)
try(chk_lt(c(-0.1, 0.2)))
# vld_lt
vld_lt(numeric(0))
vld_lt(0)
vld_lt(-0.1)
vld_lt(c(-0.1, -0.2, NA))
vld_lt(c(-0.1, 0.2))
```

```
vld_lt(c(-0.1, 0.2), value = 1)
vld_lt("a", value = "b")
```

chk_lte

Check Less Than or Equal To

Description

Checks if all non-missing values are less than or equal to y using

```
all(x[!is.na(x)] <= value)
```

Usage

```
chk_lte(x, value = 0, x_name = NULL)
```

```
vld_lte(x, value = 0)
```

Arguments

x	The object to check.
value	A non-missing scalar of a value.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_lte()`: Validate Less Than or Equal To

See Also

[all\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other range_checkers: [chk_gt\(\)](#), [chk_gte\(\)](#), [chk_lt\(\)](#), [chk_range\(\)](#)

Examples

```
# chk_lte
chk_lte(0)
try(chk_lte(0.1))
# vld_lte
vld_lte(numeric(0))
vld_lte(0)
vld_lte(0.1)
vld_lte(c(-0.1, -0.2, NA))
vld_lte(c(-0.1, -0.2, NA), value = -1)
```

chk_match

*Check Matches***Description**

Checks if all values match regular expression using
`all(grepl(regex, x[!is.na(x)]))`

Usage

```
chk_match(x, regex = ".+", x_name = NULL)

vld_match(x, regex = ".+")
```

Arguments

<code>x</code>	The object to check.
<code>regex</code>	A string of a regular expression.
<code>x_name</code>	A string of the name of object <code>x</code> or <code>NULL</code> .

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_match()`: Validate Matches

See Also

[all\(\)](#)
[grepl\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Examples

```
# chk_match
chk_match("1")
try(chk_match("1", regexp = "2"))
# vld_match
vld_match("1")
vld_match("a", regexp = "a")
vld_match("")
vld_match("1", regexp = "2")
vld_match(NA_character_, regexp = ".*")
```

chk_matrix

*Check Matrix***Description**

Checks if is a matrix using
`is.matrix(x)`

Usage

```
chk_matrix(x, x_name = NULL)

vld_matrix(x)
```

Arguments

`x` The object to check.
`x_name` A string of the name of object `x` or `NULL`.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_matrix()`: Validate Matrix

See Also

[is.matrix\(\)](#)

For more details about the use of this function, please read the article vignette("chk-families").

Other data_structure_checkers: [chk_array\(\)](#), [chk_atomic\(\)](#), [chk_list\(\)](#), [chk_vector\(\)](#)

Examples

```
# chk_matrix
chk_matrix(matrix(1))
try(chk_matrix(array(1)))
# vld_matrix
vld_matrix(1)
vld_matrix(matrix(1))
```

chk_missing

*Check Missing Argument***Description**

Checks argument missing using
missing(x)

Usage

```
chk_missing(x, x_name = NULL)

vld_missing(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Details

Currently only checks if value is available (as opposed to whether it was specified).

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_missing()`: Validate Missing Argument

See Also

[missing\(\)](#)

For more details about the use of this function, please read the article vignette("chk-families").

Other missing_checkers: [chk_function\(\)](#), [chk_not_missing\(\)](#)

Examples

```
# chk_missing
fun <- function(x) {
  chk_missing(x)
}
fun()
try(fun(1))
# vld_missing
fun <- function(x) {
  vld_missing(x)
}
fun()
fun(1)
```

chk_named

Check Named

Description

Checks if is named using

```
!is.null(names(x))
```

Usage

```
chk_named(x, x_name = NULL)
```

```
vld_named(x)
```

Arguments

x The object to check.

x_name A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_named()`: Validate Named

See Also

[names\(\)](#)
[is.null\(\)](#)

For more details about the use of this function, please read the article [vignette\("chk-families"\)](#).

Other name_checkers: [chk_valid_name\(\)](#)

Examples

```
# chk_named
chk_named(c(x = 1))
try(chk_named(list(1)))
# vld_named
vld_named(c(x = 1))
vld_named(list(x = 1))
vld_named(c(x = 1)[-1])
vld_named(list(x = 1)[-1])
vld_named(1)
vld_named(list(1))
```

chk_not_any_na	<i>Check Not Any Missing Values</i>
----------------	-------------------------------------

Description

Checks if not any missing values using

`!anyNA(x)`

Pass: 1, 1:2, "1", `logical(0)`.

Fail: NA, `c(1, NA)`.

Usage

```
chk_not_any_na(x, x_name = NULL)
```

```
vld_not_any_na(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_not_any_na()`: Validate Not Any Missing Values

See Also

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other misc_checkers: `chk_all_na()`, `chk_join()`, `chk_not_empty()`, `chk_unique()`

Examples

```
# chk_not_any_na
chk_not_any_na(1)
try(chk_not_any_na(NA))
# vld_not_any_na
vld_not_any_na(1)
vld_not_any_na(1:2)
vld_not_any_na(NA_real_)
vld_not_any_na(integer(0))
vld_not_any_na(c(NA, 1))
vld_not_any_na(TRUE)
```

chk_not_empty

Check Not Empty

Description

Checks if not empty using

`length(x) != 0L`

Pass: 1, 1:2, NA, `matrix(1:3)`, `list(1)`, `data.frame(x = 1)`.

Fail: NULL, `logical(0)`, `list()`, `data.frame()`.

Usage

```
chk_not_empty(x, x_name = NULL)
```

```
vld_not_empty(x)
```

Arguments

`x` The object to check.

`x_name` A string of the name of object `x` or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_not_empty()`: Validate Not Empty

See Also

[length\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other misc_checkers: [chk_all_na\(\)](#), [chk_join\(\)](#), [chk_not_any_na\(\)](#), [chk_unique\(\)](#)

Examples

```
# chk_not_empty
chk_not_empty(1)
try(chk_not_empty(numeric(0)))
# vld_not_empty
vld_not_empty(1)
vld_not_empty(matrix(1:3))
vld_not_empty(character(0))
vld_not_empty(list(1))
vld_not_empty(NULL)
vld_not_empty(list())
```

chk_not_missing	<i>Check Not Missing Argument</i>
-----------------	-----------------------------------

Description

Checks argument not missing using

`!missing(x)`

Usage

```
chk_not_missing(x, x_name = "`x`")
```

```
vld_not_missing(x)
```

Arguments

<code>x</code>	The object to check.
<code>x_name</code>	A string of the name of object <code>x</code> or <code>NULL</code> .

Details

Currently only checks if value is available (as opposed to whether it was specified).

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_not_missing()`: Validate Not Missing Argument

See Also

[missing\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other missing_checkers: [chk_function\(\)](#), [chk_missing\(\)](#)

Examples

```
# chk_not_missing
fun <- function(x) {
  chk_not_missing(x)
}
fun(1)
try(fun())

# vld_not_missing
fun <- function(x) {
  vld_not_missing(x)
}
fun()
fun(1)
```

chk_not_null

Check not NULL

Description

Checks if not NULL using

`!is.null(x)`

Usage

`chk_not_null(x, x_name = NULL)`

`vld_not_null(x)`

Arguments

<code>x</code>	The object to check.
<code>x_name</code>	A string of the name of object <code>x</code> or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_not_null()`: Validate Not NULL

See Also

[is.null\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other null_checkers: [chk_null\(\)](#)

Examples

```
# chk_not_null
try(chk_not_null(NULL))
chk_not_null(1)
# vld_not_null
vld_not_null(1)
vld_not_null(NULL)
```

chk_not_subset	<i>Check Not Subset</i>
----------------	-------------------------

Description

Checks if not all values in values using
`!any(x %in% values) || !length(x)`

Usage

