Package: bkcheck (via r-universe)

November 12, 2026

Title Fast and Simple Argument Checks

Version 2026.1.14.999

Description A collection of functions for argument checking and property-based testing. Designed to be fast, simple, and suitable for most purposes.

URL https://brettklamer.com/work/bkcheck, https://bitbucket.org/bklamer/bkcheck

License MIT + file LICENSE

Depends R (>= 4.3.0)

Imports bkbase

Suggests tinytest

LazyData TRUE

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.0.0

Repository https://bklamer.r-universe.dev

Date/Publication 2026-06-15 00:24:16 UTC

RemoteUrl https://bitbucket.org/bklamer/bkcheck

RemoteRef HEAD

RemoteSha 8dec3d1bbe952b06f588bff8b9c320ae012c0a22

Contents

check_all_arg_in_x 3
check_all_x_in_arg 4
check_any_arg_in_x 5
check_any_x_in_arg 7
check_atomic 8

check_character . . . 9
check_data_frame . . . 11
check_data_frame_or_matrix . . . 12
check_date . . . 14
check_date_or_datetime . . . 15
check_datetime . . . 17
check_double . . . 18
check_equals . . . 20
check_factor . . . 21
check_file_exists . . . 22
check_finite . . . 24
check_flag . . . 25
check_formula . . . 27
check_function . . . 28
check_function_or_numeric . . . 29
check_inherits . . . 31
check_integer . . . 32
check_length_equals . . . 34
check_list . . . 35
check_logical . . . 37
check_lt . . . 38
check_matrix . . . 40
check_monotonic_decreasing . . . 42
check_named . . . 44
check_names_in_arg . . . 45
check_ncol_equals . . . 46
check_nonempty_string . . . 48
check_nonmissing . . . 49
check_nrow_equals . . . 51
check_numeric . . . 52
check_numeric_or_logical . . . 54
check_probability . . . 55
check_range . . . 57
check_real . . . 59
check_scalar_arg_in_x . . . 60
check_scalar_character . . . 62
check_scalar_date . . . 63
check_scalar_datetime . . . 64
check_scalar_double . . . 66
check_scalar_factor . . . 67
check_scalar_integer . . . 69
check_scalar_logical . . . 70
check_scalar_numeric . . . 72
check_scalar_probability . . . 73
check_scalar_whole_number . . . 75
check_scalar_whole_numeric . . . 76
check_strict_decreasing . . . 78
check_string . . . 80

check_table 81
check_unique 82
check_valid 84
check_valid_numeric 85
check_whole_number 87
check_whole_numeric 88
insert_value 90
rand_chars 91
sim_character 92
sim_data.frame 94
sim_date 95
sim_datetime 96
sim_double 98
sim_factor 100
sim_integer 101
sim_logical 103

Index **106**

`check_all_arg_in_x` *Check all* `arg` *in* `x`

Description

Check if `all(arg %in% x)`.

Usage

```
check_all_arg_in_x(arg, x, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

`arg`	(object) The argument to check.
`x`	(Atomic vector) All of `arg` must have matching values in `x`.
`signal`	(string: `c("error", "warning", "message")`) Must be one of `"error"`, `"warning"`, or `"message"`. `"error"` (default) calls `base::stop()` to signal an error message. `"warning"` calls `base::warning()` to signal a warning message. `"message"` calls `base::message()` to signal a message.
`msg`	(string or `NULL`: `NULL`) A string that replaces the default message. Set as `NULL` to use the default message.
`call.`	(Scalar logical: `c(FALSE, TRUE)`) Passed to `stop()` or `warning()`. The default `msg` includes it's own formatted call, hence `call.` should be `FALSE` (default). If you insert your own `msg`, then you may want `call. = TRUE`.

Value

`invisible(arg)` or condition determined by argument `signal`.

See Also

Other check_relation: `check_all_x_in_arg()`, `check_any_arg_in_x()`, `check_any_x_in_arg()`, `check_equals()`, `check_finite()`, `check_length_equals()`, `check_lt()`, `check_monotonic_decreasing()`, `check_names_in_arg()`, `check_range()`, `check_scalar_arg_in_x()`, `check_strict_decreasing()`, `check_unique()`

Examples

```
#--------------------------------------------------------------------------
# check_all_arg_in_x() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  vec <- 1:10
  x |>
    check_all_arg_in_x(vec)
}

f(1:2)
try(f(0:10))
```

`check_all_x_in_arg` *Check all* `x` *in* `arg`

Description

Check if `all(x %in% arg)`.

Usage

```
check_all_x_in_arg(arg, x, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

`arg`	(object) The argument to check.
`x`	(Atomic vector) All of `x` must have matching values in `arg`.
`signal`	(string: `c("error", "warning", "message")`) Must be one of `"error"`, `"warning"`, or `"message"`. `"error"` (default) calls `base::stop()` to signal an error message. `"warning"` calls `base::warning()` to signal a warning message. `"message"` calls `base::message()` to signal a message.

msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_relation: check_all_arg_in_x(), check_any_arg_in_x(), check_any_x_in_arg(), check_equals(), check_finite(), check_length_equals(), check_lt(), check_monotonic_decreasing(), check_names_in_arg(), check_range(), check_scalar_arg_in_x(), check_strict_decreasing(), check_unique()

Examples

```
#--------------------------------------------------------------------------
# check_all_x_in_arg() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  vec <- 1:10
  x |>
    check_all_x_in_arg(vec)
}

f(0:10)
try(f(1:2))
```

---

check_any_arg_in_x *Check any* arg *in* x

Description

Check if any(arg %in% x).

