

# Package: chk (via r-universe)

June 26, 2024

**Title** Check User-Supplied Function Arguments

**Version** 0.9.1.9001

**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://poissonconsulting.github.io/chk/>,  
<https://github.com/poissonconsulting/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/testthat/edition** 3

**Encoding** UTF-8

**Language** en-US

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.0

**Repository** <https://poissonconsulting.r-universe.dev>

**RemoteUrl** <https://github.com/poissonconsulting/chk>

**RemoteRef** HEAD

**RemoteSha** ea59f9cd197b81f256f1726b45f91072533a1248

## Contents

abort_chk . . . . .	4
cc . . . . .	5
check_data . . . . .	6
check_dim . . . . .	7
check_dirs . . . . .	7
check_files . . . . .	8
check_key . . . . .	9
check_names . . . . .	9
check_values . . . . .	10
chkor . . . . .	11
chkor_vld . . . . .	12
chk_all . . . . .	13
chk_all_equal . . . . .	14
chk_all_equivalent . . . . .	15
chk_all_identical . . . . .	16
chk_array . . . . .	17
chk_atomic . . . . .	18
chk_character . . . . .	19
chk_character_or_factor . . . . .	20
chk_chr . . . . .	21
chk_compatible_lengths . . . . .	22
chk_count . . . . .	23
chk_data . . . . .	25
chk_date . . . . .	26
chk_date_time . . . . .	27
chk_dbl . . . . .	28
chk_dir . . . . .	29
chk_double . . . . .	30
chk_environment . . . . .	31
chk_equal . . . . .	32
chk_equivalent . . . . .	33
chk_ext . . . . .	34
chk_factor . . . . .	35
chk_false . . . . .	36
chk_file . . . . .	37
chk_flag . . . . .	38
chk_function . . . . .	39
chk_gt . . . . .	40
chk_gte . . . . .	41
chk_identical . . . . .	42
chk_integer . . . . .	43
chk_is . . . . .	44
chk_join . . . . .	45
chk_length . . . . .	46
chk_lgl . . . . .	47
chk_list . . . . .	48

chk_logical . . . . .	49
chk_lt . . . . .	50
chk_lte . . . . .	51
chk_match . . . . .	52
chk_matrix . . . . .	53
chk_missing . . . . .	54
chk_named . . . . .	55
chk_not_any_na . . . . .	56
chk_not_empty . . . . .	57
chk_not_missing . . . . .	58
chk_not_null . . . . .	59
chk_not_subset . . . . .	60
chk_null . . . . .	61
chk_null_or . . . . .	62
chk_number . . . . .	62
chk_numeric . . . . .	63
chk_orderset . . . . .	64
chk_range . . . . .	65
chk_s3_class . . . . .	66
chk_s4_class . . . . .	67
chk_scalar . . . . .	68
chk_sorted . . . . .	69
chk_string . . . . .	70
chk_superset . . . . .	71
chk_true . . . . .	72
chk_tz . . . . .	73
chk_unique . . . . .	74
chk_unused . . . . .	75
chk_used . . . . .	76
chk_valid_name . . . . .	77
chk_vector . . . . .	78
chk_whole_number . . . . .	79
chk_whole_numeric . . . . .	80
chk_wnum . . . . .	81
deparse_backtick_chk . . . . .	82
err . . . . .	83
expect_chk_error . . . . .	84
message_chk . . . . .	86
p . . . . .	87
vld_not_subset . . . . .	87
vld_orderset . . . . .	89

---

 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

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

### 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\\_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\\_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\\_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\\_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\\_names\(\)](#), [check\\_values\(\)](#)

**Examples**

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

---

check_names	<i>Check Names</i>
-------------	--------------------

---

**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\\_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\\_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

*Chk OR*

---

**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 <code>chk_function</code> .
...	Additional arguments.
x_name	A string of the name of object x or NULL.
vld_fun	A <code>vld_function</code> .

**Value**

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

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

**Functions**

- `vld_all()`: Validate All

**See Also**

Other `chk_all`s: [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**

Other `chk_`alls: [chk\\_all\\_equivalent\(\)](#), [chk\\_all\\_identical\(\)](#), [chk\\_all\(\)](#)

## 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	<i>Check All Equivalent</i>
--------------------	-----------------------------

---

## Description

Checks all elements in x equivalent using

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

## Usage

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

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

## Arguments

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

Other `chk_`alls: [chk\\_all\\_equal\(\)](#), [chk\\_all\\_identical\(\)](#), [chk\\_all\(\)](#)

**Examples**

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

---

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, 1.0000001), list(1, NA)

**Usage**

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

```
vld_all_identical(x)
```

**Arguments**

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

**Value**

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

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

**Functions**

- `vld_all_identical()`: Validate All Identical

**See Also**

Other `chk_all`s: [chk\\_all\\_equal\(\)](#), [chk\\_all\\_equivalent\(\)](#), [chk\\_all\(\)](#)



**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	<i>Check Array</i>
-----------	--------------------

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

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

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

**Functions**

- `vld_array()`: Validate Array

**See Also**

Other `chk_` is: [chk\\_atomic\(\)](#), [chk\\_data\(\)](#), [chk\\_function\(\)](#), [chk\\_is\(\)](#), [chk\\_matrix\(\)](#), [chk\\_numeric\(\)](#), [chk\\_s3\\_class\(\)](#), [chk\\_s4\\_class\(\)](#), [chk\\_vector\(\)](#), [chk\\_whole\\_numeric\(\)](#)

