

# Package ‘chk’

May 8, 2026

**Title** Check User-Supplied Function Arguments

**Version** 0.10.0

**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 (>= 3.6)

**Imports** lifecycle, methods, rlang, tools

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

**VignetteBuilder** knitr

**RdMacros** lifecycle

**Config/Needs/website** poissonconsulting/poissontemplate

**Config/testthat/edition** 3

**Encoding** UTF-8

**Language** en-US

**RoxygenNote** 7.3.2.9000

**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>>),  
Florencia 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>>),  
Poisson Consulting [cph, fnd]

**Maintainer** Joe Thorley <joe@poissonconsulting.ca>

**Repository** CRAN

**Date/Publication** 2025-01-24 21:20:05 UTC

## Contents

abort_chk . . . . .	4
cc . . . . .	5
check_data . . . . .	6
check_dim . . . . .	7
check_dirs . . . . .	7
check_files . . . . .	8
check_key . . . . .	9
check_length . . . . .	9
check_names . . . . .	10
check_values . . . . .	11
chkor . . . . .	12
chkor_vld . . . . .	13
chk_all . . . . .	13
chk_all_equal . . . . .	15
chk_all_equivalent . . . . .	16
chk_all_identical . . . . .	17
chk_array . . . . .	18
chk_atomic . . . . .	19
chk_character . . . . .	20
chk_character_or_factor . . . . .	21
chk_chr . . . . .	22
chk_compatible_lengths . . . . .	23
chk_complex . . . . .	25
chk_complex_number . . . . .	26
chk_count . . . . .	27
chk_data . . . . .	28
chk_date . . . . .	29
chk_date_time . . . . .	30
chk_dbl . . . . .	31
chk_dir . . . . .	32
chk_double . . . . .	33
chk_environment . . . . .	34
chk_equal . . . . .	35
chk_equivalent . . . . .	36
chk_ext . . . . .	37
chk_factor . . . . .	38
chk_false . . . . .	39
chk_file . . . . .	40
chk_flag . . . . .	41
chk_function . . . . .	42
chk_gt . . . . .	43
chk_gte . . . . .	44
chk_identical . . . . .	45
chk_integer . . . . .	46
chk_is . . . . .	47
chk_join . . . . .	48

chk_length . . . . .	49
chk_lgl . . . . .	50
chk_list . . . . .	51
chk_logical . . . . .	52
chk_lt . . . . .	54
chk_lte . . . . .	55
chk_match . . . . .	56
chk_matrix . . . . .	57
chk_missing . . . . .	58
chk_named . . . . .	59
chk_not_any_na . . . . .	60
chk_not_empty . . . . .	61
chk_not_missing . . . . .	62
chk_not_null . . . . .	63
chk_not_subset . . . . .	64
chk_null . . . . .	65
chk_null_or . . . . .	66
chk_number . . . . .	67
chk_numeric . . . . .	68
chk_ordinal . . . . .	69
chk_range . . . . .	70
chk_raw . . . . .	71
chk_s3_class . . . . .	72
chk_s4_class . . . . .	73
chk_scalar . . . . .	74
chk_sorted . . . . .	75
chk_string . . . . .	76
chk_superset . . . . .	77
chk_true . . . . .	78
chk_tz . . . . .	79
chk_unique . . . . .	80
chk_unused . . . . .	81
chk_used . . . . .	82
chk_valid_name . . . . .	83
chk_vector . . . . .	84
chk_whole_number . . . . .	85
chk_whole_numeric . . . . .	86
chk_wnum . . . . .	87
deparse_backtick_chk . . . . .	88
err . . . . .	89
expect_chk_error . . . . .	90
message_chk . . . . .	92
p . . . . .	93
vld_not_subset . . . . .	94
vld_ordinal . . . . .	95

---

 abort\_chk

*Abort Check*


---

### 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 <a href="#">defused function call</a> to respectively not display any call or hard-code a code to display. For more information about error calls, see <a href="#">Including function calls in error messages</a> .