Usage

```
check_any_arg_in_x(arg, x, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
x	(Atomic vector) At least one element of arg must have a matching value in x.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_relation: check_all_arg_in_x(), check_all_x_in_arg(), check_any_x_in_arg(), check_equals(), check_finite(), check_length_equals(), check_lt(), check_monotonic_decreasing(), check_names_in_arg(), check_range(), check_scalar_arg_in_x(), check_strict_decreasing(), check_unique()

Examples

```
#--------------------------------------------------------------------------
# check_any_arg_in_x() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  vec <- 1:2
  x |>
    check_any_arg_in_x(vec)
}

f(1:10)
try(f(3:10))
```

check_any_x_in_arg *Check any* x *in* arg

Description

Check if any(x %in% arg).

Usage

```
check_any_x_in_arg(arg, x, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
x	(Atomic vector) At least one element of x must have a matching value in arg.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_relation: check_all_arg_in_x(), check_all_x_in_arg(), check_any_arg_in_x(), check_equals(), check_finite(), check_length_equals(), check_lt(), check_monotonic_decreasing(), check_names_in_arg(), check_range(), check_scalar_arg_in_x(), check_strict_decreasing(), check_unique()

Examples

```
#--------------------------------------------------------------------------
# check_any_x_in_arg() examples
#--------------------------------------------------------------------------
library(bkcheck)
```

```
f <- function(x) {
  vec <- 1:2
  x |>
    check_any_x_in_arg(vec)
}

f(1:10)
try(f(3:10))
```

---

check_atomic *Check atomic vector*

Description

Check if arg is an atomic vector.

Usage

```
check_atomic(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_nu
check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#--------------------------------------------------------------------------
# check_atomic() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_atomic()
}

f(TRUE)
f(1:10)
f(letters)
f(factor(letters))
f(logical(0))
f(NA)
f(NaN)

try(f(NULL))
try(f(matrix(1)))
try(f(data.frame(1)))
try(f(list(1)))
```

---

check_character *Check character vector*

Description

Check if arg is a character vector.

Usage

```
check_character(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_nu check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#--------------------------------------------------------------------------
# check_character() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_character()
}

logical <- TRUE
integer <- 1:2
numeric <- c(1, 1.5)
character <- letters[1:4]
```

```
factor <- factor(character)

f(character)

try(f(logical))
try(f(integer))
try(f(numeric))
try(f(factor))
```

`check_data_frame` *Check data.frame*

Description

Check if `arg` is a data.frame.

Usage

```
check_data_frame(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

`arg`	(object) The argument to check.
`signal`	(string: `c("error", "warning", "message")`) Must be one of `"error"`, `"warning"`, or `"message"`. `"error"` (default) calls `base::stop()` to signal an error message. `"warning"` calls `base::warning()` to signal a warning message. `"message"` calls `base::message()` to signal a message.
`msg`	(string or `NULL`: `NULL`) A string that replaces the default message. Set as `NULL` to use the default message.
`call.`	(Scalar logical: `c(FALSE, TRUE)`) Passed to `stop()` or `warning()`. The default `msg` includes it's own formatted call, hence `call.` should be `FALSE` (default). If you insert your own `msg`, then you may want `call. = TRUE`.

Value

`invisible(arg)` or condition determined by argument `signal`.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_nu
check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#------------------------------------------------------------------------
# check_data_frame() examples
#------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_data_frame()
}

f(data.frame(1))

try(f(list(1)))
try(f(TRUE))
try(f(1L))
try(f(1))
try(f("a"))
try(f(factor("a")))
try(f(NULL))
```

`check_data_frame_or_matrix` *Check data.frame or matrix*

Description

Check if `arg` is a data.frame or matrix.

Usage

```
check_data_frame_or_matrix(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_nur check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#--------------------------------------------------------------------------
# check_data_frame_or_matrix() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_data_frame_or_matrix()
}

mat <- matrix(1:2)
integer <- 1:2
numeric <- c(1, 1.5)
character <- letters[1:4]
```

```
factor <- factor(character)

f(mat)
f(data.frame(1))

try(f(integer))
try(f(numeric))
try(f(character))
try(f(factor))
```

---

check_date *Check date vector*

Description

Check if arg is a date vector.