```
chk_not_subset(x, values, x_name = NULL)

vld_not_subset(x, values)
```

Arguments

<code>x</code>	The object to check.
<code>values</code>	A vector of the permitted values.
<code>x_name</code>	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_not_subset()`: Validate Not Subset

See Also

[any\(\)](#)

[length\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other `set_checkers`: [chk_order_set\(\)](#), [chk_setequal\(\)](#), [chk_subset\(\)](#), [chk_superset\(\)](#)

Examples

```
# chk_not_subset
chk_not_subset(11, 1:10)
try(chk_not_subset(1, 1:10))
# vld_not_subset
vld_not_subset(numeric(0), 1:10)
vld_not_subset(1, 1:10)
vld_not_subset(11, 1:10)
```

chk_null

Check NULL

Description

Checks if NULL using

`is.null(x)`

Usage

```
chk_null(x, x_name = NULL)
```

```
vld_null(x)
```

Arguments

`x` The object to check.

`x_name` A string of the name of object `x` or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_null()`: Validate NULL

See Also

[is.null\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other null_checkers: [chk_not_null\(\)](#)

Examples

```
# chk_null
try(chk_null(1))
chk_null(NULL)
# vld_null
vld_null(NULL)
vld_null(1)
```

chk_null_or

Check NULL Or

Description

Checks if NULL or passes test.

Usage

```
chk_null_or(x, chk, ..., vld, x_name = NULL)
```

Arguments

<code>x</code>	The object to check.
<code>chk</code>	A <code>chk</code> function. Soft-deprecated for <code>vld</code> . [Deprecated]
<code>...</code>	Arguments passed to <code>chk</code> .
<code>vld</code>	A <code>vld</code> function.
<code>x_name</code>	A string of the name of object <code>x</code> or NULL.

Value

An informative error if the test fails.

Examples

```
chk_null_or(NULL, chk_number)
chk_null_or(1, chk_number)
try(chk_null_or("1", chk_number))
```

chk_number

*Check Number***Description**

Checks if non-missing numeric scalar using
`is.numeric(x) && length(x) == 1L && !anyNA(x)`

Pass: 1, 2L, log(10), -Inf

Fail: "a", 1:3, NA_real_

Usage

```
chk_number(x, x_name = NULL)
```

```
vld_number(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_number()`: Validate Number

See Also

[is.numeric\(\)](#)

[length\(\)](#)

For more details about the use of this function, please read the article vignette("chk-families").

Other `data_type_checkers`: [chk_character\(\)](#), [chk_character_or_factor\(\)](#), [chk_complex\(\)](#), [chk_double\(\)](#), [chk_environment\(\)](#), [chk_factor\(\)](#), [chk_integer\(\)](#), [chk_logical\(\)](#), [chk_numeric\(\)](#), [chk_raw\(\)](#)

Other `scalar_checkers`: [chk_complex_number\(\)](#), [chk_count\(\)](#), [chk_date\(\)](#), [chk_date_time\(\)](#), [chk_false\(\)](#), [chk_flag\(\)](#), [chk_lgl\(\)](#), [chk_scalar\(\)](#), [chk_string\(\)](#), [chk_true\(\)](#), [chk_tz\(\)](#), [chk_whole_number\(\)](#)

Examples

```
# chk_number
chk_number(1.1)
try(chk_number(TRUE))
# vld_number
vld_number(1.1)
```

chk_numeric

*Check Numeric***Description**

Checks if numeric using

```
is.numeric(x)
```

Pass: 1, 1:2, NA_real_, integer(0), matrix(1:3).

Fail: TRUE, "1", NA, NULL.

Usage

```
chk_numeric(x, x_name = NULL)
```

```
vld_numeric(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_numeric()`: Validate Numeric

See Also

[is.numeric\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other `data_type_checkers`: [chk_character\(\)](#), [chk_character_or_factor\(\)](#), [chk_complex\(\)](#), [chk_double\(\)](#), [chk_environment\(\)](#), [chk_factor\(\)](#), [chk_integer\(\)](#), [chk_logical\(\)](#), [chk_number\(\)](#), [chk_raw\(\)](#)

Examples

```
# chk_numeric
chk_numeric(1)
try(chk_numeric("1"))
# vld_numeric
vld_numeric(1)
vld_numeric(1:2)
vld_numeric(NA_real_)
vld_numeric(integer(0))
vld_numeric("1")
vld_numeric(TRUE)
```

chk_orderset	<i>Check Set Ordered</i>
--------------	--------------------------

Description

Checks if the first occurrence of each shared element in `x` is equivalent to the first occurrence of each shared element in `values` using `vld_equivalent(unique(x[x %in% values]), values[values %in% x])`.

Usage

```
chk_orderset(x, values, x_name = NULL)

vld_orderset(x, values)
```

Arguments

<code>x</code>	The object to check.
<code>values</code>	A vector of the permitted values.
<code>x_name</code>	A string of the name of object <code>x</code> or <code>NULL</code> .

Value

The `chk_` function throws an informative error if the test fails.
 The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_orderset()`: Validate Set Ordered

See Also

[vld_equivalent\(\)](#)
[unique\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other set_checkers: [chk_not_subset\(\)](#), [chk_setequal\(\)](#), [chk_subset\(\)](#), [chk_superset\(\)](#)

Examples

```
# chk_orderaset
chk_orderaset(1:2, 1:2)
try(chk_orderaset(2:1, 1:2))

# vld_orderaset
vld_orderaset(1, 1)
vld_orderaset(1:2, 2:1)
vld_orderaset(1, 2:1)
vld_orderaset(1:2, 2)
```

chk_r6_class	<i>Check Inherits from R6 Class</i>
--------------	-------------------------------------

Description

Checks inherits from R6 class using
`inherits(x, "R6") && inherits(x, class)`

Usage

```
chk_r6_class(x, class, x_name = NULL)

vld_r6_class(x, class)
```

Arguments

x	The object to check.
class	A character vector specifying the possible class values.
x_name	A string of the name of object x or NULL.

Details

All R6 objects inherit from the class 'R6', so `chk_r6_class(x, "R6")` checks that x is an R6 object of any class. R6 class generators (as returned by [R6::R6Class\(\)](#)) are not themselves R6 objects.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_r6_class()`: Validate Inherits from R6 Class

See Also[inherits\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other `id_checkers`: [chk_class\(\)](#), [chk_data\(\)](#), [chk_is\(\)](#), [chk_s3_class\(\)](#), [chk_s3_class_strict\(\)](#), [chk_s4_class\(\)](#)

Examples

```
# chk_r6_class
chk_r6_class(R6::R6Class("exampleR6class")$new(), "exampleR6class")
try(chk_r6_class(1, "numeric"))

# vld_r6_class
vld_r6_class(R6::R6Class("exampleR6class")$new(), "exampleR6class")
vld_r6_class(1, "numeric")
```

chk_range

Checks range of non-missing values

Description

Checks all non-missing values fall within range using

If inclusive

```
all(x[!is.na(x)] >= range[1] & x[!is.na(x)] <= range[2])
```

else

```
all(x[!is.na(x)] > range[1] & x[!is.na(x)] < range[2])
```

Usage

```
chk_range(x, range = c(0, 1), inclusive = TRUE, x_name = NULL)
```

```
vld_range(x, range = c(0, 1), inclusive = TRUE)
```

Arguments

x	The object to check.
range	A non-missing sorted vector of length 2 of the lower and upper permitted values.
inclusive	A flag specifying whether the range is inclusive.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_range()`: Validate Range

See Also

[all\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other range_checkers: [chk_gt\(\)](#), [chk_gte\(\)](#), [chk_lt\(\)](#), [chk_lte\(\)](#)

Examples

```
# chk_range
chk_range(0)
try(chk_range(-0.1))
# vld_range
vld_range(numeric(0))
vld_range(0)
vld_range(-0.1)
vld_range(c(0.1, 0.2, NA))
vld_range(c(0.1, 0.2, NA), range = c(0, 1))
```

chk_raw

Check Raw

Description

Checks if raw using

`is.raw(x)`

Usage