### 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))
```

**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.

### 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	<i>Check Directories Exist</i>
------------	--------------------------------

---

**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

*Check Files Exist*

---

**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	<i>Check Length</i>
--------------	---------------------

---

**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 ensure values has two levels to check that the levels of x are an ordered superset of the levels of value and three or more levels to check that they are identical.

**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

*Check OR*

---

**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(...)
```

**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.

**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))
```

---

chk_all	<i>Check All</i>
---------	------------------

---

**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

x	The object to check.
tolerance	A non-negative numeric scalar.
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_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))
```

---

chk\_all\_equivalent      *Check All Equivalent*

---

**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**

x	The object to check.
tolerance	A non-negative numeric scalar.
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_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, 1.00000001)))
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))
```

---

chk_all_identical	<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, 1.00000001), 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\_array

*Check Array*


---

**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	<i>Check Atomic</i>
------------	---------------------

---

**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\\_integer\(\)](#), [chk\\_logical\(\)](#), [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**

<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_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\\_integer\(\)](#), [chk\\_logical\(\)](#), [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(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

*Check Character Scalar*

---

**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_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 <code>x</code> or <code>NULL</code> .

**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\\_integer\(\)](#), [chk\\_logical\(\)](#), [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      *Check Complex Number*

---

**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.integer\(\)](#)[vld\\_true\(\)](#)[vld\\_number\(\)](#)

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

Other scalar\_checker: [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

*Check Count*

---

**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\\_date\(\)](#), [chk\\_date\\_time\(\)](#), [chk\\_false\(\)](#), [chk\\_flag\(\)](#), [chk\\_lgl\(\)](#), [chk\\_scalar\(\)](#), [chk\\_string\(\)](#), [chk\\_true\(\)](#), [chk\\_tz\(\)](#)

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\\_is\(\)](#), [chk\\_s3\\_class\(\)](#), [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\\_count\(\)](#), [chk\\_date\\_time\(\)](#), [chk\\_false\(\)](#), [chk\\_flag\(\)](#), [chk\\_lgl\(\)](#), [chk\\_scalar\(\)](#), [chk\\_string\(\)](#), [chk\\_true\(\)](#), [chk\\_tz\(\)](#)

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

*Check Date Time*


---

**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\\_count\(\)](#), [chk\\_date\(\)](#), [chk\\_false\(\)](#), [chk\\_flag\(\)](#), [chk\\_lgl\(\)](#), [chk\\_scalar\(\)](#), [chk\\_string\(\)](#), [chk\\_true\(\)](#), [chk\\_tz\(\)](#)

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**

<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_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

*Check Directory Exists*

---

**Description**

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

**Usage**

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

```
vld_dir(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_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\\_integer\(\)](#), [chk\\_logical\(\)](#), [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**

`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_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\\_integer\(\)](#), [chk\\_logical\(\)](#), [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 `x` 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)
```

---

chk\_equivalent

*Check Equivalent*

---

**Description**

Checks if `x` 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\(\)](#)

**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**

`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_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\\_count\(\)](#), [chk\\_date\(\)](#), [chk\\_date\\_time\(\)](#), [chk\\_flag\(\)](#), [chk\\_lgl\(\)](#), [chk\\_scalar\(\)](#), [chk\\_string\(\)](#), [chk\\_true\(\)](#), [chk\\_tz\(\)](#)

**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\\_count\(\)](#), [chk\\_date\(\)](#), [chk\\_date\\_time\(\)](#), [chk\\_false\(\)](#), [chk\\_lgl\(\)](#), [chk\\_scalar\(\)](#), [chk\\_string\(\)](#), [chk\\_true\(\)](#), [chk\\_tz\(\)](#)

**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 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\\_logical\(\)](#), [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 string specifying the class.
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\\_data\(\)](#), [chk\\_s3\\_class\(\)](#), [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\\_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\\_count\(\)](#), [chk\\_date\(\)](#), [chk\\_date\\_time\(\)](#), [chk\\_false\(\)](#), [chk\\_flag\(\)](#), [chk\\_scalar\(\)](#), [chk\\_string\(\)](#), [chk\\_true\(\)](#), [chk\\_tz\(\)](#)

**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\\_integer\(\)](#), [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

<code>x</code>	The object to check.
<code>value</code>	A non-missing scalar of a value.
<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_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**

x	The object to check.
regex	A string of a regular expression.
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_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

<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_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

*Check Not Any Missing Values*


---

**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_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	<i>Check Not Empty</i>
---------------	------------------------

---

**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\\_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**

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_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

*Check Not Subset*

---

**Description**

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

**Usage**

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

**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 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.

**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\\_superset\(\)](#), [vld\\_not\\_subset\(\)](#), [vld\\_order\\_set\(\)](#)

**Examples**