Usage

```
check_date(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_nur check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#--------------------------------------------------------------------------
# check_date() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_date()
}

f(Sys.Date())
f(c(Sys.Date(), Sys.Date() - 1))

try(f(TRUE))
try(f(Sys.time()))
try(f(1L))
try(f(1))
try(f(factor("a")))
try(f("a"))
```

`check_date_or_datetime` *Check date or datetime vector*

Description

Check if `arg` is a date or datetime vector.

Usage

```
check_date_or_datetime(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_numeric(), check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#--------------------------------------------------------------------------
# check_date_or_datetime() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_date_or_datetime()
}

f(Sys.Date())
f(c(Sys.Date(), Sys.Date() - 1))
f(Sys.time())

try(f(TRUE))
```

```
try(f(1L))
try(f(1))
try(f(factor("a")))
try(f("a"))
```

---

check_datetime *Check datetime vector*

Description

Check if arg is a datetime vector.

Usage

```
check_datetime(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(),

check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_nu
check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(),
check_whole_numeric()

Examples

```
#------------------------------------------------------------------------
# check_datetime() examples
#------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_datetime()
}

f(Sys.time())
f(c(Sys.time(), Sys.time() - 1))

try(f(TRUE))
try(f(Sys.Date()))
try(f(1L))
try(f(1))
try(f(factor("a")))
try(f("a"))
```

---

check_double *Check double-precision vector*

Description

Check if arg is a double-precision vector.

Usage

```
check_double(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.

msg (string or NULL: NULL)
A string that replaces the default message. Set as NULL to use the default message.

call. (Scalar logical: c(FALSE, TRUE))
Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_nu check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#--------------------------------------------------------------------------
# check_double() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_double()
}

logical <- TRUE
integer <- 1:2
numeric <- c(1, 1.5)
character <- letters[1:4]
factor <- factor(character)

f(numeric)

try(f(logical))
try(f(integer))
try(f(character))
try(f(factor))
```

check_equals *Check equal to*

Description

Check if `arg` is equal to `x`. `arg` and `x` must be atomic vectors of the same length.

Usage

```
check_equals(arg, x, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

`arg`	(object) The argument to check.
`x`	(Atomic vector) The vector which `arg` must be equal to. Must be the same length as `arg`.
`signal`	(string: `c("error", "warning", "message")`) Must be one of `"error"`, `"warning"`, or `"message"`. `"error"` (default) calls `base::stop()` to signal an error message. `"warning"` calls `base::warning()` to signal a warning message. `"message"` calls `base::message()` to signal a message.
`msg`	(string or `NULL`: `NULL`) A string that replaces the default message. Set as `NULL` to use the default message.
`call.`	(Scalar logical: `c(FALSE, TRUE)`) Passed to `stop()` or `warning()`. The default `msg` includes it's own formatted call, hence `call.` should be `FALSE` (default). If you insert your own `msg`, then you may want `call. = TRUE`.

Value

`invisible(arg)` or condition determined by argument `signal`.

See Also

Other check_relation: `check_all_arg_in_x()`, `check_all_x_in_arg()`, `check_any_arg_in_x()`, `check_any_x_in_arg()`, `check_finite()`, `check_length_equals()`, `check_lt()`, `check_monotonic_decreasing()`, `check_names_in_arg()`, `check_range()`, `check_scalar_arg_in_x()`, `check_strict_decreasing()`, `check_unique()`

Examples

```
#--------------------------------------------------------------------------
# check_equals() examples
#--------------------------------------------------------------------------
library(bkcheck)
```

```
f <- function(x) {
  vec <- 1:2
  x |>
    check_equals(vec)
}

x1 <- 1:2
x2 <- 2:3

f(x1)
try(f(x2))
```

---

check_factor *Check factor vector*

Description

Check if arg is a factor vector.

Usage

```
check_factor(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_nur check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#--------------------------------------------------------------------------
# check_factor() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_factor()
}

f(factor("a"))

try(f(TRUE))
try(f(1L))
try(f(1))
try(f("a"))
```

`check_file_exists` *Check path existence*

Description

Check if `arg` is a path to an existing file or directory.