## Examples

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

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

---

chk\_atomic

*Check Atomic*

---

## Description

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

## Usage

```
chk_atomic(x, x_name = NULL)

vld_atomic(x)
```

## Arguments

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

## Value

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

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

## Functions

- `vld_atomic()`: Validate Atomic

## See Also

Other `chk_is`: [chk\\_array\(\)](#), [chk\\_data\(\)](#), [chk\\_function\(\)](#), [chk\\_is\(\)](#), [chk\\_matrix\(\)](#), [chk\\_numeric\(\)](#), [chk\\_s3\\_class\(\)](#), [chk\\_s4\\_class\(\)](#), [chk\\_vector\(\)](#), [chk\\_whole\\_numeric\(\)](#)

## 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	<i>Check Character</i>
---------------	------------------------

---

## 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

Other `chk_typeof`: [chk\\_character\\_or\\_factor\(\)](#), [chk\\_count\(\)](#), [chk\\_double\(\)](#), [chk\\_environment\(\)](#), [chk\\_factor\(\)](#), [chk\\_integer\(\)](#), [chk\\_list\(\)](#), [chk\\_logical\(\)](#)

## Examples

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

---

chk\_character\_or\_factor

*Check Character or Factor*

---

## Description

Checks if character or factor using  
`is.character(x) || is.factor(x)`

## Usage

```
chk_character_or_factor(x, x_name = NULL)

vld_character_or_factor(x)
```

## Arguments

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

## Value

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

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

## Functions

- `vld_character_or_factor()`: Validate Character or Factor

## See Also

Other `chk_typeof`: [chk\\_character\(\)](#), [chk\\_count\(\)](#), [chk\\_double\(\)](#), [chk\\_environment\(\)](#), [chk\\_factor\(\)](#), [chk\\_integer\(\)](#), [chk\\_list\(\)](#), [chk\\_logical\(\)](#)

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

---

**Description**

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

**[Deprecated]**

**Usage**

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

```
vld_chr(x)
```

**Arguments**

<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_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

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

### Details

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

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

### Value

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

### Functions

- `vld_compatible_lengths()`: Validate Compatible Lengths

**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)
```

**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**

Other `chk_typeof`: [chk\\_character\\_or\\_factor\(\)](#), [chk\\_character\(\)](#), [chk\\_double\(\)](#), [chk\\_environment\(\)](#), [chk\\_factor\(\)](#), [chk\\_integer\(\)](#), [chk\\_list\(\)](#), [chk\\_logical\(\)](#)

**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	<i>Check Data</i>
----------	-------------------

---

**Description**

Checks data.frame using  
`inherits(x, "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**

Other `chk_` is: [chk\\_array\(\)](#), [chk\\_atomic\(\)](#), [chk\\_function\(\)](#), [chk\\_is\(\)](#), [chk\\_matrix\(\)](#), [chk\\_numeric\(\)](#), [chk\\_s3\\_class\(\)](#), [chk\\_s4\\_class\(\)](#), [chk\\_vector\(\)](#), [chk\\_whole\\_numeric\(\)](#)

**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**

Other `chk_scalars`: [chk\\_date\\_time\(\)](#), [chk\\_number\(\)](#), [chk\\_scalar\(\)](#), [chk\\_string\(\)](#), [chk\\_tz\(\)](#), [chk\\_whole\\_number\(\)](#)

**Examples**

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

---

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

---

### Description

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

### Usage

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

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

```
vld_date_time(x)
```

```
vld_datetime(x)
```

### Arguments

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

### Value

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

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

### Functions

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

### See Also

Other `chk_scalars`: [chk\\_date\(\)](#), [chk\\_number\(\)](#), [chk\\_scalar\(\)](#), [chk\\_string\(\)](#), [chk\\_tz\(\)](#), [chk\\_whole\\_number\(\)](#)

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

---

**Description**

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

**Usage**

```
chk_dir(x, x_name = NULL)

