

# Package: checkr (via r-universe)

November 1, 2024

**Title** Check the Properties of Common R Objects

**Version** 0.5.0.9001

**Description** Expressive, assertive, pipe-friendly functions to check the properties of common R objects. In the case of failure the functions issue informative error messages. Superseded by the 'chk' package.

**License** MIT + file LICENSE

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

**BugReports** <https://github.com/poissonconsulting/checkr/issues>

**Depends** R (>= 3.5)

**Imports** err, lifecycle

**Suggests** assertthat, chk, checkmate, covr, datasets, magrittr, hms, dplyr, units, rlang, testthat, knitr, rmarkdown

**Remotes** poissonconsulting/err

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.2

**VignetteBuilder** knitr

**Language** en-CA

**Roxygen** list(markdown = TRUE)

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

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

**RemoteRef** HEAD

**RemoteSha** 6c1e922ae9bec9f6b03b07d1935759edd3adef05

## Contents

checkor . . . . .	3
check_attributes . . . . .	4
check_character . . . . .	5
check_chr . . . . .	5
check_classes . . . . .	6
check_colnames . . . . .	7
check_count . . . . .	8
check_data . . . . .	9
check_date . . . . .	10
check_dbl . . . . .	11
check_dttm . . . . .	12
check_environment . . . . .	13
check_flag_na . . . . .	14
check_function . . . . .	14
check_grepl . . . . .	15
check_homogenous . . . . .	16
check_inherits . . . . .	17
check_int . . . . .	17
check_integer . . . . .	18
check_intersection . . . . .	19
check_join . . . . .	20
check_key . . . . .	21
check_length . . . . .	22
check_length1 . . . . .	23
check_levels . . . . .	23
check_lgl . . . . .	24
check_list . . . . .	25
check_logical . . . . .	26
check_missing_colnames . . . . .	27
check_missing_names . . . . .	28
check_name . . . . .	28
check_named . . . . .	29
check_names . . . . .	30
check_nchar . . . . .	31
check_ncol . . . . .	32
check_neg_dbl . . . . .	33
check_neg_int . . . . .	33
check_nlevels . . . . .	34
check_noneg_dbl . . . . .	35
check_noneg_int . . . . .	36
check_no_attributes . . . . .	36
check_nrow . . . . .	37
check_null . . . . .	38
check_numeric . . . . .	39
check_pattern . . . . .	40
check_pos_dbl . . . . .	40

check_pos_int . . . . .	41
check_prob . . . . .	42
check_props . . . . .	43
check_rbind . . . . .	43
check_scalar . . . . .	44
check_sorted . . . . .	45
check_tzone . . . . .	46
check_unique . . . . .	47
check_unnamed . . . . .	47
check_unused . . . . .	48
check_vector . . . . .	49
chk_deparse . . . . .	50
chk_fail . . . . .	51
chk_max_dbl . . . . .	51
chk_max_int . . . . .	52
chk_min_dbl . . . . .	52
chk_min_int . . . . .	53
chk_tiny_dbl . . . . .	53

**Index** **54**

checkor *Check OR*

**Description**

Checks that at least one check passes.

**Usage**

checkor(..., error = TRUE)

**Arguments**

- ...           The checks to check.
- error        A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible flag indicating whether at least one check passes (if it doesn't throw an error).

**Examples**

```
checkor(check_null(NULL), check_null(1), error = FALSE)
checkor(check_null(1), check_null(1), error = FALSE)
checkor(check_null(1), check_null(2), error = FALSE)
```

---

check\_attributes      *Check Attributes*

---

### Description

Checks an objects attributes.

### Usage

```
check_attributes(  
  x,  
  values = NULL,  
  exclusive = FALSE,  
  order = FALSE,  
  names = TRUE,  
  class = TRUE,  
  x_name = substitute(x),  
  error = TRUE  
)
```

### Arguments

x	The object to check.
values	NULL (default) or a character vector specifying the column names or a named list specifying the column names and values.
exclusive	A flag indicating whether other elements are not permitted.
order	A flag indicating whether the elements have to occur in the same order as values.
names	A flag specifying whether names should be considered an attribute.
class	A flag specifying whether class should be considered an attribute.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

### Value

An invisible copy of x (if it doesn't throw an error).

### See Also

[check\\_list\(\)](#)

### Examples

```
x <- 1  
attributes(x) <- list(y = 2L)  
check_attributes(x, values = list(y = 3:4), error = FALSE)
```

---

check_character	<i>Check Character</i>
-----------------	------------------------

---

**Description**

Checks if `x` is a character vector with no attributes including names.

**Usage**

```
check_character(x, coerce = FALSE, x_name = substitute(x), error = TRUE)
```

**Arguments**

<code>x</code>	The object to check.
<code>coerce</code>	A flag indicating whether to coerce a factor to a character vector and drop attributes including names.
<code>x_name</code>	A string of the name of the object <code>x</code> .
<code>error</code>	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of `x` (if it doesn't throw an error).

**See Also**

[check\\_int\(\)](#)

**Examples**

```
check_character("1", error = FALSE)
check_character(1:2, error = FALSE)
```

---

check_chr	<i>Check String</i>
-----------	---------------------

---

**Description**

Checks if object is a string (non-missing character scalar with no attributes including names).