```
chk_raw(x, x_name = NULL)
```

```
vld_raw(x)
```

Arguments

`x` The object to check.

`x_name` A string of the name of object `x` or `NULL`.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_raw()`: Validate Raw

See Also

[is.raw\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other `data_type_checkers`: [chk_character\(\)](#), [chk_character_or_factor\(\)](#), [chk_complex\(\)](#), [chk_double\(\)](#), [chk_environment\(\)](#), [chk_factor\(\)](#), [chk_integer\(\)](#), [chk_logical\(\)](#), [chk_number\(\)](#), [chk_numeric\(\)](#)

Examples

```
# chk_raw
chk_raw(as.raw(1))
try(chk_raw(1))
# vld_raw
vld_raw(as.raw(1))
vld_raw(raw(0))
vld_raw(1)
vld_raw(TRUE)
```

chk_s3_class

Check Type

Description

Checks inherits from S3 class using

```
inherits(x, class) && !isS4(x) && !inherits(x, "R6")
```

Usage

```
chk_s3_class(x, class, x_name = NULL)
```

```
vld_s3_class(x, class)
```

Arguments

<code>x</code>	The object to check.
<code>class</code>	A character vector specifying the possible class values.
<code>x_name</code>	A string of the name of object <code>x</code> or <code>NULL</code> .

Details

Note that **base objects** and S7 classes are considered S3 objects.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_s3_class()`: Validate Inherits from S3 Class

See Also

[inherits\(\)](#)

[typeof\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other id_checkers: [chk_class\(\)](#), [chk_data\(\)](#), [chk_is\(\)](#), [chk_r6_class\(\)](#), [chk_s3_class_strict\(\)](#), [chk_s4_class\(\)](#)

Examples

```
# chk_s3_class
chk_s3_class(1, "numeric")
chk_s3_class(factor(1), "factor")
try(chk_s3_class(getClass("MethodDefinition"), "classRepresentation"))
# vld_s3_class
vld_s3_class(numeric(0), "numeric")
vld_s3_class(getClass("MethodDefinition"), "classRepresentation")
```

chk_s3_class_strict	<i>Check Type</i>
---------------------	-------------------

Description

Checks inherits strictly from S3 class using

```
is.object(x, class) && !isS4(x) && !inherits(x, "R6") && !inherits(x, "S7_object")
```

Usage

```
chk_s3_class_strict(x, class, x_name = NULL)
```

```
vld_s3_class_strict(x, class)
```

Arguments

x	The object to check.
class	A character vector specifying the possible class values.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_s3_class_strict()`: Validate Inherits from S3 Class

See Also

[inherits\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other `id_checkers`: [chk_class\(\)](#), [chk_data\(\)](#), [chk_is\(\)](#), [chk_r6_class\(\)](#), [chk_s3_class\(\)](#), [chk_s4_class\(\)](#)

Examples

```
# chk_s3_class_strict
chk_s3_class_strict(factor(1), "factor")
try(chk_s3_class_strict(1, "numeric"))
try(chk_s3_class_strict(getClass("MethodDefinition"), "classRepresentation"))
# vld_s3_class_strict
vld_s3_class_strict(numeric(0), "numeric")
vld_s3_class_strict(getClass("MethodDefinition"), "classRepresentation")
```

chk_s4_class

Check Inherits from S4 Class

Description

Checks inherits from S4 class using
`isS4(x) && methods::is(x, class)`

Usage

```
chk_s4_class(x, class, x_name = NULL)

vld_s4_class(x, class)
```

Arguments

x	The object to check.
class	A character vector specifying the possible class values.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_s4_class()`: Validate Inherits from S4 Class

See Also

`methods::is()`

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other `id_checkers`: `chk_class()`, `chk_data()`, `chk_is()`, `chk_r6_class()`, `chk_s3_class()`, `chk_s3_class_strict()`

Examples

```
# chk_s4_class
try(chk_s4_class(1, "numeric"))
chk_s4_class(getClass("MethodDefinition"), "classRepresentation")
# vld_s4_class
vld_s4_class(numeric(0), "numeric")
vld_s4_class(getClass("MethodDefinition"), "classRepresentation")
```

chk_scalar

Check Scalar

Description

Checks if is a scalar using

```
length(x) == 1L
```

Usage

```
chk_scalar(x, x_name = NULL)
```

```
vld_scalar(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_scalar()`: Validate Scalar

See Also

[length\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other scalar_checkers: [chk_complex_number\(\)](#), [chk_count\(\)](#), [chk_date\(\)](#), [chk_date_time\(\)](#), [chk_false\(\)](#), [chk_flag\(\)](#), [chk_lgl\(\)](#), [chk_number\(\)](#), [chk_string\(\)](#), [chk_true\(\)](#), [chk_tz\(\)](#), [chk_whole_number\(\)](#)

Examples

```
# chk_scalar
chk_scalar(1)
chk_scalar(list(1))
try(chk_scalar(1:2))
# vld_scalar
vld_scalar(1)
```

chk_setequal

Check Set Equal

Description

Checks if equal set using
`setequal(x, values)`

Usage

```
chk_setequal(x, values, x_name = NULL)

vld_setequal(x, values)
```

Arguments

x	The object to check.
values	A vector of the permitted values.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_setequal()`: Validate Set Equal

See Also

[setequal\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other `set_checkers`: [chk_not_subset\(\)](#), [chk_order_set\(\)](#), [chk_subset\(\)](#), [chk_superset\(\)](#)

Examples

```
# chk_setequal
chk_setequal(1:2, 2:1)
try(chk_setequal(1, 1:2))
# vld_setequal
vld_setequal(1, 1)
vld_setequal(1:2, 2:1)
vld_setequal(1, 2:1)
vld_setequal(1:2, 2)
```

chk_sorted

Check Sorted

Description

Checks if is sorted using

```
!is.unsorted(x, na.rm = TRUE)
```

Usage

```
chk_sorted(x, x_name = NULL)
```

```
vld_sorted(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_sorted()`: Validate Sorted

See Also

[is.unsorted\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Examples

```
# chk_sorted
chk_sorted(1:2)
try(chk_sorted(2:1))
# vld_sorted
vld_sorted(1:2)
vld_sorted(2:1)
```

chk_string

Check String

Description

Checks if string

```
is.character(x) && length(x) == 1L && !anyNA(x)
```

Usage

```
chk_string(x, x_name = NULL)
```

```
vld_string(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_string()`: Validate String

See Also

[length\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other scalar_checkers: [chk_complex_number\(\)](#), [chk_count\(\)](#), [chk_date\(\)](#), [chk_date_time\(\)](#), [chk_false\(\)](#), [chk_flag\(\)](#), [chk_lgl\(\)](#), [chk_number\(\)](#), [chk_scalar\(\)](#), [chk_true\(\)](#), [chk_tz\(\)](#), [chk_whole_number\(\)](#)

Examples

```
# chk_string
chk_string("1")
try(chk_string(1))
# vld_string
vld_string("1")
vld_string("")
vld_string(1)
vld_string(NA_character_)
vld_string(c("1", "1"))
```

chk_subset

Check Subset

Description

Checks if all values in values using

```
all(x %in% values)
```

Pay attention to the order of the arguments value and x in this function compared to [chk_superset\(\)](#)

Usage

```
chk_subset(x, values, x_name = NULL)
```

```
vld_subset(x, values)
```

Arguments

x	The object to check.
values	A vector of the permitted values.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_subset()`: Validate Subset

See Also

[all\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other `set_checkers`: [chk_not_subset\(\)](#), [chk_order_set\(\)](#), [chk_setequal\(\)](#), [chk_superset\(\)](#)