vld_dir(x)
```

**Arguments**

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**

Other `chk_files`: [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

<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_double()`: Validate Double

## See Also

Other `chk_typeof`: [chk\\_character\\_or\\_factor\(\)](#), [chk\\_character\(\)](#), [chk\\_count\(\)](#), [chk\\_environment\(\)](#), [chk\\_factor\(\)](#), [chk\\_integer\(\)](#), [chk\\_list\(\)](#), [chk\\_logical\(\)](#)

## 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	<i>Check Environment</i>
-----------------	--------------------------

---

## 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

Other `chk_typeof`: [chk\\_character\\_or\\_factor\(\)](#), [chk\\_character\(\)](#), [chk\\_count\(\)](#), [chk\\_double\(\)](#), [chk\\_factor\(\)](#), [chk\\_integer\(\)](#), [chk\\_list\(\)](#), [chk\\_logical\(\)](#)

**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	<i>Check Equal</i>
-----------	--------------------

---

**Description**

Checks if is equal (identical within tolerance) to y using  
`vld_true(all.equal(x, y, tolerance))`

**Usage**

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

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

**Arguments**

x	The object to check.
y	An object to check against.
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_equal()`: Validate Equal

**See Also**

Other `chk_` equals: [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	<i>Check Equivalent</i>
----------------	-------------------------

---

**Description**

Checks if is equivalent (equal ignoring attributes) to y using  
`vld_true(all.equal(x, y, tolerance, check.attributes = FALSE))`

**Usage**

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

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

**Arguments**

x	The object to check.
y	An object to check against.
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 used in pipes.

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

**Functions**

- `vld_equivalent()`: Validate Equivalent

**See Also**

Other `chk_equals`: [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

x	The object to check.
ext	A character vector of the permitted file extensions (without the .).
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_ext()`: Validate File Extension

## See Also

Other `chk_` files: [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	<i>Check Factor</i>
------------	---------------------

---

## 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

Other `chk_typeof`: [chk\\_character\\_or\\_factor\(\)](#), [chk\\_character\(\)](#), [chk\\_count\(\)](#), [chk\\_double\(\)](#), [chk\\_environment\(\)](#), [chk\\_integer\(\)](#), [chk\\_list\(\)](#), [chk\\_logical\(\)](#)

**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**

Other `chk_` logical: [chk\\_flag\(\)](#), [chk\\_lgl\(\)](#), [chk\\_true\(\)](#)

**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**

Other `chk_files`: [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.

**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**

Other `chk_logical`: [chk\\_false\(\)](#), [chk\\_lgl\(\)](#), [chk\\_true\(\)](#)

**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**

Other `chk_is`: [chk\\_array\(\)](#), [chk\\_atomic\(\)](#), [chk\\_data\(\)](#), [chk\\_is\(\)](#), [chk\\_matrix\(\)](#), [chk\\_numeric\(\)](#), [chk\\_s3\\_class\(\)](#), [chk\\_s4\\_class\(\)](#), [chk\\_vector\(\)](#), [chk\\_whole\\_numeric\(\)](#)

### 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

Other `chk_ranges`: [chk\\_gte\(\)](#), [chk\\_lte\(\)](#), [chk\\_lt\(\)](#), [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**

Other `chk_ranges`: [chk\\_gt\(\)](#), [chk\\_lte\(\)](#), [chk\\_lt\(\)](#), [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**

Other `chk_equals`: [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	<i>Check Integer</i>
-------------	----------------------

---

## 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

Other `chk_typeof`: [chk\\_character\\_or\\_factor\(\)](#), [chk\\_character\(\)](#), [chk\\_count\(\)](#), [chk\\_double\(\)](#), [chk\\_environment\(\)](#), [chk\\_factor\(\)](#), [chk\\_list\(\)](#), [chk\\_logical\(\)](#)

**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**

<code>x</code>	The object to check.
<code>class</code>	A string specifying the class.
<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_is()`: Validate Inherits from Class

**See Also**

Other `chk_is`: [chk\\_array\(\)](#), [chk\\_atomic\(\)](#), [chk\\_data\(\)](#), [chk\\_function\(\)](#), [chk\\_matrix\(\)](#), [chk\\_numeric\(\)](#), [chk\\_s3\\_class\(\)](#), [chk\\_s4\\_class\(\)](#), [chk\\_vector\(\)](#), [chk\\_whole\\_numeric\(\)](#)

**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**

Other `chk_set`: [chk\\_not\\_subset\(\)](#), [chk\\_order\\_set\(\)](#), [chk\\_superset\(\)](#), [vld\\_not\\_subset\(\)](#), [vld\\_order\\_set\(\)](#)

**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**

<code>x</code>	The object to check.
<code>length</code>	A count of the length.
<code>upper</code>	A count of the max length.
<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_length()`: Validate Length

**See Also**

[check\\_dim\(\)](#)

Other `chk_misc`: [chk\\_match\(\)](#), [chk\\_missing\(\)](#), [chk\\_named\(\)](#), [chk\\_not\\_any\\_na\(\)](#), [chk\\_not\\_empty\(\)](#), [chk\\_not\\_missing\(\)](#), [chk\\_sorted\(\)](#), [chk\\_unique\(\)](#), [chk\\_valid\\_name\(\)](#)

**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	<i>Check Logical Scalar</i>
---------	-----------------------------

---

**Description**

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

**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**

Other `chk_` logical: [chk\\_false\(\)](#), [chk\\_flag\(\)](#), [chk\\_true\(\)](#)

### 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

Other `chk_typeof`: [chk\\_character\\_or\\_factor\(\)](#), [chk\\_character\(\)](#), [chk\\_count\(\)](#), [chk\\_double\(\)](#), [chk\\_environment\(\)](#), [chk\\_factor\(\)](#), [chk\\_integer\(\)](#), [chk\\_logical\(\)](#)



**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)`

**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**

Other `chk_typeof`: [chk\\_character\\_or\\_factor\(\)](#), [chk\\_character\(\)](#), [chk\\_count\(\)](#), [chk\\_double\(\)](#), [chk\\_environment\(\)](#), [chk\\_factor\(\)](#), [chk\\_integer\(\)](#), [chk\\_list\(\)](#)

## 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

*Check Less Than*

---

## 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

Other `chk_ranges`: [chk\\_gte\(\)](#), [chk\\_gt\(\)](#), [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	<i>Check Less Than or Equal To</i>
---------	------------------------------------

---

**Description**

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

**Usage**

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

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

**Arguments**

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

**Value**

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

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

**Functions**

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

**See Also**

Other `chk_ranges`: [chk\\_gte\(\)](#), [chk\\_gt\(\)](#), [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, regexp = ".+", x_name = NULL)

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

**Arguments**

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

**Value**

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

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

**Functions**

- `vld_match()`: Validate Matches

**See Also**

Other `chk_misc`: [chk\\_length\(\)](#), [chk\\_missing\(\)](#), [chk\\_named\(\)](#), [chk\\_not\\_any\\_na\(\)](#), [chk\\_not\\_empty\(\)](#), [chk\\_not\\_missing\(\)](#), [chk\\_sorted\(\)](#), [chk\\_unique\(\)](#), [chk\\_valid\\_name\(\)](#)

## 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

Other `chk_` is: [chk\\_array\(\)](#), [chk\\_atomic\(\)](#), [chk\\_data\(\)](#), [chk\\_function\(\)](#), [chk\\_is\(\)](#), [chk\\_numeric\(\)](#), [chk\\_s3\\_class\(\)](#), [chk\\_s4\\_class\(\)](#), [chk\\_vector\(\)](#), [chk\\_whole\\_numeric\(\)](#)

**Examples**

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

---

chk\_missing

*Check Missing Argument*

---

**Description**

Checks argument missing using  
missing(x)

**Usage**