**Usage**

```
check_chr(x, coerce = FALSE, x_name = substitute(x), error = TRUE)
```

```
check_string(x, coerce = FALSE, x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The object to check.
coerce	A flag indicating whether to coerce a factor scalar to a string and drop attributes including names.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_scalar\(\)](#)

**Examples**

```
check_chr(1, error = FALSE)
check_chr("1", error = FALSE)
check_chr(c("1", "2"), error = FALSE)
```

---

check\_classes

*Check Classes*

---

**Description**

Checks that an object inherits from one or more classes.

**Usage**

```
check_classes(
  x,
  classes = character(0),
  exclusive = FALSE,
  order = FALSE,
  x_name = substitute(x),
  error = TRUE
)
```

**Arguments**

x	The object to check.
classes	A character vector of the classes x should inherit from.
exclusive	A flag indicating whether other classes are not permitted.

order	A flag indicating whether the object classes have to occur in the same order as classes.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

### Details

The classes of an object can be returned using the `class()` function.

### Value

An invisible copy of x (if it doesn't throw an error).

### See Also

[check\\_inherits\(\)](#)

### Examples

```
check_classes(list())
check_classes(list(), "list")
check_classes(list(), "numeric", error = FALSE)
```

---

check_colnames	<i>Check Colnames</i>
----------------	-----------------------

---

### Description

Checks the column names of a data frame as returned by the `colnames()` function. The function can check the order of the columns and whether other columns are permitted.

### Usage

```
check_colnames(  
  x,  
  colnames = character(0),  
  exclusive = FALSE,  
  order = FALSE,  
  x_name = substitute(x),  
  error = TRUE  
)
```

**Arguments**

x	The data to check.
colnames	A character vector of the column names.
exclusive	A flag indicating whether other columns are not permitted.
order	A flag indicating whether the columns have to occur in the same order as colnames.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_missing\\_colnames\(\)](#) and [check\\_data\(\)](#)

**Examples**

```
data <- data.frame(x = 1, y = 2, z = 0)
check_colnames(data, c("y", "x"), error = FALSE)
check_colnames(data, c("y", "x"), exclusive = TRUE, error = FALSE)
check_colnames(data, c("y", "x"), order = TRUE, error = FALSE)
check_colnames(data, c("a"), error = FALSE)
```

---

check\_count

*Check Count*

---

**Description**

Checks if an object is a count (non-negative integer or if coerce = TRUE non-negative numeric whole number).

**Usage**

```
check_count(x, coerce = FALSE, x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The object to check.
coerce	A flag indicating whether to coerce a non-negative numeric (dbl) whole number to a count and drop attributes (including names).
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.



**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_scalar\(\)](#)

**Examples**

```
check_count(-1L, error = FALSE)
check_count(1L, error = FALSE)
check_count(1, error = FALSE)
check_count(1, coerce = TRUE, error = FALSE)
check_count(1.01, coerce = TRUE, error = FALSE)
```

---

check\_data

*Check Data*

---

**Description**

Checks whether an object is a data frame. Can also check the number of rows, the names and order and values of the columns as well as whether particular columns form a unique key.

**Usage**

```
check_data(
  x,
  values = NULL,
  nrow = NA,
  exclusive = FALSE,
  order = FALSE,
  key = character(0),
  x_name = substitute(x),
  error = TRUE
)
```

**Arguments**

x	The object to check.
values	NULL (default) or a character vector specifying the column names or a named list specifying the column names and values.
nrow	A flag indicating whether x should have rows (versus no rows) or a missing value indicating no requirements or a count or count range of the number of rows. @return An invisible copy of x (if it doesn't throw an error).
exclusive	A flag indicating whether other columns are not permitted.
order	A flag indicating whether the columns have to occur in the same order as values.

key	A character vector of the columns that represent a unique key.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_colnames\(\)](#), [check\\_nrow\(\)](#) and [check\\_key\(\)](#)

**Examples**

```
z <- data.frame(
  Count = c(0L, 3L, 3L, 0L, NA),
  Longitude = c(0, 0, 90, 90, 180),
  Latitude = c(0, 90, 90.2, 100, -180),
  Type = factor(c("Good", "Bad", "Bad", "Bad", "Bad"), levels = c("Good", "Bad")),
  Extra = TRUE,
  Comments = c("In Greenwich", "Somewhere else", "I'm lost",
    "I didn't see any", "Help"),
  stringsAsFactors = FALSE)

check_data(z, values = list(
  Count = 1,
  Extra = NA,
  Latitude = c(45, 90)
), exclusive = TRUE, order = TRUE, nrow = 10L, key = "Longitude", error = FALSE)
```

---

check\_date

*Check Date*

---

**Description**

Checks if x is a date (non-missing unnamed Date scalar).

**Usage**

```
check_date(x, coerce = FALSE, x_name = substitute(x), error = TRUE)
```

```
check_day(x, coerce = FALSE, x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The object to check.
coerce	A flag indicating whether to coerce a date time (POSIXt scalar) to a Date and remove names.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_datetime\(\)](#)

**Examples**

```
check_date(Sys.Date(), error = FALSE)
check_date(Sys.time(), error = FALSE)
check_date(Sys.time(), coerce = TRUE, error = FALSE)
```

---

check\_dbl

*Check Dbl*

---

**Description**

Checks if x is a dbl (non-missing numeric scalar with no attributes including names).

**Usage**

```
check_dbl(x, coerce = FALSE, x_name = substitute(x), error = TRUE)

check_number(x, coerce = FALSE, x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The object to check.
coerce	A flag indicating whether to coerce a scalar integer to a dbl and drop attributes including names.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**[check\\_scalar\(\)](#)**Examples**

```

check_dbl(1, error = FALSE)
check_dbl(1L, error = FALSE)
check_dbl(c(1,2), error = FALSE)

```

---

`check_dttm`*Check Date Time*

---

**Description**

Checks if x is a datetime (non-missing unnamed POSIXct scalar).

**Usage**

```

check_dttm(
  x,
  coerce = FALSE,
  tzone = "UTC",
  x_name = substitute(x),
  error = TRUE
)

```

```

check_datetime(
  x,
  coerce = FALSE,
  tzone = "",
  x_name = substitute(x),
  error = TRUE
)

```

**Arguments**

<code>x</code>	The object to check.
<code>coerce</code>	A flag indicating whether to coerce a date to a dttm (using the time zone <code>tzone</code> ) and remove names.
<code>tzone</code>	A string of the time zone where "" is the current time zone.
<code>x_name</code>	A string of the name of the object <code>x</code> .
<code>error</code>	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of `x` (if it doesn't throw an error).