```
# chk_not_subset
chk_not_subset(11, 1:10)
try(chk_not_subset(1, 1:10))
```

---

chk_null	<i>Check NULL</i>
----------	-------------------

---

**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**

x	The object to check.
chk	A chk function. Soft-deprecated for vld. <b>[Deprecated]</b>
...	Arguments passed to chk.
vld	A vld function.
x_name	A string of the name of object x 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	<i>Check Number</i>
------------	---------------------

---

### 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")`.

### 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**

<code>x</code>	The object to check.
<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_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\\_integer\(\)](#), [chk\\_logical\(\)](#), [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)
```

### 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.

### 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\\_superset\(\)](#), [vld\\_not\\_subset\(\)](#), [vld\\_orderset\(\)](#)

### Examples

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

---

`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**

<code>x</code>	The object to check.
<code>range</code>	A non-missing sorted vector of length 2 of the lower and upper permitted values.
<code>inclusive</code>	A flag specifying whether the range is exclusive.
<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_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\\_integer\(\)](#), [chk\\_logical\(\)](#), [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	<i>Check Type</i>
--------------	-------------------

---

**Description**

Checks inherits from S3 class using  
`!isS4(x) && inherits(x, class)`

**Usage**

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

vld_s3_class(x, class)
```

**Arguments**

x	The object to check.
class	A string specifying the class.
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()`: 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\\_data\(\)](#), [chk\\_is\(\)](#), [chk\\_s4\\_class\(\)](#)

**Examples**

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

---

chk_s4_class	<i>Check Inherits from S4 Class</i>
--------------	-------------------------------------

---

**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 string specifying the class.
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\\_data\(\)](#), [chk\\_is\(\)](#), [chk\\_s3\\_class\(\)](#)

## 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 vector 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\\_count\(\)](#), [chk\\_date\(\)](#), [chk\\_date\\_time\(\)](#), [chk\\_false\(\)](#), [chk\\_flag\(\)](#), [chk\\_lgl\(\)](#), [chk\\_string\(\)](#), [chk\\_true\(\)](#), [chk\\_tz\(\)](#)

**Examples**

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

---

chk_sorted	<i>Check Sorted</i>
------------	---------------------

---

**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	<i>Check String</i>
------------	---------------------

---

## 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\\_count\(\)](#), [chk\\_date\(\)](#), [chk\\_date\\_time\(\)](#), [chk\\_false\(\)](#), [chk\\_flag\(\)](#), [chk\\_lgl\(\)](#), [chk\\_scalar\(\)](#), [chk\\_true\(\)](#), [chk\\_tz\(\)](#)

## 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\_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\\_orderset\(\)](#), [vld\\_not\\_subset\(\)](#), [vld\\_orderset\(\)](#)

**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\\_count\(\)](#), [chk\\_date\(\)](#), [chk\\_date\\_time\(\)](#), [chk\\_false\(\)](#), [chk\\_flag\(\)](#), [chk\\_lgl\(\)](#), [chk\\_scalar\(\)](#), [chk\\_string\(\)](#), [chk\\_tz\(\)](#)

## 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	<i>Check Time Zone</i>
--------	------------------------

---

## 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\\_count\(\)](#), [chk\\_date\(\)](#), [chk\\_date\\_time\(\)](#), [chk\\_false\(\)](#), [chk\\_flag\(\)](#), [chk\\_lgl\(\)](#), [chk\\_scalar\(\)](#), [chk\\_string\(\)](#), [chk\\_true\(\)](#)

**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\\_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\\_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

*Check ... Used*


---

**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\\_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\_checker: [chk\\_complex\\_number\(\)](#)

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      *Check Whole Numeric*

---

**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

---

*Check Whole Numeric Scalar*


---

**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 [deparse\(\)](#) 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**

[deparse\(\)](#)

**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.

`call` The execution environment of a currently running function, e.g. `call = caller_env()`. The corresponding function call is retrieved and mentioned in error messages as the source of the error.

You only need to supply `call` when throwing a condition from a helper function which wouldn't be relevant to mention in the message.

Can also be `NULL` 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.

### 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 <a href="#">quasi_label</a> for more details.
regexp	Regular expression to test against. <ul style="list-style-type: none"> <li>• A character vector giving a regular expression that must match the error message.</li> <li>• If NULL, the default, asserts that there should be an error, but doesn't test for a specific value.</li> <li>• If NA, asserts that there should be no errors, but we now recommend using <a href="#">expect_no_error()</a> and friends instead.</li> </ul> <p>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>.</p>
...	Arguments passed on to <a href="#">expect_match</a>
	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 <a href="#">quasi_label</a> .
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)))
```