Examples

```
# chk_subset
chk_subset(1, 1:10)
try(chk_subset(11, 1:10))
# vld_subset
vld_subset(numeric(0), 1:10)
vld_subset(1, 1:10)
vld_subset(11, 1:10)
```

chk_superset

Check Superset

Description

Checks if includes all values using

```
all(values %in% x)
```

Pay attention to the order of the arguments `value` and `x` in this function compared to [chk_subset\(\)](#)

Usage

```
chk_superset(x, values, x_name = NULL)
```

```
vld_superset(x, values)
```

Arguments

x	The object to check.
values	A vector of the permitted values.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_superset()`: Validates Superset

See Also

[all\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other `set_checkers`: [chk_not_subset\(\)](#), [chk_order_set\(\)](#), [chk_setequal\(\)](#), [chk_subset\(\)](#)

Examples

```
# chk_superset
chk_superset(1:3, 1)
try(chk_superset(1:3, 4))
# vld_superset
vld_superset(1:3, 1)
vld_superset(1:3, 4)
vld_superset(integer(0), integer(0))
```

chk_true

Check TRUE

Description

Checks if TRUE using

```
is.logical(x) && length(x) == 1L && !anyNA(x) && x
```

Usage

```
chk_true(x, x_name = NULL)
```

```
vld_true(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_true()`: Validate TRUE

See Also

[is.logical\(\)](#)

[length\(\)](#)

For more details about the use of this function, please read the article vignette("chk-families").

Other logical_checkers: [chk_false\(\)](#), [chk_flag\(\)](#), [chk_lgl\(\)](#), [chk_logical\(\)](#)

Other scalar_checkers: [chk_complex_number\(\)](#), [chk_count\(\)](#), [chk_date\(\)](#), [chk_date_time\(\)](#), [chk_false\(\)](#), [chk_flag\(\)](#), [chk_lgl\(\)](#), [chk_number\(\)](#), [chk_scalar\(\)](#), [chk_string\(\)](#), [chk_tz\(\)](#), [chk_whole_number\(\)](#)

Examples

```
# chk_true
chk_true(TRUE)
try(chk_true(1))
# vld_true
vld_true(TRUE)
vld_true(FALSE)
vld_true(NA)
vld_true(0)
vld_true(c(TRUE, TRUE))
```

chk_tz

Check Time Zone

Description

Checks if non-missing valid scalar timezone using

```
is.character(x) && length(x) == 1L && !anyNA(x) && x %in% OlsonNames()
```

Usage

```
chk_tz(x, x_name = NULL)

vld_tz(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_tz()`: Validate Time Zone

See Also

[length\(\)](#)
[OlsonNames\(\)](#)
[is.character\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other scalar_checkers: [chk_complex_number\(\)](#), [chk_count\(\)](#), [chk_date\(\)](#), [chk_date_time\(\)](#), [chk_false\(\)](#), [chk_flag\(\)](#), [chk_lgl\(\)](#), [chk_number\(\)](#), [chk_scalar\(\)](#), [chk_string\(\)](#), [chk_true\(\)](#), [chk_whole_number\(\)](#)

Examples

```
chk_tz("UTC")
try(chk_tz("TCU"))
vld_tz("UTC")
vld_tz("TCU")
```

chk_unique

Check Unique

Description

Checks if unique using

```
!anyDuplicated(x, incomparables = incomparables)
```

Usage

```
chk_unique(x, incomparables = FALSE, x_name = NULL)
```

```
vld_unique(x, incomparables = FALSE)
```

Arguments

x	The object to check.
incomparables	A vector of values that cannot be compared. FALSE means that all values can be compared.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_unique()`: Validate Unique

See Also

[anyDuplicated\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other `misc_checkers`: [chk_all_na\(\)](#), [chk_join\(\)](#), [chk_not_any_na\(\)](#), [chk_not_empty\(\)](#)

Examples

```
# chk_unique
chk_unique(c(NA, 2))
try(chk_unique(c(NA, NA, 2)))
chk_unique(c(NA, NA, 2), incomparables = NA)
# vld_unique
vld_unique(NULL)
vld_unique(numeric(0))
vld_unique(c(NA, 2))
vld_unique(c(NA, NA, 2))
vld_unique(c(NA, NA, 2), incomparables = NA)
```

chk_unused	<i>Check ... Unused</i>
------------	-------------------------

Description

Checks if ... is unused
`length(list(...)) == 0L`

Usage

```
chk_unused(...)  
  
vld_unused(...)
```

Arguments

... Additional arguments.

Value

The `chk_` function throws an informative error if the test fails.

Functions

- `vld_unused()`: Validate ... Unused

See Also

[length\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other ellipsis_checkers: [chk_function\(\)](#), [chk_used\(\)](#)

Examples

```
# chk_unused  
fun <- function(x, ...) {  
  chk_unused(...)  
  x  
}  
fun(1)  
try(fun(1, 2))  
# vld_unused  
fun <- function(x, ...) {  
  vld_unused(...)  
}  
fun(1)  
try(fun(1, 2))
```

chk_used	<i>Check ... Used</i>
----------	-----------------------

Description

Checks if is ... used using
`length(list(...)) != 0L`

Usage

```
chk_used(...)  
  
vld_used(...)
```

Arguments

... Additional arguments.

Value

The `chk_` function throws an informative error if the test fails.

Functions

- `vld_used()`: Validate ... Used

See Also

[length\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other ellipsis_checkers: [chk_function\(\)](#), [chk_unused\(\)](#)

Examples

```
# chk_used  
fun <- function(x, ...) {  
  chk_used(...)  
  x  
}  
try(fun(1))  
fun(1, 2)  
# vld_used  
fun <- function(x, ...) {  
  vld_used(...)  
}  
fun(1)  
fun(1, 2)
```

chk_valid_name	<i>Check Valid Name</i>
----------------	-------------------------

Description

Checks if valid name using

```
identical(make.names(x[!is.na(x)]), as.character(x[!is.na(x)]))
```

Usage

```
chk_valid_name(x, x_name = NULL)
```

```
vld_valid_name(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_valid_name()`: Validate Valid Name

See Also

[identical\(\)](#)

[make.names\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other name_checkers: [chk_named\(\)](#)

Examples

```
# chk_valid_name
chk_valid_name("text")
try(chk_valid_name(".1"))
# vld_valid_name
vld_valid_name(".1")
```

chk_vector

*Check Vector***Description**

Checks if is a vector using

```
(is.atomic(x) && !is.matrix(x) && !is.array(x)) || is.list(x)
```

Usage

```
chk_vector(x, x_name = NULL)
```

```
vld_vector(x)
```

Arguments

x The object to check.

x_name A string of the name of object x or NULL.

Details

`is.vector(x)` is not reliable because it returns TRUE only if the object is a vector with no attributes apart from names.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_vector()`: Validate Vector

See Also

[is.atomic\(\)](#), [is.matrix\(\)](#), [is.array\(\)](#), [is.list\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other `data_structure_checkers`: [chk_array\(\)](#), [chk_atomic\(\)](#), [chk_list\(\)](#), [chk_matrix\(\)](#)

Examples

```
# chk_vector
chk_vector(1)
chk_vector(list())
try(chk_vector(matrix(1)))
# vld_vector
vld_vector(1)
```

chk_whole_number	<i>Check Whole Number</i>
------------------	---------------------------

Description

Checks if non-missing integer scalar or double equivalent using
`vld_number(x) && (is.integer(x) || vld_true(all.equal(x, trunc(x))))`

Pass: 1, 2L, 1e10, -Inf

Fail: "a", 1:3, NA_integer_, log(10)

Usage

```
chk_whole_number(x, x_name = NULL)
```

```
vld_whole_number(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_whole_number()`: Validate Whole Number

See Also

[is.integer\(\)](#)

[vld_true\(\)](#)

[vld_number\(\)](#)

For more details about the use of this function, please read the article `vignette("chk-families")`.