**See Also**[check\\_scalar\(\)](#)**Examples**

```
check_dttm(Sys.Date(), error = FALSE)
check_dttm(Sys.time(), error = FALSE)
```

---

check_environment	<i>Check Environment</i>
-------------------	--------------------------

---

**Description**

Checks if `x` is an environment.

**Usage**

```
check_environment(x, x_name = substitute(x), error = TRUE)
```

**Arguments**

<code>x</code>	The object to check.
<code>x_name</code>	A string of the name of the object <code>x</code> .
<code>error</code>	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of `x` (if it doesn't throw an error).

**Examples**

```
check_environment(1, error = FALSE)
check_environment(.GlobalEnv, error = FALSE)
```

---

check_flag_na	<i>Check Flag or NA</i>
---------------	-------------------------

---

**Description**

Checks if `x` is a flag or NA (missing logical scalar).

**Usage**

```
check_flag_na(x, coerce = TRUE, x_name = substitute(x), error = TRUE)
```

**Arguments**

<code>x</code>	The object to check.
<code>coerce</code>	A flag indicating whether to drop attributes including names.
<code>x_name</code>	A string of the name of the object <code>x</code> .
<code>error</code>	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Details**

Useful when using flag to pass one of three options.

**Value**

An invisible copy of `x` (if it doesn't throw an error).

---

check_function	<i>Check Function</i>
----------------	-----------------------

---

**Description**

Checks if `x` is a function.

**Usage**

```
check_function(x, nargs = NA, x_name = substitute(x), error = TRUE)
```

**Arguments**

<code>x</code>	The object to check.
<code>nargs</code>	A count of the number of arguments or count range of the minimum and maximum number of arguments.
<code>x_name</code>	A string of the name of the object <code>x</code> .
<code>error</code>	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**Examples**

```
check_function(character, error = FALSE)
check_function(character, nargs = 0L, error = FALSE)
```

---

check_grepl	<i>Check Matches Regular Expression</i>
-------------	---

---

**Description**

Checks whether all the elements of an object match a regular expression.

**Usage**

```
check_grepl(
  x,
  pattern = ".*",
  regex = pattern,
  x_name = substitute(x),
  error = TRUE
)

check_regex(x, regex = ".*", x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The object to check.
pattern	A string of the regular expression.
regex	A string of the regular expression (deprecated for pattern).
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_nchar\(\)](#) and [check\\_pattern\(\)](#)

**Examples**

```
check_grepl("foo", "fo")
check_grepl("foo", "fo$", error = FALSE)
```

---

check\_homogenous      *Check Homogenous*

---

### Description

Checks whether the elements of `x` are all of the same class. It works on vectors, matrices and arrays which, by definition will always be homogenous and lists and data frames which may or may not be homogenous.

### Usage

```
check_homogenous(  
  x,  
  strict = FALSE,  
  recursive = FALSE,  
  x_name = substitute(x),  
  error = TRUE  
)
```

### Arguments

<code>x</code>	The object to check.
<code>strict</code>	A flag indicating whether all the objects must have identical classes or just share one or more classes.
<code>recursive</code>	A flag indicating whether the check should be applied recursively.
<code>x_name</code>	A string of the name of the object <code>x</code> .
<code>error</code>	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

### Value

An invisible copy of `x` (if it doesn't throw an error).

### See Also

[check\\_vector\(\)](#), [check\\_list\(\)](#) and [check\\_data\(\)](#)

### Examples

```
check_homogenous(1:2)  
check_homogenous(list(1,2))  
check_homogenous(list(1,TRUE), error = FALSE)
```



---

check_inherits	<i>Check Inherits</i>
----------------	-----------------------

---

**Description**

Checks if an object inherits from a class.

**Usage**

```
check_inherits(x, class, x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The object to check.
class	A string of the class x should inherit from.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_classes\(\)](#)

**Examples**

```
check_inherits(list(), "list")
check_inherits(list(), "numeric", error = FALSE)
```

---

check_int	<i>Check Int</i>
-----------	------------------

---

**Description**

Checks if x is a int (non-missing integer scalar with no attributes including names).

**Usage**

```
check_int(x, coerce = FALSE, x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The object to check.
coerce	A flag indicating whether to coerce a numeric (dbl) whole number to an int and drop attributes including names.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_scalar\(\)](#)

**Examples**

```
check_int(1, error = FALSE)
check_int(1L, error = FALSE)
check_int(1:2, error = FALSE)
```

---

check_integer	<i>Check Integer</i>
---------------	----------------------

---

**Description**

Checks if x is an integer vector with no attributes including names.

**Usage**

```
check_integer(x, coerce = FALSE, x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The object to check.
coerce	A flag indicating whether to coerce a numeric (dbl) whole number vector to an integer vector and drop attributes including names.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**[check\\_int\(\)](#)**Examples**

```
check_integer(1, error = FALSE)
check_integer(1L, error = FALSE)
check_integer(1:2, error = FALSE)
```

---

check_intersection	<i>Check Atomic Vector Intersection</i>
--------------------	---

---

**Description**

Checks that all the elements in atomic vector `x` intersect with those in atomic vector `y`.

**Usage**

```
check_intersection(  
  x,  
  y,  
  all_y = FALSE,  
  x_name = substitute(x),  
  y_name = substitute(y),  
  error = TRUE  
)
```

**Arguments**

<code>x</code>	The object to check.
<code>y</code>	The second atomic vector.
<code>all_y</code>	A flag indicating whether all the elements in <code>y</code> should have a match in <code>x</code> .
<code>x_name</code>	A string of the name of the object <code>x</code> .
<code>y_name</code>	A string of the name of the object <code>y</code> .
<code>error</code>	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of `x` (if it doesn't throw an error).

**See Also**[check\\_join\(\)](#)

**Examples**

```
x1 <- 1:3
x2 <- 1:4
check_intersection(x1, x2)
check_intersection(x2, x1, error = FALSE)
```

---

`check_join`*Check Join*

---

**Description**

Checks that the columns in data frame `x` form a many-to-one join with the corresponding columns in `y`, ie, the join is a unique key in `y` and all the rows in `x` have a match in `y`.