---

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'
r	'is' if n == 1 otherwise 'are'
y	'y' if n == 1 otherwise 'ie'

**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)
```

p

*Concatenate Strings***Description**

A wrapper on `base::paste()`.

**Usage**

```
p(..., sep = " ", collapse = NULL)
```

```
p0(..., collapse = NULL)
```

**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).

**Value**

A character vector.

**Functions**

- `p0()`: A wrapper on `base::paste0()`

**Examples**

```
p("a", "b")
p(c("a", "b"), collapse = " ")
p0("a", "b")
p0(c("a", "b"), collapse = "")
```

---

vld_not_subset	<i>Check Subset</i>
----------------	---------------------

---

### 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

```
vld_not_subset(x, values)
```

```
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_not_subset()`: Validate Not Subset
- `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\\_orderset\(\)](#), [chk\\_superset\(\)](#), [vld\\_orderset\(\)](#)

**Examples**

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

---

vld_orderset	<i>Check Set Equal</i>
--------------	------------------------

---

**Description**

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

**Usage**

```
vld_orderset(x, values)

chk_setequal(x, values, x_name = NULL)

vld_setequal(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 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_orderset()`: Validate Set Ordered
- `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\\_orderset\(\)](#), [chk\\_superset\(\)](#), [vld\\_not\\_subset\(\)](#)

**Examples**

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

# Index

- \* **all\_checkers**
  - chk\_all, 13
  - chk\_all\_equal, 15
  - chk\_all\_equivalent, 16
  - chk\_all\_identical, 17
- \* **check**
  - check\_data, 6
  - check\_dim, 7
  - check\_dirs, 7
  - check\_files, 8
  - check\_key, 9
  - check\_length, 9
  - check\_names, 10
  - check\_values, 11
- \* **chk\_character**
  - chk\_chr, 22
- \* **data-type\_checkers**
  - chk\_factor, 38
- \* **data\_structure\_checkers**
  - chk\_array, 18
  - chk\_atomic, 19
  - chk\_list, 51
  - chk\_matrix, 57
  - chk\_vector, 84
- \* **data\_type\_checkers scalar\_checkers**
  - chk\_number, 67
- \* **data\_type\_checkers**
  - chk\_character, 20
  - chk\_character\_or\_factor, 21
  - chk\_complex, 25
  - chk\_double, 33
  - chk\_environment, 34
  - chk\_integer, 46
  - chk\_logical, 52
  - chk\_numeric, 68
  - chk\_raw, 71
- \* **date\_checkers**
  - chk\_tz, 79
- \* **datetime\_checkers**