```
chk_missing(x, x_name = NULL)

vld_missing(x)
```

**Arguments**

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

**Details**

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

**Value**

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

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

**Functions**

- `vld_missing()`: Validate Missing Argument

**See Also**

Other `chk_misc`: [chk\\_length\(\)](#), [chk\\_match\(\)](#), [chk\\_named\(\)](#), [chk\\_not\\_any\\_na\(\)](#), [chk\\_not\\_empty\(\)](#), [chk\\_not\\_missing\(\)](#), [chk\\_sorted\(\)](#), [chk\\_unique\(\)](#), [chk\\_valid\\_name\(\)](#)

## 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

Other `chk_misc`: [chk\\_length\(\)](#), [chk\\_match\(\)](#), [chk\\_missing\(\)](#), [chk\\_not\\_any\\_na\(\)](#), [chk\\_not\\_empty\(\)](#), [chk\\_not\\_missing\(\)](#), [chk\\_sorted\(\)](#), [chk\\_unique\(\)](#), [chk\\_valid\\_name\(\)](#)

**Examples**

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

---

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

---

**Description**

Checks if not any missing values using

`!anyNA(x)`

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

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

**Usage**

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

```
vld_not_any_na(x)
```

**Arguments**

<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_any_na()`: Validate Not Any Missing Values

**See Also**

Other `chk_misc`: [chk\\_length\(\)](#), [chk\\_match\(\)](#), [chk\\_missing\(\)](#), [chk\\_named\(\)](#), [chk\\_not\\_empty\(\)](#), [chk\\_not\\_missing\(\)](#), [chk\\_sorted\(\)](#), [chk\\_unique\(\)](#), [chk\\_valid\\_name\(\)](#)



**Examples**

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

---

chk\_not\_empty

*Check Not Empty*


---

**Description**

Checks if not empty using

`length(x) != 0L`

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

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

**Usage**

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

```
vld_not_empty(x)
```

**Arguments**

<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_not_empty()`: Validate Not Empty

**See Also**

Other `chk_misc`: [chk\\_length\(\)](#), [chk\\_match\(\)](#), [chk\\_missing\(\)](#), [chk\\_named\(\)](#), [chk\\_not\\_any\\_na\(\)](#), [chk\\_not\\_missing\(\)](#), [chk\\_sorted\(\)](#), [chk\\_unique\(\)](#), [chk\\_valid\\_name\(\)](#)

## Examples

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

---

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

---

## Description

Checks argument not missing using  
`!missing(x)`

## Usage

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

## Arguments

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

## Details

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

## Value

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

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

## Functions

- `vld_not_missing()`: Validate Not Missing Argument

**See Also**

Other `chk_misc`: [chk\\_length\(\)](#), [chk\\_match\(\)](#), [chk\\_missing\(\)](#), [chk\\_named\(\)](#), [chk\\_not\\_any\\_na\(\)](#), [chk\\_not\\_empty\(\)](#), [chk\\_sorted\(\)](#), [chk\\_unique\(\)](#), [chk\\_valid\\_name\(\)](#)

**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**

Other chk\_nulls: [chk\\_null\(\)](#)

**Examples**

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

---

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

---

**Description**

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

**Usage**

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

**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.

**See Also**

Other `chk_set`: [chk\\_join\(\)](#), [chk\\_orderset\(\)](#), [chk\\_superset\(\)](#), [vld\\_not\\_subset\(\)](#), [vld\\_orderset\(\)](#)

**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**

Other `chk_nulls`: [chk\\_not\\_null\(\)](#)

**Examples**

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

---

chk_null_or	<i>Check NULL Or</i>
-------------	----------------------

---

**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**

Other `chk_scalars`: [chk\\_date\\_time\(\)](#), [chk\\_date\(\)](#), [chk\\_scalar\(\)](#), [chk\\_string\(\)](#), [chk\\_tz\(\)](#), [chk\\_whole\\_number\(\)](#)

**Examples**

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

---

 chk\_numeric

*Check Numeric*


---

**Description**

Checks if numeric using

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

**Pass:** 1, 1:2, NA\_real\_, integer(0), matrix(1:3).

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

**Usage**

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