**Usage**

```
check_join(
  x,
  y,
  by = NULL,
  all_y = FALSE,
  x_name = substitute(x),
  y_name = substitute(y),
  error = TRUE
)
```

**Arguments**

<code>x</code>	The object to check.
<code>y</code>	The parent data frame.
<code>by</code>	A character vector or named character vector of the columns to join by.
<code>all_y</code>	A flag indicating whether all the rows in <code>y</code> should have a match in <code>x</code> .
<code>x_name</code>	A string of the name of the object <code>x</code> .
<code>y_name</code>	A string of the name of the object <code>y</code> .
<code>error</code>	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of `x` (if it doesn't throw an error).

**See Also**

[check\\_data\(\)](#)

**Examples**

```
data1 <- data.frame(x = 1:2)
data2 <- data.frame(x = 3:5, y = 2L)
check_join(data1, data2, error = FALSE)
check_join(data1, data2, by = c(x = "y"), error = FALSE)
```

---

`check_key`*Check Key*

---

**Description**

Checks that columns in a data frame represent a unique key. By default all the columns are checked.

**Usage**

```
check_key(
  x,
  key = names(x),
  na_distinct = FALSE,
  x_name = substitute(x),
  error = TRUE
)
```

**Arguments**

<code>x</code>	The data to check.
<code>key</code>	A character vector of the column names representing the key.
<code>na_distinct</code>	A flag specifying whether missing values should be considered distinct.
<code>x_name</code>	A string of the name of the object <code>x</code> .
<code>error</code>	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of `x` (if it doesn't throw an error).

**See Also**

[check\\_data\(\)](#)

**Examples**

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

---

check_length	<i>Check Length</i>
--------------	---------------------

---

**Description**

Checks whether the number of elements in an object is an exact number, within a range or 0 vs positive.

**Usage**

```
check_length(x, length = TRUE, x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The object to check.
length	A flag indicating whether x should have elements (versus no elements) or a missing value indicating no requirements or a count or count range of the number of elements or a count vector of the permitted number of elements.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_vector\(\)](#), [check\\_list\(\)](#) and [check\\_data\(\)](#)

**Examples**

```
check_length(2)
check_length(character(0), length = 0)
check_length(NULL, error = FALSE)
check_length(list(), error = FALSE)
```

---

check_length1	<i>Check Length One</i>
---------------	-------------------------

---

**Description**

Checks whether x is an object of length 1.

**Usage**

```
check_length1(x, x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The object to check.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**Examples**

```
check_length1(2)
check_length1(1:2, error = FALSE)
check_length1(NULL, error = FALSE)
check_length1(list(), error = FALSE)
```

---

check_levels	<i>Check Levels</i>
--------------	---------------------

---

**Description**

Checks the levels in a factor including the order and whether other levels are permitted.

**Usage**

```
check_levels(
  x,
  levels,
  exclusive = TRUE,
  order = TRUE,
  x_name = substitute(x),
  error = TRUE
)
```

**Arguments**

x	The object to check.
levels	A character vector of the levels.
exclusive	A flag indicating whether other levels are not permitted.
order	A flag indicating whether the object levels have to occur in the same order as names. To check whether x is an ordered factor use <code>check_vector(x, ordered(1))</code> .
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_nlevels\(\)](#) and [check\\_vector\(\)](#)

**Examples**

```
check_levels(1, c("x", "y"), error = FALSE)
check_levels(factor(1), c("x", "y"), error = FALSE)
```

---

check\_lgl

*Check Flag*

---

**Description**

Checks if x is a flag (non-missing logical scalar with no attributes including names).

**Usage**

```
check_lgl(x, coerce = FALSE, x_name = substitute(x), error = TRUE)
check_flag(x, coerce = FALSE, x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The object to check.
coerce	A flag indicating whether to drop attributes including names.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.



**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_scalar\(\)](#)

**Examples**

```
check_lgl(1, error = FALSE)
check_lgl(FALSE, error = FALSE)
check_lgl(c(FALSE, TRUE), error = FALSE)
```

---

check\_list

*Check List*

---

**Description**

Checks whether an object is a list and optionally the names and values of its elements.

**Usage**

```
check_list(
  x,
  values = NULL,
  length = NA,
  unique = FALSE,
  named = NA,
  exclusive = FALSE,
  order = FALSE,
  x_name = substitute(x),
  error = TRUE
)
```

**Arguments**

x	The object to check.
values	NULL (default) or a character vector specifying the column names or a named list specifying the column names and values.
length	A flag indicating whether x should have elements (versus no elements) or a missing value indicating no requirements or a count or count range of the number of elements or a count vector of the permitted number of elements.
unique	A flag indicating whether the values must be unique.
named	A flag indicating whether the list must be named or unnamed or a regular expression that must match all the names or count or count range of the number of characters in the names or NA if it doesn't matter if the list is named.

exclusive	A flag indicating whether other elements are not permitted.
order	A flag indicating whether the elements have to occur in the same order as values.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_length\(\)](#) and [check\\_unique\(\)](#)

**Examples**

```
check_list(list())
check_list(list(x1 = 2, x2 = 1:2), values = list(x1 = 1, x2 = 1L))
```

---

check\_logical

*Check Logical*

---

**Description**

Checks if x is a logical vector with no attributes including names.

**Usage**

```
check_logical(x, coerce = FALSE, x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The object to check.
coerce	A flag indicating whether to drop attributes including names.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_scalar\(\)](#)

**Examples**

```
check_logical(1, error = FALSE)
check_logical(FALSE, error = FALSE)
check_logical(c(FALSE, TRUE), error = FALSE)
```

---

check\_missing\_colnames

*Check Missing Colnames*

---

**Description**

Checks whether specific colnames are missing from a data frame.