Other scalar_checkers: [chk_complex_number\(\)](#), [chk_count\(\)](#), [chk_date\(\)](#), [chk_date_time\(\)](#), [chk_false\(\)](#), [chk_flag\(\)](#), [chk_lgl\(\)](#), [chk_number\(\)](#), [chk_scalar\(\)](#), [chk_string\(\)](#), [chk_true\(\)](#), [chk_tz\(\)](#)

Other whole_number_checkers: [chk_count\(\)](#), [chk_whole_numeric\(\)](#)

Examples

```
# chk_whole_number
chk_whole_number(2)
try(chk_whole_number(1.1))
# vld_whole_number
vld_whole_number(2)
```

chk_whole_numeric	<i>Check Whole Numeric</i>
-------------------	----------------------------

Description

Checks if integer vector or double equivalent using
`is.integer(x) || (is.double(x) && vld_true(all.equal(x, as.integer(x))))`

Usage

```
chk_whole_numeric(x, x_name = NULL)

vld_whole_numeric(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_whole_numeric()`: Validate Whole Numeric

See Also

[is.integer\(\)](#)
[is.double\(\)](#)
[vld_true\(\)](#)
[all.equal\(\)](#)

For more details about the use of this function, please read the article vignette("chk-families").

Other whole_number_checkers: [chk_count\(\)](#), [chk_whole_number\(\)](#)

Examples

```
# chk_whole_numeric
chk_whole_numeric(1)
try(chk_whole_numeric(1.1))
# vld_whole_numeric
vld_whole_numeric(1)
vld_whole_numeric(NA_real_)
vld_whole_numeric(1:2)
vld_whole_numeric(double(0))
vld_whole_numeric(TRUE)
vld_whole_numeric(1.5)
```

chk_wnum	<i>Check Whole Numeric Scalar</i>
----------	-----------------------------------

Description

Checks if whole numeric scalar using

```
is.numeric(x) && length(x) == 1L && (is.integer(x) || vld_true(all.equal(x, trunc(x))))
```

[Deprecated]

Usage

```
chk_wnum(x, x_name = NULL)
```

```
vld_wnum(x)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

The `chk_` function throws an informative error if the test fails or returns the original object if successful so it can be used in pipes.

The `vld_` function returns a flag indicating whether the test was met.

Functions

- `vld_wnum()`: Validate Whole Numeric Scalar

[Deprecated]

See Also

Other deprecated: [chk_chr\(\)](#), [chk_dbl\(\)](#), [chk_deprecated](#)

Examples

```
# chk_wnum
chk_wnum(1)
try(chk_wnum(1.1))
# vld_wnum
vld_wnum(1)
vld_wnum(double(0))
vld_wnum(NA_real_)
vld_wnum(c(1, 1))
vld_wnum(1L)
```

deparse_backtick_chk *Deparse Backtick*

Description

deparse_backtick_chk is a wrapper on [deparse1\(\)](#) and backtick_chk.

Usage

```
deparse_backtick_chk(x)
```

```
backtick_chk(x)
```

```
unbacktick_chk(x)
```

Arguments

x A substituted object to deparse.

Details

It is exported to allow users to easily construct their own chk_ functions.

Value

A string of the backticked substituted object.

Functions

- backtick_chk(): Backtick
- unbacktick_chk(): Unbacktick

See Also

[deparse1\(\)](#)

Examples

```
# deparse_backtick_chk
deparse_backtick_chk(2)
deparse_backtick_chk(2^2)
```

err

Stop, Warning and Message Messages

Description

The functions call `message_chk()` to process the message and then `rlang::abort()`, `rlang::warn()` and `rlang::inform()`, respectively.

Usage

```
err(
  ...,
  n = NULL,
  tidy = TRUE,
  .subclass = NULL,
  class = NULL,
  call = rlang::caller_call(3)
)

wrn(..., n = NULL, tidy = TRUE, .subclass = NULL, class = NULL)

msg(..., n = NULL, tidy = TRUE, .subclass = NULL, class = NULL)
```

Arguments

<code>...</code>	zero or more objects which can be coerced to character (and which are pasted together with no separator) or a single condition object.
<code>n</code>	The value of <code>n</code> for converting <code>sprintf</code> -like types.
<code>tidy</code>	A flag specifying whether capitalize the first character and add a missing period.
<code>.subclass</code>	A string of the class of the error message.
<code>class</code>	Subclass of the condition.
<code>call</code>	The execution environment of a currently running function, e.g. <code>call = caller_env()</code>. The corresponding function call is retrieved and mentioned in error messages as the source of the error. You only need to supply <code>call</code> when throwing a condition from a helper function which wouldn't be relevant to mention in the message. Can also be <code>NULL</code> or a defused function call to respectively not display any call or hard-code a code to display. For more information about error calls, see Including function calls in error messages.

Details

The user can set the subclass.

Value

`err()` throws an error. `wrn()` signals a warning and `msg()` signals a message; both return `NULL` invisibly.

Functions

- `err()`: Error
- `wrn()`: Warning
- `msg()`: Message

Examples

```
# err
try(err("there %r %n problem value%s", n = 2))

# wrn
wrn("there %r %n problem value%s", n = 2)

# msg
msg("there %r %n problem value%s", n = 2)
```

expect_chk_error	<i>Expect Chk Error</i>
------------------	-------------------------

Description

`expect_chk_error()` checks that code throws an error of class `"chk_error"` with a message that matches regexp. See below for more details.

Usage

```
expect_chk_error(
  object,
  regexp = NULL,
  ...,
  info = NULL,
  label = NULL,
  class = NULL
)
```

Arguments

object	Object to test. Supports limited unquoting to make it easier to generate readable failures within a function or for loop. See quasi_label for more details.
regexp	Regular expression to test against. • A character vector giving a regular expression that must match the error message. • If NULL, the default, asserts that there should be an error, but doesn't test for a specific value. • If NA, asserts that there should be no errors, but we now recommend using expect_no_error() and friends instead. Note that you should only use message with errors/warnings/messages that you generate. Avoid tests that rely on the specific text generated by another package since this can easily change. If you do need to test text generated by another package, either protect the test with <code>skip_on_cran()</code> or use <code>expect_snapshot()</code>.
...	Arguments passed on to expect_match fixed If TRUE, treats regexp as a string to be matched exactly (not a regular expressions). Overrides perl. perl logical. Should Perl-compatible regexps be used?
info	Extra information to be included in the message. This argument is soft-deprecated and should not be used in new code. Instead see alternatives in quasi_label .
label	Used to customise failure messages. For expert use only.
class	Must be NULL.

Value

If regexp = NA, the value of the first argument; otherwise the captured condition.

Testing message vs class

When checking that code generates an error, it's important to check that the error is the one you expect. There are two ways to do this. The first way is the simplest: you just provide a regexp that match some fragment of the error message. This is easy, but fragile, because the test will fail if the error message changes (even if its the same error).

A more robust way is to test for the class of the error, if it has one. You can learn more about custom conditions at <https://adv-r.hadley.nz/conditions.html#custom-conditions>, but in short, errors are S3 classes and you can generate a custom class and check for it using class instead of regexp.

If you are using `expect_error()` to check that an error message is formatted in such a way that it makes sense to a human, we recommend using [expect_snapshot\(\)](#) instead.

See Also

[expect_no_error\(\)](#), [expect_no_warning\(\)](#), [expect_no_message\(\)](#), and [expect_no_condition\(\)](#) to assert that code runs without errors/warnings/messages/conditions.

Other expectations: [comparison-expectations](#), [equality-expectations](#), [expect_length\(\)](#), [expect_match\(\)](#), [expect_named\(\)](#), [expect_null\(\)](#), [expect_output\(\)](#), [expect_reference\(\)](#), [expect_silent\(\)](#), [inheritance-expectations](#), [logical-expectations](#)

Examples

```
expect_chk_error(chk_true(FALSE))
try(expect_chk_error(chk_false(FALSE)))
```

generate_check_data	<i>Generate check_data() Code</i>
---------------------	-----------------------------------

Description

Generates the code for a [check_data\(\)](#) call from an example data frame so the column specification does not have to be written out by hand.