Usage

```
check_file_exists(arg, signal = "error", msg = NULL, call. = FALSE)

check_dir_exists(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(string) A file system path.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Details

check_file_exists() requires arg to point to an existing file. An existing directory fails, matching the split base R draws between base::file.exists(), which is TRUE for both files and directories, and base::dir.exists(), which is TRUE for directories only. check_dir_exists() requires arg to point to an existing directory.

A missing value (NA) fails, since it names no path. A vector of length other than one fails, since each check validates a single path.

Value

invisible(arg) or condition determined by argument signal.

Examples

```
#--------------------------------------------------------------------------
# check path existence examples
#--------------------------------------------------------------------------
library(bkcheck)

f_file <- function(x) {
  x |>
    check_file_exists()
}

f_dir <- function(x) {
  x |>
    check_dir_exists()
}

#--------------------------------------------------------------------------
# check_file_exists()
#--------------------------------------------------------------------------
```

```
file <- tempfile()
invisible(file.create(file))

f_file(x = file)

try(f_file(x = tempfile()))
try(f_file(tempdir()))

#--------------------------------------------------------------------------
# check_dir_exists()
#--------------------------------------------------------------------------
f_dir(x = tempdir())

try(f_dir(x = tempfile()))
```

`check_finite` *Check finite*

Description

Check if `arg` is all finite.

Usage

```
check_finite(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

`arg`	(object) The argument to check.
`signal`	(string: `c("error", "warning", "message")`) Must be one of `"error"`, `"warning"`, or `"message"`. `"error"` (default) calls `base::stop()` to signal an error message. `"warning"` calls `base::warning()` to signal a warning message. `"message"` calls `base::message()` to signal a message.
`msg`	(string or `NULL`: `NULL`) A string that replaces the default message. Set as `NULL` to use the default message.
`call.`	(Scalar logical: `c(FALSE, TRUE)`) Passed to `stop()` or `warning()`. The default `msg` includes it's own formatted call, hence `call.` should be `FALSE` (default). If you insert your own `msg`, then you may want `call. = TRUE`.

Value

`invisible(arg)` or condition determined by argument `signal`.

See Also

Other check_relation: check_all_arg_in_x(), check_all_x_in_arg(), check_any_arg_in_x(), check_any_x_in_arg(), check_equals(), check_length_equals(), check_lt(), check_monotonic_decreasing(), check_names_in_arg(), check_range(), check_scalar_arg_in_x(), check_strict_decreasing(), check_unique()

Examples

```
#--------------------------------------------------------------------------
# check_finite() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_finite()
}

x1 <- 1000
x2 <- Inf

f(x1)
try(f(x2))
```

---

check_flag *Check flag*

Description

Check if arg is a flag, a scalar logical that is TRUE or FALSE.

Usage

```
check_flag(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.

`call.` (Scalar logical: `c(FALSE, TRUE)`)
Passed to `stop()` or `warning()`. The default `msg` includes it's own formatted call, hence `call.` should be `FALSE` (default). If you insert your own `msg`, then you may want `call. = TRUE`.

Details

Unlike `check_scalar_logical()`, a flag rejects `NA`. This matches the common case of a boolean toggle argument that must be decidably on or off.

Value

`invisible(arg)` or condition determined by argument `signal`.

See Also

`check_scalar_logical()`

Other check_type: `check_atomic()`, `check_character()`, `check_data_frame()`, `check_data_frame_or_matrix()`, `check_date()`, `check_date_or_datetime()`, `check_datetime()`, `check_double()`, `check_factor()`, `check_formula()`, `check_function()`, `check_function_or_numeric()`, `check_inherits()`, `check_integer()`, `check_list()`, `check_logical()`, `check_matrix()`, `check_named()`, `check_nonempty_string()`, `check_nonmissing()`, `check_numeric()`, `check_numeric_or_logical()`, `check_probability()`, `check_real()`, `check_scalar_character()`, `check_scalar_date()`, `check_scalar_datetime()`, `check_scalar_double()`, `check_scalar_factor()`, `check_scalar_integer()`, `check_scalar_logical()`, `check_scalar_numeric()`, `check_scalar_probability()`, `check_scalar_whole_number()`, `check_scalar_whole_nur` `check_string()`, `check_table()`, `check_valid()`, `check_valid_numeric()`, `check_whole_number()`, `check_whole_numeric()`

Examples

```
#------------------------------------------------------------------------------
# check_flag() examples
#------------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_flag()
}

f(TRUE)
f(FALSE)

try(f(NA))
try(f(1L))
try(f(c(TRUE, FALSE)))
try(f("TRUE"))
```

check_formula *Check formula*

Description

Check if arg is a formula.

Usage

```
check_formula(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_nur check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#--------------------------------------------------------------------------
# check_formula() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_formula()
}

f(y ~ x)

try(f(TRUE))
try(f(1L))
try(f(1))
try(f("a"))
```

`check_function` *Check function*

Description

Check if `arg` is a function.

Usage

```
check_function(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

`arg`	(object) The argument to check.
`signal`	(string: `c("error", "warning", "message")`) Must be one of `"error"`, `"warning"`, or `"message"`. `"error"` (default) calls `base::stop()` to signal an error message. `"warning"` calls `base::warning()` to signal a warning message. `"message"` calls `base::message()` to signal a message.
`msg`	(string or `NULL`: `NULL`) A string that replaces the default message. Set as `NULL` to use the default message.
`call.`	(Scalar logical: `c(FALSE, TRUE)`) Passed to `stop()` or `warning()`. The default `msg` includes it's own formatted call, hence `call.` should be `FALSE` (default). If you insert your own `msg`, then you may want `call. = TRUE`.