**Usage**

```
check_missing_colnames(x, colnames, x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The data to check.
colnames	A character vector of the column names that must be missing from x.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_colnames\(\)](#) and [check\\_data\(\)](#)

**Examples**

```
data <- data.frame(x = 1, y = 2, z = 0)
check_missing_colnames(data, c("y", "x", "a"), error = FALSE)
check_missing_colnames(data, "a", error = FALSE)
```

---

check\_missing\_names      *Check Missing Names*

---

**Description**

Checks whether specific names are missing from an object.

**Usage**

```
check_missing_names(x, names, x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The named object to check.
names	A character vector of the names that must be missing from x.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_names\(\)](#)

**Examples**

```
vec <- c(x = 1, y = 2, z = 0)
check_missing_names(vec, c("y", "x", "a"), error = FALSE)
check_missing_names(vec, "a", error = FALSE)
```

---

check\_name      *Check Name*

---

**Description**

Checks whether each element of a character vector is a syntactically valid name.

**Usage**

```
check_name(x, x_name = substitute(x), coerce = FALSE, error = TRUE)
```

**Arguments**

x	The object to check.
x_name	A string of the name of the object x.
coerce	A flag specifying whether to coerce a factor to a character vector and drop attributes including names.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_character\(\)](#)

**Examples**

```
vec <- c("x", "x.y", "x y")
check_name(vec, error = FALSE)
```

---

check\_named

*Check Named*

---

**Description**

Checks whether an object is named.

**Usage**

```
check_named(
  x,
  nchar = c(0L, chk_max_int()),
  pattern = ".*",
  regex = pattern,
  unique = FALSE,
  x_name = substitute(x),
  error = TRUE
)
```

**Arguments**

x	The object to check.
nchar	A count or count range of the number of characters.
pattern	A string of the regular expression that must match all names.
regex	A string of the regular expression that must match all names.

unique	A flag indicating whether the names must be unique.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_unnamed\(\)](#), [check\\_names\(\)](#) and [check\\_missing\\_names\(\)](#)

**Examples**

```
check_named(2, error = FALSE)
x <- 1
names(x) <- "y"
check_named(x, error = FALSE)
```

---

check\_names

*Check Names*

---

**Description**

Checks the names of an object as returned by the `names()` function. The function can check the order of the names and whether other names are permitted.

**Usage**

```
check_names(
  x,
  names = character(0),
  exclusive = FALSE,
  order = FALSE,
  unique = FALSE,
  complete = TRUE,
  x_name = substitute(x),
  error = TRUE
)
```

**Arguments**

x	The object to check.
names	A character vector of the names.
exclusive	A flag indicating whether other names are not permitted.

order	A flag indicating whether the object names have to occur in the same order as names.
unique	A flag indicating whether all the object names have to be unique.
complete	A flag indicating whether all the possible names have to be represented.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_named\(\)](#)

**Examples**

```
vec <- c(x = 1, y = 2, z = 0)
check_names(vec, c("y", "x"), error = FALSE)
check_names(vec, c("y", "x"), exclusive = TRUE, error = FALSE)
check_names(vec, c("y", "x"), order = TRUE, error = FALSE)
check_names(vec, c("a"), error = FALSE)
```

---

check_nchar	<i>Check Number of Characters</i>
-------------	-----------------------------------

---

**Description**

Checks the number of characters in the elements of an object.

**Usage**

```
check_nchar(x, nchar = TRUE, x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The object to check.
nchar	A flag indicating whether x should have characters or a missing value indicating no requirements or a count or count range of the number of characters.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_pattern\(\)](#) and [check\\_regex\(\)](#)

**Examples**

```
check_nchar(c("foo", "bar"), nchar = 3)
```

---

check_ncol	<i>Check Number of Columns</i>
------------	--------------------------------

---

**Description**

Checks the number of columns of a data frame.

**Usage**

```
check_ncol(x, ncol = TRUE, x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The object to check.
ncol	A flag indicating whether x should have columns (versus no columns) or a missing value indicating no requirements or a count or count range of the number of columns.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_nrow\(\)](#) and [check\\_data\(\)](#)

**Examples**

```
check_ncol(data.frame(x = 1), error = FALSE)
check_ncol(data.frame(x = 1:2), ncol = 1, error = FALSE)
```



---

check_neg_dbl	<i>Check Negative Dbl</i>
---------------	---------------------------

---

**Description**

Checks if *x* is a negative dbl (non-missing numeric scalar with no attributes including names).

**Usage**

```
check_neg_dbl(x, coerce = FALSE, x_name = substitute(x), error = TRUE)
```

**Arguments**

<i>x</i>	The object to check.
<i>coerce</i>	A flag indicating whether to coerce a scalar integer to a dbl and drop attributes including names.
<i>x_name</i>	A string of the name of the object <i>x</i> .
<i>error</i>	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of *x* (if it doesn't throw an error).

**See Also**

[check\\_dbl\(\)](#)

**Examples**

```
check_neg_dbl(1, error = FALSE)
check_neg_dbl(0L, error = FALSE)
check_neg_dbl(0, error = FALSE)
```

---

check_neg_int	<i>Check Negative Int</i>
---------------	---------------------------

---

**Description**

Checks if *x* is a negative int (non-missing integer scalar with no attributes including names).

**Usage**

```
check_neg_int(x, coerce = FALSE, x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The object to check.
coerce	A flag indicating whether to coerce a numeric (dbl) whole number to an int and drop attributes including names.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_int\(\)](#)

**Examples**

```
check_neg_int(0L, error = FALSE)
check_neg_int(-1L, error = FALSE)
```

---

check_nlevels	<i>Check nlevels</i>
---------------	----------------------

---

**Description**

Checks the number of levels of an object.

**Usage**

```
check_nlevels(x, nlevels = TRUE, x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The data to check.
nlevels	A flag indicating whether x should have elements (versus no elements) or a missing value indicating no requirements or a count or count range of the number of elements.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_levels\(\)](#) and [check\\_vector\(\)](#)

**Examples**

```
check_nlevels(factor(1), error = FALSE)
check_nlevels(factor(1), nlevels = 2, error = FALSE)
```

---

check_nonneg_dbl	<i>Check Non-Negative Dbl</i>
------------------	-------------------------------

---

**Description**

Checks if `x` is a non-negative dbl (non-missing numeric scalar with no attributes including names).