```
vld_numeric(x)
```

**Arguments**

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

**Value**

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

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

**Functions**

- `vld_numeric()`: Validate Numeric

**See Also**

Other `chk_is`: [chk\\_array\(\)](#), [chk\\_atomic\(\)](#), [chk\\_data\(\)](#), [chk\\_function\(\)](#), [chk\\_is\(\)](#), [chk\\_matrix\(\)](#), [chk\\_s3\\_class\(\)](#), [chk\\_s4\\_class\(\)](#), [chk\\_vector\(\)](#), [chk\\_whole\\_numeric\(\)](#)

**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\_orderiset

*Check Set Ordered*

---

**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_orderiset(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**

Other `chk_set`: [chk\\_join\(\)](#), [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**

Other `chk_ranges`: `chk_gte()`, `chk_gt()`, `chk_lte()`, `chk_lt()`

**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_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**

<code>x</code>	The object to check.
<code>class</code>	A string specifying the class.
<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_s3_class()`: Validate Inherits from S3 Class

**See Also**

Other `chk_is`: [chk\\_array\(\)](#), [chk\\_atomic\(\)](#), [chk\\_data\(\)](#), [chk\\_function\(\)](#), [chk\\_is\(\)](#), [chk\\_matrix\(\)](#), [chk\\_numeric\(\)](#), [chk\\_s4\\_class\(\)](#), [chk\\_vector\(\)](#), [chk\\_whole\\_numeric\(\)](#)

**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**

<code>x</code>	The object to check.
<code>class</code>	A string specifying the class.
<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_s4_class()`: Validate Inherits from S4 Class

**See Also**

Other `chk_is`: [chk\\_array\(\)](#), [chk\\_atomic\(\)](#), [chk\\_data\(\)](#), [chk\\_function\(\)](#), [chk\\_is\(\)](#), [chk\\_matrix\(\)](#), [chk\\_numeric\(\)](#), [chk\\_s3\\_class\(\)](#), [chk\\_vector\(\)](#), [chk\\_whole\\_numeric\(\)](#)

## 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

Other `chk_` scalars: [chk\\_date\\_time\(\)](#), [chk\\_date\(\)](#), [chk\\_number\(\)](#), [chk\\_string\(\)](#), [chk\\_tz\(\)](#), [chk\\_whole\\_number\(\)](#)

## Examples

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

---

chk\_sorted

*Check Sorted*

---

## Description

Checks if is sorted using  
`is.unsorted(x, na.rm = TRUE)`

## Usage

```
chk_sorted(x, x_name = NULL)

vld_sorted(x)
```

## Arguments

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

## Value

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

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

## Functions

- `vld_sorted()`: Validate Sorted

## See Also

Other `chk_misc`: [chk\\_length\(\)](#), [chk\\_match\(\)](#), [chk\\_missing\(\)](#), [chk\\_named\(\)](#), [chk\\_not\\_any\\_na\(\)](#), [chk\\_not\\_empty\(\)](#), [chk\\_not\\_missing\(\)](#), [chk\\_unique\(\)](#), [chk\\_valid\\_name\(\)](#)

## Examples

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

---

chk\_string

*Check String*

---

## Description

Checks if string

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

## Usage

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

```
vld_string(x)
```

## Arguments

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

## Value

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

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

## Functions

- `vld_string()`: Validate String

## See Also

Other `chk_scalars`: [chk\\_date\\_time\(\)](#), [chk\\_date\(\)](#), [chk\\_number\(\)](#), [chk\\_scalar\(\)](#), [chk\\_tz\(\)](#), [chk\\_whole\\_number\(\)](#)

**Examples**

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

---

chk_superset	<i>Check Superset</i>
--------------	-----------------------

---

**Description**

Checks if includes all values using  
`all(values %in% x)`

**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**

Other `chk_set`: [chk\\_join\(\)](#), [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

Other `chk_` logical: [chk\\_false\(\)](#), [chk\\_flag\(\)](#), [chk\\_lgl\(\)](#)



**Examples**

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

---

chk\_tz

*Check Time Zone*


---

**Description**

Checks if non-missing valid scalar timezone using

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

**Usage**

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