Value

`invisible(arg)` or condition determined by argument `signal`.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_nu
check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#------------------------------------------------------------------------
# check_function() examples
#------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_function()
}

f(function(x) x)

try(f(TRUE))
try(f(1L))
try(f(1))
try(f("a"))
```

`check_function_or_numeric` *Check function or numeric*

Description

Check if `arg` is a function or numeric.

Usage

```
check_function_or_numeric(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_nu
check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#--------------------------------------------------------------------------
# check_function_or_numeric() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_function_or_numeric()
}

f(function(x) x)
f(1L)
f(1)
```

```
try(f(TRUE))
try(f("a"))
```

---

check_inherits *Check class inheritance*

Description

Check if arg inherits from any of the classes in class.

Usage

```
check_inherits(arg, class, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
class	(Character) A character vector of class names. arg passes if it inherits from at least one of these classes, following base::inherits() semantics.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(),

check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_nu
check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#------------------------------------------------------------------------------
# check_inherits() examples
#------------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_inherits("data.frame")
}

f(data.frame(a = 1))
try(f(1:2))

# Pass if `arg` inherits from any one of several classes
g <- function(x) {
  x |>
    check_inherits(c("Date", "POSIXct"))
}

g(Sys.Date())
g(Sys.time())
try(g(1:2))
```

---

check_integer *Check integer vector*

Description

Check if arg is a integer vector.

Usage

```
check_integer(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning()

to signal a warning message. "message" calls base::message() to signal a message.

msg (string or NULL: NULL)
A string that replaces the default message. Set as NULL to use the default message.

call. (Scalar logical: c(FALSE, TRUE))
Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_nur check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#--------------------------------------------------------------------------
# check_integer() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_integer()
}

logical <- TRUE
integer <- 1:2
numeric <- c(1, 1.5)
character <- letters[1:4]
factor <- factor(character)

f(integer)

try(f(logical))
try(f(numeric))
try(f(character))
```

```
try(f(factor))
```

`check_length_equals` *Check length of argument*

Description

Check if `length(arg)` is ==, <=, or >= `length`.

Usage

```
check_length_equals(arg, length, signal = "error", msg = NULL, call. = FALSE)

check_length_leq(arg, length, signal = "error", msg = NULL, call. = FALSE)

check_length_geq(arg, length, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

`arg`	(object) The argument to check.
`length`	(Scalar integer) An integer for the length of `arg`.
`signal`	(string: `c("error", "warning", "message")`) Must be one of `"error"`, `"warning"`, or `"message"`. `"error"` (default) calls `base::stop()` to signal an error message. `"warning"` calls `base::warning()` to signal a warning message. `"message"` calls `base::message()` to signal a message.
`msg`	(string or `NULL`: `NULL`) A string that replaces the default message. Set as `NULL` to use the default message.
`call.`	(Scalar logical: `c(FALSE, TRUE)`) Passed to `stop()` or `warning()`. The default `msg` includes it's own formatted call, hence `call.` should be `FALSE` (default). If you insert your own `msg`, then you may want `call. = TRUE`.

Value

`invisible(arg)` or condition determined by argument `signal`.

See Also

Other check_relation: `check_all_arg_in_x()`, `check_all_x_in_arg()`, `check_any_arg_in_x()`, `check_any_x_in_arg()`, `check_equals()`, `check_finite()`, `check_lt()`, `check_monotonic_decreasing()`, `check_names_in_arg()`, `check_range()`, `check_scalar_arg_in_x()`, `check_strict_decreasing()`, `check_unique()`

Examples

```
#--------------------------------------------------------------------------
# check_length_*() examples
#--------------------------------------------------------------------------
library(bkcheck)

#--------------------------------------------------------------------------
# Length equals
#--------------------------------------------------------------------------
f <- function(x) {
  x |>
    check_length_equals(2L)
}

f(1:2)
try(f(1))

#--------------------------------------------------------------------------
# Length less than or equal to
#--------------------------------------------------------------------------
f <- function(x) {
  x |>
    check_length_leq(2L)
}

f(1:2)
try(f(1:3))

#--------------------------------------------------------------------------
# Length greater than or equal to
#--------------------------------------------------------------------------
f <- function(x) {
  x |>
    check_length_geq(3L)
}

f(1:3)
try(f(1:2))
```

`check_list` *Check list*

Description

Check if `arg` is a list.

Usage

```
check_list(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_nur check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#--------------------------------------------------------------------------
# check_list() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_list()
}

f(list(1))

try(f(data.frame(1)))
try(f(TRUE))
```

```
try(f(1L))
try(f(1))
try(f("a"))
try(f(factor("a")))
```

---

check_logical *Check logical vector*

Description

Check if arg is a logical vector.