**Usage**

```
check_nonneg_dbl(x, coerce = FALSE, x_name = substitute(x), error = TRUE)
```

**Arguments**

<code>x</code>	The object to check.
<code>coerce</code>	A flag indicating whether to coerce a scalar integer to a dbl and drop attributes including names.
<code>x_name</code>	A string of the name of the object <code>x</code> .
<code>error</code>	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of `x` (if it doesn't throw an error).

**See Also**

[check\\_dbl\(\)](#)

**Examples**

```
check_nonneg_dbl(1, error = FALSE)
check_nonneg_dbl(0L, error = FALSE)
check_nonneg_dbl(0, error = FALSE)
```

---

check_noneg_int	<i>Check Non-Negative Int</i>
-----------------	-------------------------------

---

**Description**

Checks if x is a count (non-missing non-negative integer scalar with no attributes including names).

**Usage**

```
check_noneg_int(x, coerce = FALSE, x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The object to check.
coerce	A flag indicating whether to coerce a non-negative numeric (dbl) whole number to a count and drop attributes (including names).
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_int\(\)](#)

**Examples**

```
check_noneg_int(1, error = FALSE)
check_noneg_int(0L, error = FALSE)
check_noneg_int(1L, error = FALSE)
```

---

check_no_attributes	<i>Check No Attributes</i>
---------------------	----------------------------

---

**Description**

Checks an object has no attributes.

**Usage**

```
check_no_attributes(  
  x,  
  names = TRUE,  
  class = TRUE,  
  x_name = substitute(x),  
  error = TRUE  
)
```

**Arguments**

x	The object to check.
names	A flag specifying whether names should be considered an attribute.
class	A flag specifying whether class should be considered an attribute.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_attributes\(\)](#)

**Examples**

```
x <- 1  
attributes(x) <- list(y = 2L)  
check_no_attributes(x, error = FALSE)
```

---

check\_nrow

*Check Number of Rows*

---

**Description**

Checks the number of rows of a data frame.

**Usage**

```
check_nrow(x, nrow = TRUE, x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The object to check.
nrow	A flag indicating whether x should have rows (versus no rows) or a missing value indicating no requirements or a count or count range of the number of rows. @return An invisible copy of x (if it doesn't throw an error).
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**See Also**

[check\\_ncol\(\)](#) and [check\\_data\(\)](#)

**Examples**

```
check_nrow(data.frame(x = 1), error = FALSE)
check_nrow(data.frame(x = integer(0)), error = FALSE)
check_nrow(data.frame(x = 1:2), nrow = 1, error = FALSE)
```

---

check\_null

*Check NULL*

---

**Description**

Checks whether an object is NULL.

**Usage**

```
check_null(x, x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The object to check.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**Examples**

```
check_null(1, error = FALSE)
check_null(NULL, error = FALSE)
```

---

check_numeric	<i>Check Numeric</i>
---------------	----------------------

---

### Description

Checks if `x` is an numeric (double) vector with no attributes including names.

### Usage

```
check_numeric(x, coerce = FALSE, x_name = substitute(x), error = TRUE)
```

```
check_double(x, coerce = FALSE, x_name = substitute(x), error = TRUE)
```

### Arguments

<code>x</code>	The object to check.
<code>coerce</code>	A flag indicating whether to coerce a integer vector to an double vector and drop attributes including names.
<code>x_name</code>	A string of the name of the object <code>x</code> .
<code>error</code>	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

### Value

An invisible copy of `x` (if it doesn't throw an error).

### See Also

[check\\_number\(\)](#)

### Examples

```
check_numeric(1, error = FALSE)
check_numeric(1L, error = FALSE)
check_numeric(1:2, error = FALSE)
```

---

check_pattern	<i>Check Pattern</i>
---------------	----------------------

---

**Description**

Checks whether all or some of the elements of `x` match `pattern` using `grepl()`.

**Usage**

```
check_pattern(x, pattern, all = TRUE, x_name = substitute(x), error = TRUE)
```

**Arguments**

<code>x</code>	The object to check.
<code>pattern</code>	A string of the regular expression.
<code>all</code>	A flag indicating whether all or some of the element must match <code>pattern</code> .
<code>x_name</code>	A string of the name of the object <code>x</code> .
<code>error</code>	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of `x` (if it doesn't throw an error).

**See Also**

[check\\_nchar\(\)](#) and [check\\_regex\(\)](#)

---

check_pos_dbl	<i>Check Positive Dbl</i>
---------------	---------------------------

---

**Description**

Checks if `x` is a positive dbl (non-missing numeric scalar with no attributes including names).

**Usage**

```
check_pos_dbl(x, coerce = FALSE, x_name = substitute(x), error = TRUE)
```

**Arguments**

<code>x</code>	The object to check.
<code>coerce</code>	A flag indicating whether to coerce a scalar integer to a dbl and drop attributes including names.
<code>x_name</code>	A string of the name of the object <code>x</code> .
<code>error</code>	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.



**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_dbl\(\)](#)

**Examples**

```
check_pos_dbl(1, error = FALSE)
check_pos_dbl(0L, error = FALSE)
check_pos_dbl(0, error = FALSE)
```

---

check\_pos\_int

*Check Positive Int*

---

**Description**

Checks if x is a positive int (non-missing integer scalar with no attributes including names).

**Usage**

```
check_pos_int(x, coerce = FALSE, x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The object to check.
coerce	A flag indicating whether to coerce a numeric (dbl) whole number to an int and drop attributes including names.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_int\(\)](#)

**Examples**

```
check_pos_int(0, error = FALSE)
check_pos_int(1L, error = FALSE)
check_pos_int(1:2, error = FALSE)
```

---

check_prob	<i>Check Probability</i>
------------	--------------------------

---

### Description

Checks if `x` is a probability (non-missing dbl between 0 and 1 inclusive with no attributes including names).