```
vld_tz(x)
```

**Arguments**

x                   The object to check.

x\_name              A string of the name of object x or NULL.

**Value**

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

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

**Functions**

- `vld_tz()`: Validate Time Zone

**See Also**

Other `chk_scalars`: [chk\\_date\\_time\(\)](#), [chk\\_date\(\)](#), [chk\\_number\(\)](#), [chk\\_scalar\(\)](#), [chk\\_string\(\)](#), [chk\\_whole\\_number\(\)](#)

## Examples

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

---

chk\_unique

*Check Unique*

---

## Description

Checks if unique using

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

## Usage

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

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

## Arguments

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

## Value

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

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

## Functions

- `vld_unique()`: Validate Unique

## See Also

Other `chk_misc`: [chk\\_length\(\)](#), [chk\\_match\(\)](#), [chk\\_missing\(\)](#), [chk\\_named\(\)](#), [chk\\_not\\_any\\_na\(\)](#), [chk\\_not\\_empty\(\)](#), [chk\\_not\\_missing\(\)](#), [chk\\_sorted\(\)](#), [chk\\_valid\\_name\(\)](#)

**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**

Other `chk_ellipsis`: [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**

Other `chk_ellipsis`: [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**

Other `chk_misc`: [chk\\_length\(\)](#), [chk\\_match\(\)](#), [chk\\_missing\(\)](#), [chk\\_named\(\)](#), [chk\\_not\\_any\\_na\(\)](#), [chk\\_not\\_empty\(\)](#), [chk\\_not\\_missing\(\)](#), [chk\\_sorted\(\)](#), [chk\\_unique\(\)](#)

**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**

Other `chk_is`: [chk\\_array\(\)](#), [chk\\_atomic\(\)](#), [chk\\_data\(\)](#), [chk\\_function\(\)](#), [chk\\_is\(\)](#), [chk\\_matrix\(\)](#), [chk\\_numeric\(\)](#), [chk\\_s3\\_class\(\)](#), [chk\\_s4\\_class\(\)](#), [chk\\_whole\\_numeric\(\)](#)

## 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

Other `chk_scalars`: [chk\\_date\\_time\(\)](#), [chk\\_date\(\)](#), [chk\\_number\(\)](#), [chk\\_scalar\(\)](#), [chk\\_string\(\)](#), [chk\\_tz\(\)](#)

## 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

Other `chk_` is: [chk\\_array\(\)](#), [chk\\_atomic\(\)](#), [chk\\_data\(\)](#), [chk\\_function\(\)](#), [chk\\_is\(\)](#), [chk\\_matrix\(\)](#), [chk\\_numeric\(\)](#), [chk\\_s3\\_class\(\)](#), [chk\\_s4\\_class\(\)](#), [chk\\_vector\(\)](#)



**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.
<code>call</code>	The execution environment of a currently running function, e.g. <code>call = caller_env()</code> . The corresponding function call is retrieved and mentioned in error messages as the source of the error. You only need to supply <code>call</code> when throwing a condition from a helper function which wouldn't be relevant to mention in the message. Can also be <code>NULL</code> or a <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**

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

*Expect Chk Error*

---

**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**

- |                     |  |
|---------------------|--|
| <code>object</code> | Object to test.<br>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. |
| <code>regexp</code> | 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> </ul>      |

- If NULL, the default, asserts that there should be an error, but doesn't test for a specific value.
- If NA, asserts that there should be no errors, but we now recommend using `expect_no_error()` and friends instead.

Note that you should only use message with errors/warnings/messages that you generate. Avoid tests that rely on the specific text generated by another package since this can easily change. If you do need to test text generated by another package, either protect the test with `skip_on_cran()` or use `expect_snapshot()`.

...	Arguments passed on to <code>expect_match</code>
	<code>fixed</code> If TRUE, treats regexp as a string to be matched exactly (not a regular expressions). Overrides <code>perl</code> .
	<code>perl</code> logical. Should Perl-compatible regexps be used?
<code>info</code>	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> .
<code>label</code>	Used to customise failure messages. For expert use only.
<code>class</code>	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**

<code>...</code>	Multiple objects that are converted to a string using <code>paste0(..., collapse = '')</code> .
<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.

**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:

<code>n</code>	The value of <code>n</code> .
<code>s</code>	" if <code>n == 1</code> otherwise 's'
<code>r</code>	'is' if <code>n == 1</code> otherwise 'are'