Usage

```
check_logical(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(),

check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_nu
check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(),
check_whole_numeric()

Examples

```
#------------------------------------------------------------------------------
# check_logical() examples
#------------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_logical()
}

logical <- TRUE
integer <- 1:2
numeric <- c(1, 1.5)
character <- letters[1:4]
factor <- factor(character)

f(logical)

try(f(integer))
try(f(numeric))
try(f(character))
try(f(factor))
```

---

check_lt *Check* arg *is bounded*

Description

Check if arg is <, <=, >, or >= bound.

Usage

```
check_lt(arg, bound, signal = "error", msg = NULL, call. = FALSE)

check_leq(arg, bound, signal = "error", msg = NULL, call. = FALSE)

check_gt(arg, bound, signal = "error", msg = NULL, call. = FALSE)

check_geq(arg, bound, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(Scalar numeric) The argument to check.
bound	(Scalar numeric) The bound for arg.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_relation: check_all_arg_in_x(), check_all_x_in_arg(), check_any_arg_in_x(), check_any_x_in_arg(), check_equals(), check_finite(), check_length_equals(), check_monotonic_decreasing(check_names_in_arg(), check_range(), check_scalar_arg_in_x(), check_strict_decreasing(), check_unique()

Examples

```
#----------------------------------------------------------------------------
# check bounded examples
#----------------------------------------------------------------------------
library(bkcheck)

#----------------------------------------------------------------------------
# check_lt()
#----------------------------------------------------------------------------
f <- function(x) {
  x |>
    check_numeric() |>
    check_lt(2)
}

f(0:1)
try(f(0:2))

#----------------------------------------------------------------------------
```

```
# check_leq()
#--------------------------------------------------------------------------------
f <- function(x) {
  x |>
    check_numeric() |>
    check_leq(1)
}

f(0:1)
try(f(0:2))

#--------------------------------------------------------------------------------
# check_gt()
#--------------------------------------------------------------------------------
f <- function(x) {
  x |>
    check_numeric() |>
    check_gt(0)
}

f(1:2)
try(f(0:2))

#--------------------------------------------------------------------------------
# check_geq()
#--------------------------------------------------------------------------------
f <- function(x) {
  x |>
    check_numeric() |>
    check_geq(2)
}

f(2:3)
try(f(1:3))
```

`check_matrix` *Check matrix*

Description

Check if `arg` is a matrix.

Usage

```
check_matrix(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

`arg`	(object) The argument to check.

signal (string: c("error", "warning", "message"))
Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.

msg (string or NULL: NULL)
A string that replaces the default message. Set as NULL to use the default message.

call. (Scalar logical: c(FALSE, TRUE))
Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_nu check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#--------------------------------------------------------------------------
# check_matrix() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_matrix()
}

mat <- matrix(1:2)
integer <- 1:2
numeric <- c(1, 1.5)
character <- letters[1:4]
factor <- factor(character)

f(mat)
```

```
try(f(data.frame(1)))
try(f(integer))
try(f(numeric))
try(f(character))
try(f(factor))
```

`check_monotonic_decreasing` *Check monotonic vector*

Description

Check if `arg` is a monotonic (increasing or decreasing) numeric vector.

Usage

```
check_monotonic_decreasing(
  arg,
  na.rm = FALSE,
  signal = "error",
  msg = NULL,
  call. = FALSE
)

check_monotonic_increasing(
  arg,
  na.rm = FALSE,
  signal = "error",
  msg = NULL,
  call. = FALSE
)
```

Arguments

`arg`	(object) The argument to check.
`na.rm`	(scalar logical: `c(FALSE, TRUE)`) Whether or not to remove missing values before checking.
`signal`	(string: `c("error", "warning", "message")`) Must be one of `"error"`, `"warning"`, or `"message"`. `"error"` (default) calls `base::stop()` to signal an error message. `"warning"` calls `base::warning()` to signal a warning message. `"message"` calls `base::message()` to signal a message.
`msg`	(string or `NULL`: `NULL`) A string that replaces the default message. Set as `NULL` to use the default message.

`call.` (Scalar logical: `c(FALSE, TRUE)`)
Passed to `stop()` or `warning()`. The default `msg` includes it's own formatted call, hence `call.` should be `FALSE` (default). If you insert your own `msg`, then you may want `call. = TRUE`.

Value

`invisible(arg)` or condition determined by argument `signal`.

See Also

Other check_relation: `check_all_arg_in_x()`, `check_all_x_in_arg()`, `check_any_arg_in_x()`, `check_any_x_in_arg()`, `check_equals()`, `check_finite()`, `check_length_equals()`, `check_lt()`, `check_names_in_arg()`, `check_range()`, `check_scalar_arg_in_x()`, `check_strict_decreasing()`, `check_unique()`

Examples

```
#--------------------------------------------------------------------------
# check_monotonic_*() examples
#--------------------------------------------------------------------------
library(bkcheck)

#--------------------------------------------------------------------------
# check_monotonic_decreasing()
#--------------------------------------------------------------------------
f <- function(x) {
  x |>
    check_numeric() |>
    check_monotonic_decreasing()
}

f(10:1)
f(c(3, 3, 2, 1))
try(f(1:2))

#--------------------------------------------------------------------------
# check_monotonic_increasing()
#--------------------------------------------------------------------------
f <- function(x) {
  x |>
    check_numeric() |>
    check_monotonic_increasing()
}

f(1:10)
f(c(1, 1, 2, 2, 3))
try(f(2:1))
```

check_named *Check named*

Description

Check if `arg` has names.