Usage

```
generate_check_data(x, data_name = NULL)
```

Arguments

x	A data frame to generate the check_data() code from.
data_name	A string of the name to use for the data frame in the generated code or NULL to use the name of x.

Details

For each column the generated values entry captures

- the column's class,
- whether it permits missing values,
- the range (minimum and maximum) of numeric, Date and POSIXct columns, and
- the levels of factors.

The result is a starting point that is expected to be edited, for example to widen a range or relax a set of factor levels. The code is printed to the console (so it can be copied) and returned invisibly as a string.

Value

A string of the generated code, invisibly.

See Also[check_data\(\)](#)**Examples**

```
data <- data.frame(
  count = c(1L, 3L, NA),
  letter = c("a", "b", "c"),
  height = c(1.5, 2.5, 3.5)
)
generate_check_data(data)
```

message_chk*Construct Tidyverse Style Message*

Description

If tidy = TRUE constructs a tidyverse style message by

Usage

```
message_chk(..., n = NULL, tidy = TRUE)
```

Arguments

...	Multiple objects that are converted to a string using <code>paste0(..., collapse = '')</code> .
n	The value of n for converting <code>sprintf</code> -like types.
tidy	A flag specifying whether capitalize the first character and add a missing period.

Details

- Capitalizing the first character if possible.
- Adding a trailing . if missing.

Also if n != NULL replaces the recognized `sprintf`-like types.

Value

A string of the message.

sprintf-like types

The following recognized sprintf-like types can be used in a message:

```
n The value of n.
s " if n == 1 otherwise 's'
es " if n == 1 otherwise 'es'
r 'is' if n == 1 otherwise 'are'
y 'y' if n == 1 otherwise 'ie'
ies 'y' if n == 1 otherwise 'ies'
% A literal '%'.
```

The types are replaced in a single left to right pass, so %% escapes a literal '%' and a replacement is never rescanned. A '%' that does not start a recognized type is left as is.

Examples

```
message_chk("there %r %n", " problem director%y%s")
message_chk("there %r %n", " problem director%y%s", n = 1)
message_chk("There %r %n", " problem director%y%s.", n = 3)
message_chk("there %r %n problem class%s", n = 1)
message_chk("there %r %n problem class%s", n = 3)
message_chk("%n%% of value%s", n = 2)
```

p

Concatenate Strings

Description

A wrapper on `base::paste()`.

Usage

```
p(..., sep = " ", collapse = NULL, recycle0 = FALSE)
```

```
p0(..., collapse = NULL, recycle0 = FALSE)
```

Arguments

...	one or more R objects, to be converted to character vectors.
sep	a character string to separate the terms. Not <code>NA_character_</code> .
collapse	an optional character string to separate the results. Not <code>NA_character_</code> . When collapse is a string, the result is always a string (<code>character</code> of length 1).
recycle0	<code>logical</code> indicating if zero-length character arguments should result in the zero-length <code>character</code> (0). Note that when collapse is a string, <code>recycle0</code> does <i>not</i> recycle to zero-length, but to "".

Value

A character vector.

Functions

- `p0()`: A wrapper on `base::paste0()`

Examples

```
p("a", "b")
p(c("a", "b"), collapse = " ")
p("a", character(0))
p("a", character(0), recycle0 = TRUE)
p0("a", "b")
p0(c("a", "b"), collapse = "")
p0("a", character(0))
p0("a", character(0), recycle0 = TRUE)
```

Index

- * **all_checkers**
 - chk_all, 14
 - chk_all_equal, 15
 - chk_all_equivalent, 16
 - chk_all_identical, 17
- * **check**
 - check_data, 6
 - check_dim, 7
 - check_dirs, 8
 - check_files, 9
 - check_key, 9
 - check_length, 10
 - check_names, 11
 - check_values, 12
- * **data_structure_checkers**
 - chk_array, 19
 - chk_atomic, 20
 - chk_list, 54
 - chk_matrix, 60
 - chk_vector, 92
- * **data_type_checkers**
 - chk_character, 21
 - chk_character_or_factor, 22
 - chk_complex, 27
 - chk_double, 36
 - chk_environment, 37
 - chk_factor, 41
 - chk_integer, 49
 - chk_logical, 55
 - chk_number, 70
 - chk_numeric, 71
 - chk_raw, 75
- * **date_checkers**
 - chk_tz, 86
- * **datetime_checkers**
 - chk_date, 32
 - chk_date_time, 33
- * **deprecated**
 - chk_chr, 24
 - chk_dbl, 34
 - chk_wnum, 95
- * **ellipsis_checkers**
 - chk_function, 45
 - chk_unused, 89
 - chk_used, 90
- * **equal_checkers**
 - chk_all_equal, 15
 - chk_all_equivalent, 16
 - chk_all_identical, 17
 - chk_equal, 38
 - chk_equivalent, 39
 - chk_identical, 48
- * **factor_checkers**
 - chk_character_or_factor, 22
 - chk_factor, 41
- * **file_checkers**
 - chk_dir, 35
 - chk_ext, 40
 - chk_file, 43
- * **function_checkers**
 - chk_function, 45
- * **id_checkers**
 - chk_class, 25
 - chk_data, 31
 - chk_is, 50
 - chk_r6_class, 73
 - chk_s3_class, 76
 - chk_s3_class_strict, 77
 - chk_s4_class, 78
- * **length_checkers**
 - chk_compatible_lengths, 26
 - chk_length, 52
- * **logical_checkers**
 - chk_false, 42
 - chk_flag, 44
 - chk_lgl, 53
 - chk_logical, 55
 - chk_true, 85

- * **misc_checkers**
 - chk_all_na, 18
 - chk_join, 51
 - chk_not_any_na, 63
 - chk_not_empty, 64
 - chk_unique, 87
- * **missing_checkers**
 - chk_function, 45
 - chk_missing, 61
 - chk_not_missing, 65
- * **name_checkers**
 - chk_named, 62
 - chk_valid_name, 91
- * **null_checkers**
 - chk_not_null, 66
 - chk_null, 68
- * **range_checkers**
 - chk_gt, 46
 - chk_gte, 47
 - chk_lt, 57
 - chk_lte, 58
 - chk_range, 74
- * **regex_checkers**
 - chk_match, 59
- * **scalar_checkers**
 - chk_complex_number, 29
 - chk_count, 30
 - chk_date, 32
 - chk_date_time, 33
 - chk_false, 42
 - chk_flag, 44
 - chk_lgl, 53
 - chk_number, 70
 - chk_scalar, 79
 - chk_string, 82
 - chk_true, 85
 - chk_tz, 86
 - chk_whole_number, 93
- * **set_checkers**
 - chk_not_subset, 67
 - chk_ordinalset, 72
 - chk_setequal, 80
 - chk_subset, 83
 - chk_superset, 84
- * **sorted_checkers**
 - chk_sorted, 81
- * **tz_checkers**
 - chk_tz, 86
- * **whole_number_checkers**
 - chk_count, 30
 - chk_whole_number, 93
 - chk_whole_numeric, 94
- abort_chk, 4
- all(), 15, 47, 48, 57–59, 75, 84, 85
- all.equal(), 39, 40, 94
- any(), 68
- anyDuplicated(), 88
- backtick_chk (deparse_backtick_chk), 96
- base::paste(), 102
- base::paste0(), 103
- cc, 5
- character, 102
- check_data, 6
- check_data(), 7–12, 31, 100, 101
- check_dim, 7
- check_dim(), 7–12, 53
- check_dirs, 8
- check_dirs(), 7, 9–12
- check_files, 9
- check_files(), 7, 8, 10–12
- check_key, 9
- check_key(), 7–12
- check_length, 10
- check_length(), 7–12, 53
- check_names, 11
- check_names(), 7–10, 12
- check_values, 12
- check_values(), 6–11
- chk_all, 14
- chk_all(), 16–18
- chk_all_equal, 15
- chk_all_equal(), 15, 17, 18, 39, 40, 49
- chk_all_equivalent, 16
- chk_all_equivalent(), 15, 16, 18, 39, 40, 49
- chk_all_identical, 17
- chk_all_identical(), 15–17, 39, 40, 49
- chk_all_na, 18
- chk_all_na(), 52, 64, 65, 88
- chk_array, 19
- chk_array(), 21, 55, 60, 92
- chk_atomic, 20
- chk_atomic(), 20, 55, 60, 92
- chk_character, 21