<code>y</code>	'y' if <code>n == 1</code> 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 `NA_character_`.

`collapse` an optional character string to separate the results. Not `NA_character_`. When `collapse` is a string, the result is always a string (`character` 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 *Check Subset*

---

**Description**

Checks if all values in values using  
`all(x %in% values)`

**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**

Other `chk_set`: [chk\\_join\(\)](#), [chk\\_not\\_subset\(\)](#), [chk\\_order\\_set\(\)](#), [chk\\_superset\(\)](#), [vld\\_order\\_set\(\)](#)

**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**

Other `chk_set`: [chk\\_join\(\)](#), [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

- \* **check**
  - check\_data, 6
  - check\_dim, 7
  - check\_dirs, 7
  - check\_files, 8
  - check\_key, 9
  - check\_names, 9
  - check\_values, 10
- \* **chk\_all**
  - chk\_all, 13
  - chk\_all\_equal, 14
  - chk\_all\_equivalent, 15
  - chk\_all\_identical, 16
- \* **chk\_character**
  - chk\_chr, 21
- \* **chk\_ellipsis**
  - chk\_unused, 75
  - chk\_used, 76
- \* **chk\_equals**
  - chk\_equal, 32
  - chk\_equivalent, 33
  - chk\_identical, 42
- \* **chk\_files**
  - chk\_dir, 29
  - chk\_ext, 34
  - chk\_file, 37
- \* **chk\_is**
  - chk\_array, 17
  - chk\_atomic, 18
  - chk\_data, 25
  - chk\_function, 39
  - chk\_is, 44
  - chk\_matrix, 53
  - chk\_numeric, 63
  - chk\_s3\_class, 66
  - chk\_s4\_class, 67
  - chk\_vector, 78
  - chk\_whole\_numeric, 80
- \* **chk\_logical**
  - chk\_false, 36
  - chk\_flag, 38
  - chk\_lgl, 47
  - chk\_true, 72
- \* **chk\_misc**
  - chk\_length, 46
  - chk\_match, 52
  - chk\_missing, 54
  - chk\_named, 55
  - chk\_not\_any\_na, 56
  - chk\_not\_empty, 57
  - chk\_not\_missing, 58
  - chk\_sorted, 69
  - chk\_unique, 74
  - chk\_valid\_name, 77
- \* **chk\_nulls**
  - chk\_not\_null, 59
  - chk\_null, 61
- \* **chk\_ranges**
  - chk\_gt, 40
  - chk\_gte, 41
  - chk\_lt, 50
  - chk\_lte, 51
  - chk\_range, 65
- \* **chk\_scalars**
  - chk\_date, 26
  - chk\_date\_time, 27
  - chk\_number, 62
  - chk\_scalar, 68
  - chk\_string, 70
  - chk\_tz, 73
  - chk\_whole\_number, 79
- \* **chk\_set**
  - chk\_join, 45
  - chk\_not\_subset, 60
  - chk\_orderset, 64
  - chk\_superset, 71
  - vld\_not\_subset, 87
  - vld\_orderset, 89

- \* **chk\_typeof**
  - chk\_character, 19
  - chk\_character\_or\_factor, 20
  - chk\_count, 23
  - chk\_double, 30
  - chk\_environment, 31
  - chk\_factor, 35
  - chk\_integer, 43
  - chk\_list, 48
  - chk\_logical, 49
- \* **deprecated**
  - chk\_chr, 21
  - chk\_dbl, 28
  - chk\_wnum, 81
- abort\_chk, 4
- backtick\_chk (deparse\_backtick\_chk), 82
- base::paste(), 87
- base::paste0(), 87
- cc, 5
- character, 87
- check\_data, 6, 7–11
- check\_dim, 6, 7, 8–11
- check\_dim(), 46
- check\_dirs, 6, 7, 7, 8–11
- check\_files, 6–8, 8, 9–11
- check\_key, 6–8, 9, 10, 11
- check\_names, 6–9, 9, 11
- check\_values, 6–10, 10
- chk\_all, 13, 14–16
- chk\_all\_equal, 13, 14, 15, 16
- chk\_all\_equivalent, 13, 14, 15, 16
- chk\_all\_identical, 13–15, 16
- chk\_array, 17, 18, 25, 39, 44, 53, 64, 67, 78, 80
- chk\_atomic, 17, 18, 25, 39, 44, 53, 64, 67, 78, 80
- chk\_character, 19, 20, 24, 30, 31, 35, 43, 48, 49
- chk\_character\_or\_factor, 19, 20, 24, 30, 31, 35, 43, 48, 49
- chk\_chr, 21, 28, 81
- chk\_compatible\_lengths, 22
- chk\_count, 19, 20, 23, 30, 31, 35, 43, 48, 49
- chk\_data, 17, 18, 25, 39, 44, 53, 64, 67, 78, 80
- chk\_date, 26, 27, 63, 68, 70, 73, 79
- chk\_date\_time, 26, 27, 63, 68, 70, 73, 79
- chk\_datetime (chk\_date\_time), 27
- chk\_dbl, 21, 28, 81
- chk\_deprecated, 21, 28, 81
- chk\_dir, 29, 34, 37
- chk\_double, 19, 20, 24, 30, 31, 35, 43, 48, 49
- chk\_environment, 19, 20, 24, 30, 31, 35, 43, 48, 49
- chk\_equal, 32, 33, 42
- chk\_equivalent, 32, 33, 42
- chk\_ext, 29, 34, 37
- chk\_factor, 19, 20, 24, 30, 31, 35, 43, 48, 49
- chk\_false, 36, 38, 47, 72
- chk\_file, 29, 34, 37
- chk\_flag, 36, 38, 47, 72
- chk\_function, 17, 18, 25, 39, 44, 53, 64, 67, 78, 80
- chk\_gt, 40, 41, 50, 51, 66
- chk\_gte, 40, 41, 50, 51, 66
- chk\_identical, 32, 33, 42
- chk\_integer, 19, 20, 24, 30, 31, 35, 43, 48, 49
- chk\_is, 17, 18, 25, 39, 44, 53, 64, 67, 78, 80
- chk\_join, 45, 60, 65, 71, 88, 89
- chk\_length, 46, 52, 54–57, 59, 69, 74, 77
- chk\_lgl, 36, 38, 47, 72
- chk\_list, 19, 20, 24, 30, 31, 35, 43, 48, 49
- chk\_logical, 19, 20, 24, 30, 31, 35, 43, 48, 49
- chk\_lt, 40, 41, 50, 51, 66
- chk\_lte, 40, 41, 50, 51, 66
- chk\_match, 46, 52, 54–57, 59, 69, 74, 77
- chk\_matrix, 17, 18, 25, 39, 44, 53, 64, 67, 78, 80
- chk\_missing, 46, 52, 54, 55–57, 59, 69, 74, 77
- chk\_named, 46, 52, 54, 55, 56, 57, 59, 69, 74, 77
- chk\_not\_any\_na, 46, 52, 54, 55, 56, 57, 59, 69, 74, 77
- chk\_not\_empty, 46, 52, 54–56, 57, 59, 69, 74, 77
- chk\_not\_missing, 46, 52, 54–57, 58, 69, 74, 77
- chk\_not\_null, 59, 61
- chk\_not\_subset, 45, 60, 65, 71, 88, 89
- chk\_null, 60, 61
- chk\_null\_or, 62