Usage

```
check_named(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

`arg`	(object) The argument to check.
`signal`	(string: `c("error", "warning", "message")`) Must be one of `"error"`, `"warning"`, or `"message"`. `"error"` (default) calls `base::stop()` to signal an error message. `"warning"` calls `base::warning()` to signal a warning message. `"message"` calls `base::message()` to signal a message.
`msg`	(string or `NULL`: `NULL`) A string that replaces the default message. Set as `NULL` to use the default message.
`call.`	(Scalar logical: `c(FALSE, TRUE)`) Passed to `stop()` or `warning()`. The default `msg` includes it's own formatted call, hence `call.` should be `FALSE` (default). If you insert your own `msg`, then you may want `call. = TRUE`.

Value

`invisible(arg)` or condition determined by argument `signal`.

See Also

Other check_type: `check_atomic()`, `check_character()`, `check_data_frame()`, `check_data_frame_or_matrix()`, `check_date()`, `check_date_or_datetime()`, `check_datetime()`, `check_double()`, `check_factor()`, `check_flag()`, `check_formula()`, `check_function()`, `check_function_or_numeric()`, `check_inherits()`, `check_integer()`, `check_list()`, `check_logical()`, `check_matrix()`, `check_nonempty_string()`, `check_nonmissing()`, `check_numeric()`, `check_numeric_or_logical()`, `check_probability()`, `check_real()`, `check_scalar_character()`, `check_scalar_date()`, `check_scalar_datetime()`, `check_scalar_double()`, `check_scalar_factor()`, `check_scalar_integer()`, `check_scalar_logical()`, `check_scalar_numeric()`, `check_scalar_probability()`, `check_scalar_whole_number()`, `check_scalar_whole_nur` `check_string()`, `check_table()`, `check_valid()`, `check_valid_numeric()`, `check_whole_number()`, `check_whole_numeric()`

Examples

```
#------------------------------------------------------------------------------
# check_named() examples
#------------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_named()
}

f(c(a = 1, b = 2))
f(data.frame(a = 1))
f(list(a = 1, b = 2))

try(f(1:2))
try(f(c(a = 1, 2)))
try(f(list(a = 1, 2)))
```

`check_names_in_arg` *Check names*

Description

Check if `arg` has specified names.

Usage

```
check_names_in_arg(arg, names, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

`arg`	(object) The argument to check.
`names`	(character) The names that should match `names(arg)`.
`signal`	(string: `c("error", "warning", "message")`) Must be one of `"error"`, `"warning"`, or `"message"`. `"error"` (default) calls `base::stop()` to signal an error message. `"warning"` calls `base::warning()` to signal a warning message. `"message"` calls `base::message()` to signal a message.
`msg`	(string or `NULL`: `NULL`) A string that replaces the default message. Set as `NULL` to use the default message.
`call.`	(Scalar logical: `c(FALSE, TRUE)`) Passed to `stop()` or `warning()`. The default `msg` includes it's own formatted call, hence `call.` should be `FALSE` (default). If you insert your own `msg`, then you may want `call. = TRUE`.

Value

`invisible(arg)` or condition determined by argument `signal`.

See Also

Other check_relation: `check_all_arg_in_x()`, `check_all_x_in_arg()`, `check_any_arg_in_x()`, `check_any_x_in_arg()`, `check_equals()`, `check_finite()`, `check_length_equals()`, `check_lt()`, `check_monotonic_decreasing()`, `check_range()`, `check_scalar_arg_in_x()`, `check_strict_decreasing()`, `check_unique()`

Examples

```
#------------------------------------------------------------------------
# check_names_in_arg() examples
#------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_names_in_arg(c("a", "b"))
}

f(c(a = 1, b = 2))
f(data.frame(a = 1, b = 2))
f(list(a = 1, b = 2))

try(f(1:2))
try(f(c(a = 1, 2)))
try(f(list(a = 1, 2)))
```

---

check_ncol_equals *Check number of columns*

Description

Check if `arg` is a 2-dimensional object with number of columns equal to or greater/less than or equal to `ncol`.