- `chk_character()`, 23, 28, 37, 38, 42, 50, 56, 70, 71, 76
- `chk_character_or_factor`, 22
- `chk_character_or_factor()`, 22, 28, 37, 38, 42, 50, 56, 70, 71, 76
- `chk_chr`, 24
- `chk_chr()`, 35, 95
- `chk_class`, 25
- `chk_class()`, 31, 51, 74, 77–79
- `chk_compatible_lengths`, 26
- `chk_compatible_lengths()`, 53
- `chk_complex`, 27
- `chk_complex()`, 22, 23, 37, 38, 42, 50, 56, 70, 71, 76
- `chk_complex_number`, 29
- `chk_complex_number()`, 30, 32, 34, 43, 45, 54, 70, 80, 83, 86, 87, 93
- `chk_count`, 30
- `chk_count()`, 29, 32, 34, 43, 45, 54, 70, 80, 83, 86, 87, 93, 94
- `chk_data`, 31
- `chk_data()`, 25, 51, 74, 77–79
- `chk_date`, 32
- `chk_date()`, 29, 30, 34, 43, 45, 54, 70, 80, 83, 86, 87, 93
- `chk_date_time`, 33
- `chk_date_time()`, 29, 30, 32, 43, 45, 54, 70, 80, 83, 86, 87, 93
- `chk_datetime(chk_date_time)`, 33
- `chk_dbl`, 34
- `chk_dbl()`, 24, 95
- `chk_deprecated`, 24, 35, 95
- `chk_dir`, 35
- `chk_dir()`, 41, 44
- `chk_double`, 36
- `chk_double()`, 22, 23, 28, 38, 42, 50, 56, 70, 71, 76
- `chk_environment`, 37
- `chk_environment()`, 22, 23, 28, 37, 42, 50, 56, 70, 71, 76
- `chk_equal`, 38
- `chk_equal()`, 16–18, 40, 49
- `chk_equivalent`, 39
- `chk_equivalent()`, 16–18, 39, 49
- `chk_ext`, 40
- `chk_ext()`, 36, 44
- `chk_factor`, 41
- `chk_factor()`, 22, 23, 28, 37, 38, 50, 56, 70, 71, 76
- `chk_false`, 42
- `chk_false()`, 29, 30, 32, 34, 45, 54, 56, 70, 80, 83, 86, 87, 93
- `chk_file`, 43
- `chk_file()`, 36, 41
- `chk_flag`, 44
- `chk_flag()`, 29, 30, 32, 34, 43, 54, 56, 70, 80, 83, 86, 87, 93
- `chk_function`, 45
- `chk_function()`, 61, 66, 89, 90
- `chk_gt`, 46
- `chk_gt()`, 48, 57, 58, 75
- `chk_gte`, 47
- `chk_gte()`, 47, 57, 58, 75
- `chk_identical`, 48
- `chk_identical()`, 16–18, 39, 40
- `chk_integer`, 49
- `chk_integer()`, 22, 23, 28, 37, 38, 42, 56, 70, 71, 76
- `chk_is`, 50
- `chk_is()`, 25, 31, 74, 77–79
- `chk_join`, 51
- `chk_join()`, 19, 64, 65, 88
- `chk_length`, 52
- `chk_length()`, 26
- `chk_lgl`, 53
- `chk_lgl()`, 29, 30, 32, 34, 43–45, 55, 56, 70, 80, 83, 86, 87, 93
- `chk_list`, 54
- `chk_list()`, 20, 21, 60, 92
- `chk_logical`, 55
- `chk_logical()`, 22, 23, 28, 37, 38, 42, 43, 45, 50, 53, 54, 70, 71, 76, 86
- `chk_lt`, 57
- `chk_lt()`, 47, 48, 58, 75
- `chk_lte`, 58
- `chk_lte()`, 47, 48, 57, 75
- `chk_match`, 59
- `chk_matrix`, 60
- `chk_matrix()`, 20, 21, 55, 92
- `chk_missing`, 61
- `chk_missing()`, 46, 66
- `chk_named`, 62
- `chk_named()`, 91
- `chk_not_any_na`, 63
- `chk_not_any_na()`, 19, 52, 65, 88
- `chk_not_empty`, 64

- chk_not_empty(), [19](#), [52](#), [64](#), [88](#)
- chk_not_missing, [65](#)
- chk_not_missing(), [46](#), [61](#)
- chk_not_null, [66](#)
- chk_not_null(), [69](#)
- chk_not_subset, [67](#)
- chk_not_subset(), [72](#), [81](#), [84](#), [85](#)
- chk_null, [68](#)
- chk_null(), [67](#)
- chk_null_or, [69](#)
- chk_null_or(), [13](#), [14](#)
- chk_number, [70](#)
- chk_number(), [22](#), [23](#), [28–30](#), [32](#), [34](#), [37](#), [38](#), [42](#), [43](#), [45](#), [50](#), [54](#), [56](#), [71](#), [76](#), [80](#), [83](#), [86](#), [87](#), [93](#)
- chk_numeric, [71](#)
- chk_numeric(), [22](#), [23](#), [28](#), [37](#), [38](#), [42](#), [50](#), [56](#), [70](#), [76](#)
- chk_orderset, [72](#)
- chk_orderset(), [68](#), [81](#), [84](#), [85](#)
- chk_r6_class, [73](#)
- chk_r6_class(), [25](#), [31](#), [51](#), [77–79](#)
- chk_range, [74](#)
- chk_range(), [47](#), [48](#), [57](#), [58](#)
- chk_raw, [75](#)
- chk_raw(), [22](#), [23](#), [28](#), [37](#), [38](#), [42](#), [50](#), [56](#), [70](#), [71](#)
- chk_s3_class, [76](#)
- chk_s3_class(), [25](#), [31](#), [51](#), [74](#), [78](#), [79](#)
- chk_s3_class_strict, [77](#)
- chk_s3_class_strict(), [25](#), [31](#), [51](#), [74](#), [77](#), [79](#)
- chk_s4_class, [78](#)
- chk_s4_class(), [25](#), [31](#), [51](#), [74](#), [77](#), [78](#)
- chk_scalar, [79](#)
- chk_scalar(), [29](#), [30](#), [32](#), [34](#), [43](#), [45](#), [54](#), [70](#), [83](#), [86](#), [87](#), [93](#)
- chk_setequal, [80](#)
- chk_setequal(), [68](#), [72](#), [84](#), [85](#)
- chk_sorted, [81](#)
- chk_string, [82](#)
- chk_string(), [29](#), [30](#), [32](#), [34](#), [43](#), [45](#), [54](#), [70](#), [80](#), [86](#), [87](#), [93](#)
- chk_subset, [83](#)
- chk_subset(), [68](#), [72](#), [81](#), [84](#), [85](#)
- chk_superset, [84](#)
- chk_superset(), [68](#), [72](#), [81](#), [83](#), [84](#)
- chk_true, [85](#)
- chk_true(), [29](#), [30](#), [32](#), [34](#), [43](#), [45](#), [54](#), [56](#), [70](#), [80](#), [83](#), [87](#), [93](#)
- chk_tz, [86](#)
- chk_tz(), [29](#), [30](#), [32](#), [34](#), [43](#), [45](#), [54](#), [70](#), [80](#), [83](#), [86](#), [93](#)
- chk_unique, [87](#)
- chk_unique(), [19](#), [52](#), [64](#), [65](#)
- chk_unused, [89](#)
- chk_unused(), [46](#), [90](#)
- chk_used, [90](#)
- chk_used(), [46](#), [89](#)
- chk_valid_name, [91](#)
- chk_valid_name(), [63](#)
- chk_vector, [92](#)
- chk_vector(), [20](#), [21](#), [55](#), [60](#)
- chk_whole_number, [93](#)
- chk_whole_number(), [29](#), [30](#), [32](#), [34](#), [43](#), [45](#), [54](#), [70](#), [80](#), [83](#), [86](#), [87](#), [94](#)
- chk_whole_numeric, [94](#)
- chk_whole_numeric(), [30](#), [93](#)
- chk_wnum, [95](#)
- chk_wnum(), [24](#), [35](#)
- chkor, [13](#)
- chkor_vld, [13](#)
- defused function call, [4](#), [97](#)
- deparse1(), [96](#)
- deparse_backtick_chk, [96](#)
- dir.exists(), [36](#), [44](#)
- err, [97](#)
- err(), [4](#)
- expect_chk_error, [98](#)
- expect_chk_error(), [98](#)
- expect_length, [100](#)
- expect_match, [99](#), [100](#)
- expect_named, [100](#)
- expect_no_error(), [99](#), [100](#)
- expect_null, [100](#)
- expect_output, [100](#)
- expect_reference, [100](#)
- expect_silent, [100](#)
- expect_snapshot(), [99](#)
- file.exists(), [44](#)
- formals(), [46](#)
- generate_check_data, [100](#)
- grepl(), [59](#)