  - chk\_date, 29
  - chk\_date\_time, 30
- \* **deprecated**
  - chk\_chr, 22
  - chk\_dbl, 31
  - chk\_wnum, 87
- \* **ellipsis\_checkers**
  - chk\_unused, 81
  - chk\_used, 82
- \* **ellipsis\_checkers**
  - chk\_function, 42
- \* **equal\_checkers**
  - chk\_all\_equal, 15
  - chk\_all\_equivalent, 16
  - chk\_all\_identical, 17
  - chk\_equal, 35
  - chk\_equivalent, 36
  - chk\_identical, 45
- \* **factor\_checkers**
  - chk\_character\_or\_factor, 21
  - chk\_factor, 38
- \* **file\_checkers**
  - chk\_dir, 32
  - chk\_ext, 37
  - chk\_file, 40
- \* **function\_checkers**
  - chk\_function, 42
- \* **id\_checkers**
  - chk\_data, 28
  - chk\_is, 47
  - chk\_s3\_class, 72
  - chk\_s4\_class, 73
- \* **length\_checkers**
  - chk\_compatible\_lengths, 23
  - chk\_length, 49
- \* **logical\_checkers**
  - chk\_false, 39
  - chk\_flag, 41
  - chk\_lgl, 50

- chk\_logical, 52
- chk\_true, 78
- \* **misc\_checkers**
  - chk\_join, 48
  - chk\_not\_any\_na, 60
  - chk\_not\_empty, 61
  - chk\_unique, 80
- \* **missing\_checkers**
  - chk\_function, 42
  - chk\_missing, 58
  - chk\_not\_missing, 62
- \* **name\_checkers**
  - chk\_named, 59
  - chk\_valid\_name, 83
- \* **null\_checkers**
  - chk\_not\_null, 63
  - chk\_null, 65
- \* **range\_checkers**
  - chk\_gt, 43
  - chk\_gte, 44
  - chk\_lt, 54
  - chk\_lte, 55
  - chk\_range, 70
- \* **regex\_checkers**
  - chk\_match, 56
- \* **scalar\_checkers**
  - chk\_count, 27
  - chk\_date, 29
  - chk\_date\_time, 30
  - chk\_false, 39
  - chk\_flag, 41
  - chk\_lgl, 50
  - chk\_scalar, 74
  - chk\_string, 76
  - chk\_true, 78
  - chk\_tz, 79
- \* **scalar\_checker**
  - chk\_complex\_number, 26
  - chk\_whole\_number, 85
- \* **set\_checkers**
  - chk\_not\_subset, 64
  - chk\_ordinalset, 69
  - chk\_superset, 77
  - vld\_not\_subset, 94
  - vld\_ordinalset, 95
- \* **sorted\_checkers**
  - chk\_sorted, 75
- \* **tz\_checkers**
  - chk\_tz, 79
- \* **whole\_number\_checkers**
  - chk\_count, 27
  - chk\_whole\_number, 85
  - chk\_whole\_numeric, 86
- abort\_chk, 4
- all(), 14, 44, 45, 54–56, 70, 77, 94
- all.equal(), 36, 37, 87
- any(), 65
- anyDuplicated(), 80
- backtick\_chk (deparse\_backtick\_chk), 88
- base::paste(), 93
- base::paste0(), 93
- cc, 5
- character, 93
- check\_data, 6, 7–12
- check\_data(), 28
- check\_dim, 6, 7, 8–12
- check\_dim(), 50
- check\_dirs, 6, 7, 7, 8–12
- check\_files, 6–8, 8, 9–12
- check\_key, 6–8, 9, 10–12
- check\_length, 6–9, 9, 11, 12
- check\_length(), 50
- check\_names, 6–10, 10, 12
- check\_values, 6–11, 11