### Usage

```
check_prob(x, coerce = FALSE, x_name = substitute(x), error = TRUE)
```

```
check_probability(x, coerce = FALSE, x_name = substitute(x), error = TRUE)
```

```
check_prop(x, coerce = FALSE, x_name = substitute(x), error = TRUE)
```

### Arguments

<code>x</code>	The object to check.
<code>coerce</code>	A flag indicating whether to coerce an integer to a dbl and drop attributes including names.
<code>x_name</code>	A string of the name of the object <code>x</code> .
<code>error</code>	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

### Value

An invisible copy of `x` (if it doesn't throw an error).

### See Also

[check\\_scalar\(\)](#)

### Examples

```
check_prob(1, error = FALSE)
check_prob(1.1, error = FALSE)
check_prob(c(0, 1), error = FALSE)
```

---

check_props	<i>Check Proportions</i>
-------------	--------------------------

---

**Description**

Checks if x is proportions vector - non-missing dbls between 0 and 1 that sum to 1.

**Usage**

```
check_props(x, x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The object to check.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

---

check_rbind	<i>Check Row Bind</i>
-------------	-----------------------

---

**Description**

Checks whether a data frame has the same columns of the same classes as a second data frame which means they can be `rbind()`ed without a problem.

**Usage**

```
check_rbind(  
  x,  
  y,  
  exclusive = TRUE,  
  order = FALSE,  
  x_name = substitute(x),  
  y_name = substitute(y),  
  error = TRUE  
)
```

**Arguments**

x	The first data frame.
y	The second data frame.
exclusive	A flag indicating whether other columns are not permitted.
order	A flag indicating whether the columns have to occur in the same order.
x_name	A string of the name of the object x.
y_name	A string of the name of the object y.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_join\(\)](#)

**Examples**

```
check_rbind(datasets::mtcars, datasets::mtcars)
```

---

check\_scalar

*Check Scalar*

---

**Description**

Checks whether an object is an atomic vector with one element.

**Usage**

```
check_scalar(  
  x,  
  values = NULL,  
  named = FALSE,  
  attributes = named,  
  only = FALSE,  
  x_name = substitute(x),  
  error = TRUE  
)
```

**Arguments**

x	The object to check.
values	NULL or a vector specifying the values.
named	A flag indicating whether the scalar must be named or unnamed or NA if it doesn't matter if the scalar is named.
attributes	A flag indicating whether the scalar must or must not have attributes or NA if it doesn't matter if the scalar is named.
only	A flag indicating whether only the actual values are permitted. It only affects values with two or less non-missing elements.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_vector\(\)](#)

**Examples**

```
check_scalar(1)
check_scalar(c(1,2), error = FALSE)
check_scalar(1, c(2,3), error = FALSE)
```

---

check\_sorted

*Check Sorted*

---

**Description**

Checks whether object x is sorted using `!is.unsorted(x, na.rm = TRUE)`.

**Usage**

```
check_sorted(x, x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The object to check.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_vector\(\)](#) and [check\\_list\(\)](#)

**Examples**

```
check_sorted(1:2, error = FALSE)
check_sorted(2:1, error = FALSE)
```

---

check_tzone	<i>Check TimeZone</i>
-------------	-----------------------

---

**Description**

Checks an objects tzone attribute.

**Usage**

```
check_tzone(x, tzone = "UTC", x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The object to check.
tzone	A string of the time zone.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**Examples**

```
check_tzone(Sys.Date(), error = FALSE)
x <- as.POSIXct("2000-01-02 03:04:55", tz = "Etc/GMT+8")
check_tzone(x, tzone = "PST8PDT", error = FALSE)
```

---

check_unique	<i>Check Unique</i>
--------------	---------------------

---

**Description**

Checks whether all elements of an object are unique.

**Usage**

```
check_unique(x, x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The object to check.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_vector\(\)](#)

**Examples**

```
check_unique(2, error = FALSE)
check_unique(c(2,2), error = FALSE)
check_unique(1:2, error = FALSE)
check_unique(character(0), error = FALSE)
check_unique(NULL, error = FALSE)
check_unique(list(), error = FALSE)
```

---

check_unnamed	<i>Check Unnamed</i>
---------------	----------------------

---

**Description**

Checks whether an objects is unnamed.

**Usage**

```
check_unnamed(x, x_name = substitute(x), error = TRUE)
```

**Arguments**

x	The object to check.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**See Also**

[check\\_named\(\)](#), [check\\_names\(\)](#) and [check\\_missing\\_names\(\)](#)

**Examples**

```
check_unnamed(2, error = FALSE)
x <- 1
names(x) <- "y"
check_unnamed(x, error = FALSE)
```

---

check\_unused

*Check Unused*

---

**Description**

Checks whether ... is unused. It can only be used in functions.

**Usage**

```
check_unused(..., x_name = "...", error = TRUE)
```

**Arguments**

...	The arguments to check.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**Examples**

```
fun <- function(...) check_unused(..., error = FALSE)
fun()
fun(1)
```



---

check_vector	<i>Check Atomic Vector</i>
--------------	----------------------------

---

**Description**

Check Atomic Vector

**Usage**

```
check_vector(
  x,
  values = NULL,
  length = NA,
  unique = FALSE,
  sorted = FALSE,
  named = NA,
  attributes = named,
  names = TRUE,
  class = TRUE,
  only = FALSE,
  x_name = substitute(x),
  error = TRUE
)
```

**Arguments**

x	The object to check.
values	NULL or a vector specifying the values.
length	A flag indicating whether x should have elements (versus no elements) or a missing value indicating no requirements or a count or count range of the number of elements or a count vector of the permitted number of elements.
unique	A flag indicating whether the values must be unique.
sorted	A flag indicating whether the vector must be sorted.
named	A flag indicating whether the vector must be named or unnamed or NA if it doesn't matter.
attributes	A flag indicating whether the vector must or must not have attributes or NA if it doesn't matter.
names	A flag specifying whether names should be considered an attribute.
class	A flag specifying whether class should be considered an attribute.
only	A flag indicating whether only the actual values are permitted. It only affects values with two or less non-missing elements.
x_name	A string of the name of the object x.
error	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

**Value**

An invisible copy of x (if it doesn't throw an error).