- identical(), [49](#), [91](#)
- Including function calls in error messages, [4](#), [97](#)
- inherits(), [25](#), [31](#), [32](#), [34](#), [51](#), [74](#), [77](#), [78](#)
- is.array(), [20](#), [92](#)
- is.atomic(), [21](#), [92](#)
- is.character(), [22](#), [23](#), [87](#)
- is.complex(), [28](#), [29](#)
- is.double(), [37](#), [94](#)
- is.environment(), [38](#)
- is.factor(), [23](#), [42](#)
- is.function(), [46](#)
- is.integer(), [50](#), [93](#), [94](#)
- is.list(), [55](#), [92](#)
- is.logical(), [43](#), [45](#), [54](#), [56](#), [86](#)
- is.matrix(), [60](#), [92](#)
- is.null(), [63](#), [67](#), [69](#)
- is.numeric(), [70](#), [71](#)
- is.raw(), [76](#)
- is.unsorted(), [82](#)
- length(), [16–18](#), [29](#), [30](#), [32](#), [34](#), [43](#), [45](#), [53](#), [54](#), [65](#), [68](#), [70](#), [80](#), [83](#), [86](#), [87](#), [89](#), [90](#)
- logical, [102](#)
- make.names(), [91](#)
- message_chk, [101](#)
- message_chk(), [97](#)
- methods::is(), [79](#)
- missing(), [61](#), [66](#)
- msg(err), [97](#)
- NA_character_, [102](#)
- names(), [63](#)
- OlsonNames(), [87](#)
- p, [102](#)
- p0(p), [102](#)
- quasi_label, [99](#)
- R6::R6Class(), [73](#)
- rlang::abort(), [97](#)
- rlang::inform(), [97](#)
- rlang::warn(), [97](#)
- setequal(), [81](#)
- tolower(), [40](#)
- toupper(), [40](#)
- typeof(), [77](#)
- unbacktick_chk (deparse_backtick_chk), [96](#)
- unique(), [6](#), [72](#)
- vld_all (chk_all), [14](#)
- vld_all_equal (chk_all_equal), [15](#)
- vld_all_equivalent (chk_all_equivalent), [16](#)
- vld_all_identical (chk_all_identical), [17](#)
- vld_all_na (chk_all_na), [18](#)
- vld_array (chk_array), [19](#)
- vld_atomic (chk_atomic), [20](#)
- vld_character (chk_character), [21](#)
- vld_character_or_factor (chk_character_or_factor), [22](#)
- vld_chr (chk_chr), [24](#)
- vld_class (chk_class), [25](#)
- vld_compatible_lengths (chk_compatible_lengths), [26](#)
- vld_complex (chk_complex), [27](#)
- vld_complex_number (chk_complex_number), [29](#)
- vld_count (chk_count), [30](#)
- vld_data (chk_data), [31](#)
- vld_date (chk_date), [32](#)
- vld_date_time (chk_date_time), [33](#)
- vld_datetime (chk_date_time), [33](#)
- vld_dbl (chk_dbl), [34](#)
- vld_dir (chk_dir), [35](#)
- vld_double (chk_double), [36](#)
- vld_environment (chk_environment), [37](#)
- vld_equal (chk_equal), [38](#)
- vld_equal(), [16](#)
- vld_equivalent (chk_equivalent), [39](#)
- vld_equivalent(), [17](#), [72](#)
- vld_ext (chk_ext), [40](#)
- vld_factor (chk_factor), [41](#)
- vld_false (chk_false), [42](#)
- vld_file (chk_file), [43](#)
- vld_flag (chk_flag), [44](#)
- vld_function (chk_function), [45](#)
- vld_gt (chk_gt), [46](#)
- vld_gte (chk_gte), [47](#)
- vld_identical (chk_identical), [48](#)
- vld_identical(), [18](#)

`vld_integer (chk_integer)`, 49
`vld_is (chk_is)`, 50
`vld_join (chk_join)`, 51
`vld_length (chk_length)`, 52
`vld_lgl (chk_lgl)`, 53
`vld_list (chk_list)`, 54
`vld_logical (chk_logical)`, 55
`vld_lt (chk_lt)`, 57
`vld_lte (chk_lte)`, 58
`vld_match (chk_match)`, 59
`vld_matrix (chk_matrix)`, 60
`vld_missing (chk_missing)`, 61
`vld_named (chk_named)`, 62
`vld_not_any_na (chk_not_any_na)`, 63
`vld_not_empty (chk_not_empty)`, 64
`vld_not_missing (chk_not_missing)`, 65
`vld_not_null (chk_not_null)`, 66
`vld_not_subset (chk_not_subset)`, 67
`vld_null (chk_null)`, 68
`vld_number (chk_number)`, 70
`vld_number()`, 93
`vld_numeric (chk_numeric)`, 71
`vld_orderset (chk_orderset)`, 72
`vld_r6_class (chk_r6_class)`, 73
`vld_range (chk_range)`, 74
`vld_raw (chk_raw)`, 75
`vld_s3_class (chk_s3_class)`, 76
`vld_s3_class_strict (chk_s3_class_strict)`, 77
`vld_s4_class (chk_s4_class)`, 78
`vld_scalar (chk_scalar)`, 79
`vld_setequal (chk_setequal)`, 80
`vld_sorted (chk_sorted)`, 81
`vld_string (chk_string)`, 82
`vld_string()`, 36, 41, 44
`vld_subset (chk_subset)`, 83
`vld_subset()`, 41
`vld_superset (chk_superset)`, 84
`vld_true (chk_true)`, 85
`vld_true()`, 39, 40, 93, 94
`vld_tz (chk_tz)`, 86
`vld_unique (chk_unique)`, 87
`vld_unused (chk_unused)`, 89
`vld_used (chk_used)`, 90
`vld_valid_name (chk_valid_name)`, 91
`vld_vector (chk_vector)`, 92
`vld_whole_number (chk_whole_number)`, 93
`vld_whole_number()`, 30
`vld_whole_numeric (chk_whole_numeric)`, 94
`vld_wnum (chk_wnum)`, 95
`wrn (err)`, 97