- chk\_all, 13, 15, 16, 18
- chk\_all\_equal, 14, 15, 16, 18, 36, 37, 46
- chk\_all\_equivalent, 14, 15, 16, 18, 36, 37, 46
- chk\_all\_identical, 14–16, 17, 36, 37, 46
- chk\_array, 18, 20, 52, 57, 85
- chk\_atomic, 19, 19, 52, 57, 85
- chk\_character, 20, 22, 26, 34, 35, 47, 53, 68, 71
- chk\_character\_or\_factor, 21, 21, 26, 34, 35, 39, 47, 53, 68, 71
- chk\_chr, 22, 32, 88
- chk\_compatible\_lengths, 23, 50
- chk\_complex, 21, 22, 25, 34, 35, 47, 53, 68, 71
- chk\_complex\_number, 26, 86
- chk\_count, 27, 30, 31, 40, 42, 51, 74, 76, 78, 79, 86, 87
- chk\_data, 28, 48, 72, 73
- chk\_date, 28, 29, 31, 40, 42, 51, 74, 76, 78, 79

- chk\_date\_time, 28, 30, 30, 40, 42, 51, 74, 76, 78, 79
- chk\_datetime (chk\_date\_time), 30
- chk\_dbl, 23, 31, 88
- chk\_deprecated, 23, 32, 88
- chk\_dir, 32, 38, 41
- chk\_double, 21, 22, 26, 33, 35, 47, 53, 68, 71
- chk\_environment, 21, 22, 26, 34, 34, 47, 53, 68, 71
- chk\_equal, 15, 16, 18, 35, 37, 46
- chk\_equivalent, 15, 16, 18, 36, 36, 46
- chk\_ext, 33, 37, 41
- chk\_factor, 22, 38
- chk\_false, 28, 30, 31, 39, 42, 51, 53, 74, 76, 78, 79
- chk\_file, 33, 38, 40
- chk\_flag, 28, 30, 31, 40, 41, 51, 53, 74, 76, 78, 79
- chk\_function, 42, 58, 63
- chk\_gt, 43, 45, 54, 55, 70
- chk\_gte, 44, 44, 54, 55, 70
- chk\_identical, 15, 16, 18, 36, 37, 45
- chk\_integer, 21, 22, 26, 34, 35, 46, 53, 68, 71
- chk\_is, 29, 47, 72, 73
- chk\_join, 48, 61, 62, 80
- chk\_length, 24, 49
- chk\_lgl, 28, 30, 31, 40, 42, 50, 53, 74, 76, 78, 79
- chk\_lgl(), 41, 52
- chk\_list, 19, 20, 51, 57, 85
- chk\_logical, 21, 22, 26, 34, 35, 40, 42, 47, 51, 52, 68, 71, 78
- chk\_logical(), 50
- chk\_lt, 44, 45, 54, 55, 70
- chk\_lte, 44, 45, 54, 55, 70
- chk\_match, 56
- chk\_matrix, 19, 20, 52, 57, 85
- chk\_missing, 43, 58, 63
- chk\_named, 59, 84
- chk\_not\_any\_na, 49, 60, 62, 80
- chk\_not\_empty, 49, 61, 61, 80
- chk\_not\_missing, 43, 58, 62
- chk\_not\_null, 63, 65
- chk\_not\_subset, 64, 69, 77, 94, 96
- chk\_null, 64, 65
- chk\_null\_or, 66
- chk\_null\_or(), 12, 13
- chk\_number, 67
- chk\_numeric, 21, 22, 26, 34, 35, 47, 53, 68, 71
- chk\_orderset, 65, 69, 77, 94, 96
- chk\_range, 44, 45, 54, 55, 70
- chk\_raw, 21, 22, 26, 34, 35, 47, 53, 68, 71
- chk\_s3\_class, 29, 48, 72, 73
- chk\_s4\_class, 29, 48, 72, 73
- chk\_scalar, 28, 30, 31, 40, 42, 51, 74, 76, 78, 79
- chk\_setequal (vld\_orderset), 95
- chk\_sorted, 75
- chk\_string, 28, 30, 31, 40, 42, 51, 74, 76, 78, 79
- chk\_subset (vld\_not\_subset), 94
- chk\_subset(), 77
- chk\_superset, 65, 69, 77, 94, 96
- chk\_superset(), 94