**Examples**

```
check_vector(2:1, length = 3, sorted = TRUE, named = TRUE, error = FALSE)
check_vector(c("one", "two", "four"), values = c("one", "two", "two"), error = FALSE)
```

---

chk\_deparse

*Deparse*

---

**Description**

chk\_deparse is a wrapper on [deparse\(\)](#) that sets a missing value to be "NA"

**Usage**

```
chk_deparse(x)
```

**Arguments**

x                    A substituted object to deparse

**Value**

A string

**See Also**

[deparse\(\)](#)

**Examples**

```
chk_deparse(1^2)
```

---

chk_fail	<i>Fail</i>
----------	-------------

---

**Description**

Fail

**Usage**

```
chk_fail(..., error)
```

**Arguments**

...	The message.
error	A flag indicating whether to return an error (the default) or a warning.

---

chk_max_dbl	<i>Max Double</i>
-------------	-------------------

---

**Description**

Max Double

**Usage**

```
chk_max_dbl()
```

**Value**

An dbl of the maximum numeric value for the system.

**Examples**

```
chk_max_dbl()
```

---

`chk_max_int`*Max Int*

---

**Description**

Max Int

**Usage**`chk_max_int()`**Value**

An int of the maximum integer value for the system.

**Examples**`chk_max_int()`

---

`chk_min_dbl`*Min Double*

---

**Description**

Min Double

**Usage**`chk_min_dbl()`**Value**

An dbl of the minimum numeric value for the system.

**Examples**`chk_min_dbl()`

---

chk_min_int	<i>Min Integer</i>
-------------	--------------------

---

**Description**

Min Integer

**Usage**

chk\_min\_int()

**Value**

An int of the minimum integer value for the system.

**Examples**

chk\_min\_int()

---

chk_tiny_dbl	<i>Tiny Positive Double</i>
--------------	-----------------------------

---

**Description**

Tiny Positive Double

**Usage**

chk\_tiny\_dbl()

**Value**

An dbl of the tiniest positive numeric value for the system.

**Examples**

chk\_tiny\_dbl()

# Index

check\_attributes, 4  
check\_attributes(), 37  
check\_character, 5  
check\_character(), 29  
check\_chr, 5  
check\_classes, 6  
check\_classes(), 17  
check\_colnames, 7  
check\_colnames(), 10, 27  
check\_count, 8  
check\_data, 9  
check\_data(), 8, 16, 20–22, 27, 32, 38  
check\_date, 10  
check\_datetime (check\_dttm), 12  
check\_datetime(), 11  
check\_day (check\_date), 10  
check\_dbl, 11  
check\_dbl(), 33, 35, 41  
check\_double (check\_numeric), 39  
check\_dttm, 12  
check\_environment, 13  
check\_flag (check\_lgl), 24  
check\_flag\_na, 14  
check\_function, 14  
check\_grepl, 15  
check\_homogenous, 16  
check\_inherits, 17  
check\_inherits(), 7  
check\_int, 17  
check\_int(), 5, 19, 34, 36, 41  
check\_integer, 18  
check\_intersection, 19  
check\_join, 20  
check\_join(), 19, 44  
check\_key, 21  
check\_key(), 10  
check\_length, 22  
check\_length(), 26  
check\_length1, 23  
check\_levels, 23  
check\_levels(), 35  
check\_lgl, 24  
check\_list, 25  
check\_list(), 4, 16, 22, 46  
check\_logical, 26  
check\_missing\_colnames, 27  
check\_missing\_colnames(), 8  
check\_missing\_names, 28  
check\_missing\_names(), 30, 48  
check\_name, 28  
check\_named, 29  
check\_named(), 31, 48  
check\_names, 30  
check\_names(), 28, 30, 48  
check\_nchar, 31  
check\_nchar(), 15, 40  
check\_ncol, 32  
check\_ncol(), 38  
check\_neg\_dbl, 33  
check\_neg\_int, 33  
check\_nlevels, 34  
check\_nlevels(), 24  
check\_no\_attributes, 36  
check\_nonneg\_dbl, 35  
check\_nonneg\_int, 36  
check\_nrow, 37  
check\_nrow(), 10, 32  
check\_null, 38  
check\_number (check\_dbl), 11  
check\_number(), 39  
check\_numeric, 39  
check\_pattern, 40  
check\_pattern(), 15, 32  
check\_pos\_dbl, 40  
check\_pos\_int, 41  
check\_prob, 42  
check\_probability (check\_prob), 42  
check\_prop (check\_prob), 42

check\_props, 43  
check\_rbind, 43  
check\_regex (check\_grepl), 15  
check\_regex(), 32, 40  
check\_scalar, 44  
check\_scalar(), 6, 9, 12, 13, 18, 25, 26, 42  
check\_sorted, 45  
check\_string (check\_chr), 5  
check\_tzone, 46  
check\_unique, 47  
check\_unique(), 26  
check\_unnamed, 47  
check\_unnamed(), 30  
check\_unused, 48  
check\_vector, 49  
check\_vector(), 16, 22, 24, 35, 45–47  
checkor, 3  
chk\_deparse, 50  
chk\_fail, 51  
chk\_max\_dbl, 51  
chk\_max\_int, 52  
chk\_min\_dbl, 52  
chk\_min\_int, 53  
chk\_tiny\_dbl, 53  
  
deparse(), 50  
  
grepl(), 40  
  
rbind(), 43