- chk\_null\_or(), 12
- chk\_number, 26, 27, 62, 68, 70, 73, 79
- chk\_numeric, 17, 18, 25, 39, 44, 53, 63, 67, 78, 80

- chk\_order, [45](#), [60](#), [64](#), [71](#), [88](#), [89](#)
- chk\_range, [40](#), [41](#), [50](#), [51](#), [65](#)
- chk\_s3\_class, [17](#), [18](#), [25](#), [39](#), [44](#), [53](#), [64](#), [66](#), [67](#), [78](#), [80](#)
- chk\_s4\_class, [17](#), [18](#), [25](#), [39](#), [44](#), [53](#), [64](#), [67](#), [67](#), [78](#), [80](#)
- chk\_scalar, [26](#), [27](#), [63](#), [68](#), [70](#), [73](#), [79](#)
- chk\_setequal (vld\_order), [89](#)
- chk\_sorted, [46](#), [52](#), [54–57](#), [59](#), [69](#), [74](#), [77](#)
- chk\_string, [26](#), [27](#), [63](#), [68](#), [70](#), [73](#), [79](#)
- chk\_subset (vld\_not\_subset), [87](#)
- chk\_superset, [45](#), [60](#), [65](#), [71](#), [88](#), [89](#)
- chk\_true, [36](#), [38](#), [47](#), [72](#)
- chk\_tz, [26](#), [27](#), [63](#), [68](#), [70](#), [73](#), [79](#)
- chk\_unique, [46](#), [52](#), [54–57](#), [59](#), [69](#), [74](#), [77](#)
- chk\_unused, [75](#), [76](#)
- chk\_used, [75](#), [76](#)
- chk\_valid\_name, [46](#), [52](#), [54–57](#), [59](#), [69](#), [74](#), [77](#)
- chk\_vector, [17](#), [18](#), [25](#), [39](#), [44](#), [53](#), [64](#), [67](#), [78](#), [80](#)
- chk\_whole\_number, [26](#), [27](#), [63](#), [68](#), [70](#), [73](#), [79](#)
- chk\_whole\_numeric, [17](#), [18](#), [25](#), [39](#), [44](#), [53](#), [64](#), [67](#), [78](#), [80](#)
- chk\_wnum, [21](#), [28](#), [81](#)
- chkor, [11](#)
- chkor\_vld, [12](#)
- defused function call, [4](#), [83](#)
- deparse(), [82](#)
- deparse\_backtick\_chk, [82](#)
- err, [83](#)
- err(), [4](#)
- expect\_chk\_error, [84](#)
- expect\_chk\_error(), [84](#)
- expect\_length, [85](#)
- expect\_match, [85](#)
- expect\_named, [85](#)
- expect\_no\_error(), [85](#)
- expect\_null, [85](#)
- expect\_output, [85](#)
- expect\_reference, [85](#)
- expect\_silent, [85](#)
- expect\_snapshot(), [85](#)
- Including function calls in error messages, [4](#), [83](#)
- message\_chk, [86](#)
- message\_chk(), [83](#)
- msg (err), [83](#)
- NA\_character\_, [87](#)
- p, [87](#)
- p0 (p), [87](#)
- quasi\_label, [84](#), [85](#)
- rlang::abort(), [83](#)
- rlang::inform(), [83](#)
- rlang::warn(), [83](#)
- tolower(), [34](#)
- toupper(), [34](#)
- unbacktick\_chk (deparse\_backtick\_chk), [82](#)
- vld\_all (chk\_all), [13](#)
- vld\_all\_equal (chk\_all\_equal), [14](#)
- vld\_all\_equivalent (chk\_all\_equivalent), [15](#)
- vld\_all\_identical (chk\_all\_identical), [16](#)
- vld\_array (chk\_array), [17](#)
- vld\_atomic (chk\_atomic), [18](#)
- vld\_character (chk\_character), [19](#)
- vld\_character\_or\_factor (chk\_character\_or\_factor), [20](#)
- vld\_chr (chk\_chr), [21](#)
- vld\_compatible\_lengths (chk\_compatible\_lengths), [22](#)
- vld\_count (chk\_count), [23](#)
- vld\_data (chk\_data), [25](#)
- vld\_date (chk\_date), [26](#)
- vld\_date\_time (chk\_date\_time), [27](#)
- vld\_datetime (chk\_date\_time), [27](#)
- vld\_dbl (chk\_dbl), [28](#)
- vld\_dir (chk\_dir), [29](#)
- vld\_double (chk\_double), [30](#)
- vld\_environment (chk\_environment), [31](#)
- vld\_equal (chk\_equal), [32](#)
- vld\_equivalent (chk\_equivalent), [33](#)
- vld\_ext (chk\_ext), [34](#)
- vld\_factor (chk\_factor), [35](#)
- vld\_false (chk\_false), [36](#)
- vld\_file (chk\_file), [37](#)
- vld\_flag (chk\_flag), [38](#)

vld\_function (chk\_function), 39  
vld\_gt (chk\_gt), 40  
vld\_gte (chk\_gte), 41  
vld\_identical (chk\_identical), 42  
vld\_integer (chk\_integer), 43  
vld\_is (chk\_is), 44  
vld\_join (chk\_join), 45  
vld\_length (chk\_length), 46  
vld\_lgl (chk\_lgl), 47  
vld\_list (chk\_list), 48  
vld\_logical (chk\_logical), 49  
vld\_lt (chk\_lt), 50  
vld\_lte (chk\_lte), 51  
vld\_match (chk\_match), 52  
vld\_matrix (chk\_matrix), 53  
vld\_missing (chk\_missing), 54  
vld\_named (chk\_named), 55  
vld\_not\_any\_na (chk\_not\_any\_na), 56  
vld\_not\_empty (chk\_not\_empty), 57  
vld\_not\_missing (chk\_not\_missing), 58  
vld\_not\_null (chk\_not\_null), 59  
vld\_not\_subset, 45, 60, 65, 71, 87, 89  
vld\_null (chk\_null), 61  
vld\_number (chk\_number), 62  
vld\_numeric (chk\_numeric), 63  
vld\_orderset, 45, 60, 65, 71, 88, 89  
vld\_range (chk\_range), 65  
vld\_s3\_class (chk\_s3\_class), 66  
vld\_s4\_class (chk\_s4\_class), 67  
vld\_scalar (chk\_scalar), 68  
vld\_setequal (vld\_orderset), 89  
vld\_sorted (chk\_sorted), 69  
vld\_string (chk\_string), 70  
vld\_subset (vld\_not\_subset), 87  
vld\_superset (chk\_superset), 71  
vld\_true (chk\_true), 72  
vld\_tz (chk\_tz), 73  
vld\_unique (chk\_unique), 74  
vld\_unused (chk\_unused), 75  
vld\_used (chk\_used), 76  
vld\_valid\_name (chk\_valid\_name), 77  
vld\_vector (chk\_vector), 78  
vld\_whole\_number (chk\_whole\_number), 79  
vld\_whole\_numeric (chk\_whole\_numeric),  
80  
vld\_wnum (chk\_wnum), 81  
  
wrn (err), 83