- chk\_true, 28, 30, 31, 40, 42, 51, 53, 74, 76, 78, 79
- chk\_tz, 28, 30, 31, 40, 42, 51, 74, 76, 78, 79
- chk\_unique, 49, 61, 62, 80
- chk\_unused, 81, 82
- chk\_used, 81, 82
- chk\_valid\_name, 60, 83
- chk\_vector, 19, 20, 52, 57, 84
- chk\_whole\_number, 27, 28, 85, 87
- chk\_whole\_numeric, 28, 86, 86
- chk\_wnum, 23, 32, 87
- chkor, 12
- chkor\_vld, 13
- defused function call, 4, 90
- deparse(), 88, 89
- deparse\_backtick\_chk, 88
- dir.exists(), 33, 41
- err, 89
- err(), 4
- expect\_chk\_error, 90
- expect\_chk\_error(), 90
- expect\_length, 92
- expect\_match, 91, 92
- expect\_named, 92
- expect\_no\_error(), 91, 92
- expect\_null, 92
- expect\_output, 92
- expect\_reference, 92
- expect\_silent, 92
- expect\_snapshot(), 91

- file.exists(), [41](#)
- formals(), [43](#)
- grepl(), [56](#)
- identical(), [46](#), [84](#)
- Including function calls in error
  - messages, [4](#), [90](#)
- inherits(), [29–31](#), [48](#), [72](#)
- is.array(), [19](#), [85](#)
- is.atomic(), [20](#), [85](#)
- is.character(), [21](#), [22](#), [79](#)
- is.complex(), [26](#)
- is.double(), [34](#), [87](#)
- is.environment(), [35](#)
- is.factor(), [22](#), [39](#)
- is.function(), [43](#)
- is.integer(), [27](#), [47](#), [86](#), [87](#)
- is.list(), [52](#), [85](#)
- is.logical(), [40](#), [42](#), [51](#), [53](#), [78](#)
- is.matrix(), [57](#), [85](#)
- is.null(), [60](#), [64](#), [65](#)
- is.numeric(), [67](#), [68](#)
- is.raw(), [71](#)
- is.unsorted(), [75](#)
- length(), [15](#), [16](#), [18](#), [28](#), [30](#), [31](#), [40](#), [42](#), [50](#), [51](#), [62](#), [65](#), [67](#), [74](#), [76](#), [78](#), [79](#), [81](#), [82](#)
- make.names(), [84](#)
- message\_chk, [92](#)
- message\_chk(), [89](#)
- methods::is(), [73](#)
- missing(), [58](#), [63](#)
- msg(err), [89](#)
- NA\_character\_, [93](#)
- names(), [60](#)
- OlsonNames(), [79](#)
- p, [93](#)
- p0 (p), [93](#)
- quasi\_label, [91](#)
- rlang::abort(), [89](#)
- rlang::inform(), [89](#)
- rlang::warn(), [89](#)
- setequal(), [96](#)
- tolower(), [37](#)
- toupper(), [37](#)
- unbacktick\_chk (deparse\_backtick\_chk), [88](#)
- unique(), [69](#)
- vld\_all (chk\_all), [13](#)
- vld\_all\_equal (chk\_all\_equal), [15](#)
- vld\_all\_equivalent (chk\_all\_equivalent), [16](#)
- vld\_all\_identical (chk\_all\_identical), [17](#)
- vld\_array (chk\_array), [18](#)
- vld\_atomic (chk\_atomic), [19](#)
- vld\_character (chk\_character), [20](#)
- vld\_character\_or\_factor (chk\_character\_or\_factor), [21](#)
- vld\_chr (chk\_chr), [22](#)
- vld\_compatible\_lengths (chk\_compatible\_lengths), [23](#)
- vld\_complex (chk\_complex), [25](#)
- vld\_complex\_number (chk\_complex\_number), [26](#)
- vld\_count (chk\_count), [27](#)
- vld\_data (chk\_data), [28](#)
- vld\_date (chk\_date), [29](#)
- vld\_date\_time (chk\_date\_time), [30](#)
- vld\_datetime (chk\_date\_time), [30](#)
- vld\_dbl (chk\_dbl), [31](#)
- vld\_dir (chk\_dir), [32](#)
- vld\_double (chk\_double), [33](#)
- vld\_environment (chk\_environment), [34](#)
- vld\_equal (chk\_equal), [35](#)
- vld\_equal(), [15](#)
- vld\_equivalent (chk\_equivalent), [36](#)
- vld\_equivalent(), [16](#), [69](#)
- vld\_ext (chk\_ext), [37](#)
- vld\_factor (chk\_factor), [38](#)
- vld\_false (chk\_false), [39](#)
- vld\_file (chk\_file), [40](#)
- vld\_flag (chk\_flag), [41](#)
- vld\_function (chk\_function), [42](#)
- vld\_gt (chk\_gt), [43](#)
- vld\_gte (chk\_gte), [44](#)
- vld\_identical (chk\_identical), [45](#)
- vld\_identical(), [18](#)
- vld\_integer (chk\_integer), [46](#)
- vld\_is (chk\_is), [47](#)

vld\_join (chk\_join), 48  
vld\_length (chk\_length), 49  
vld\_lgl (chk\_lgl), 50  
vld\_list (chk\_list), 51  
vld\_logical (chk\_logical), 52  
vld\_lt (chk\_lt), 54  
vld\_lte (chk\_lte), 55  
vld\_match (chk\_match), 56  
vld\_matrix (chk\_matrix), 57  
vld\_missing (chk\_missing), 58  
vld\_named (chk\_named), 59  
vld\_not\_any\_na (chk\_not\_any\_na), 60  
vld\_not\_empty (chk\_not\_empty), 61  
vld\_not\_missing (chk\_not\_missing), 62  
vld\_not\_null (chk\_not\_null), 63  
vld\_not\_subset, 65, 69, 77, 94, 96  
vld\_null (chk\_null), 65  
vld\_number (chk\_number), 67  
vld\_number(), 27, 86  
vld\_numeric (chk\_numeric), 68  
vld\_orderiset, 65, 69, 77, 94, 95  
vld\_range (chk\_range), 70  
vld\_raw (chk\_raw), 71  
vld\_s3\_class (chk\_s3\_class), 72  
vld\_s4\_class (chk\_s4\_class), 73  
vld\_scalar (chk\_scalar), 74  
vld\_setequal (vld\_orderiset), 95  
vld\_sorted (chk\_sorted), 75  
vld\_string (chk\_string), 76  
vld\_string(), 33, 38, 41  
vld\_subset (vld\_not\_subset), 94  
vld\_subset(), 38  
vld\_superset (chk\_superset), 77  
vld\_true (chk\_true), 78  
vld\_true(), 27, 36, 37, 86, 87  
vld\_tz (chk\_tz), 79  
vld\_unique (chk\_unique), 80  
vld\_unused (chk\_unused), 81  
vld\_used (chk\_used), 82  
vld\_valid\_name (chk\_valid\_name), 83  
vld\_vector (chk\_vector), 84  
vld\_whole\_number (chk\_whole\_number), 85  
vld\_whole\_number(), 28  
vld\_whole\_numeric (chk\_whole\_numeric),  
86  
vld\_wnum (chk\_wnum), 87  
  
wrn (err), 89