Usage

```
check_ncol_equals(arg, ncol, signal = "error", msg = NULL, call. = FALSE)

check_ncol_leq(arg, ncol, signal = "error", msg = NULL, call. = FALSE)

check_ncol_geq(arg, ncol, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
ncol	(scalar integer) The number of columns to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_dims: check_nrow_equals()

Examples

```
#--------------------------------------------------------------------------
# check_ncol*() examples
#--------------------------------------------------------------------------
library(bkcheck)

#--------------------------------------------------------------------------
# check_ncol_equals()
#--------------------------------------------------------------------------
f <- function(x) {
  x |>
    check_ncol_equals(1L)
}
f(data.frame(1))
try(f(data.frame(1:2, 1:2)))

#--------------------------------------------------------------------------
# check_ncol_leq()
#--------------------------------------------------------------------------
f <- function(x) {
  x |>
    check_ncol_leq(2L)
```

```
}
f(matrix(1))
try(f(data.frame(1, 2, 3)))

#--------------------------------------------------------------------------
# check_ncol_geq()
#--------------------------------------------------------------------------
f <- function(x) {
  x |>
    check_data_frame() |>
    check_ncol_geq(1L)
}
f(data.frame(1))
try(f(data.frame()))
```

`check_nonempty_string` *Check non-empty string*

Description

Check if `arg` is a non-empty string, a scalar character vector that is not `""`.

Usage

```
check_nonempty_string(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

`arg`	(object) The argument to check.
`signal`	(string: `c("error", "warning", "message")`) Must be one of `"error"`, `"warning"`, or `"message"`. `"error"` (default) calls `base::stop()` to signal an error message. `"warning"` calls `base::warning()` to signal a warning message. `"message"` calls `base::message()` to signal a message.
`msg`	(string or `NULL`: `NULL`) A string that replaces the default message. Set as `NULL` to use the default message.
`call.`	(Scalar logical: `c(FALSE, TRUE)`) Passed to `stop()` or `warning()`. The default `msg` includes it's own formatted call, hence `call.` should be `FALSE` (default). If you insert your own `msg`, then you may want `call. = TRUE`.

Details

Unlike `check_string()`, a non-empty string rejects `""`. This matches the common case of a name, path, or label argument that must carry content. A missing value (`NA`) is accepted, matching `check_string()`. Chain `check_nonmissing()` to also reject `NA`.

Value

`invisible(arg)` or condition determined by argument `signal`.

See Also

`check_string()`

Other check_type: `check_atomic()`, `check_character()`, `check_data_frame()`, `check_data_frame_or_matrix()`, `check_date()`, `check_date_or_datetime()`, `check_datetime()`, `check_double()`, `check_factor()`, `check_flag()`, `check_formula()`, `check_function()`, `check_function_or_numeric()`, `check_inherits()`, `check_integer()`, `check_list()`, `check_logical()`, `check_matrix()`, `check_named()`, `check_nonmissing()`, `check_numeric()`, `check_numeric_or_logical()`, `check_probability()`, `check_real()`, `check_scalar_character()`, `check_scalar_date()`, `check_scalar_datetime()`, `check_scalar_double()`, `check_scalar_factor()`, `check_scalar_integer()`, `check_scalar_logical()`, `check_scalar_numeric()`, `check_scalar_probability()`, `check_scalar_whole_number()`, `check_scalar_whole_numeric()`, `check_string()`, `check_table()`, `check_valid()`, `check_valid_numeric()`, `check_whole_number()`, `check_whole_numeric()`

Examples

```
#--------------------------------------------------------------------------
# check_nonempty_string() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_nonempty_string()
}

f("a")
f(NA_character_)

try(f(""))
try(f(1L))
try(f(c("a", "b")))
try(f(factor("a")))
```

---

check_nonmissing *Check nonmissing*

Description

Check if `arg` contains no missing values (`NA` or `NaN`).

Usage

```
check_nonmissing(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_numeric(), check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#--------------------------------------------------------------------------
# check_nonmissing() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_nonmissing()
}

f(c(1, 2))
f(numeric(0))
f(NULL)

try(f(c(1, NA)))
```

```
try(f(c(1, NaN)))
```

`check_nrow_equals` *Check number of rows*

Description

Check if `arg` is a 2-dimensional object with number of rows equal to or greater/less than or equal to `nrow`.

Usage

```
check_nrow_equals(arg, nrow, signal = "error", msg = NULL, call. = FALSE)

check_nrow_leq(arg, nrow, signal = "error", msg = NULL, call. = FALSE)

check_nrow_geq(arg, nrow, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

`arg`	(object) The argument to check.
`nrow`	(scalar integer) The number of rows to check.
`signal`	(string: `c("error", "warning", "message")`) Must be one of `"error"`, `"warning"`, or `"message"`. `"error"` (default) calls `base::stop()` to signal an error message. `"warning"` calls `base::warning()` to signal a warning message. `"message"` calls `base::message()` to signal a message.
`msg`	(string or `NULL`: `NULL`) A string that replaces the default message. Set as `NULL` to use the default message.
`call.`	(Scalar logical: `c(FALSE, TRUE)`) Passed to `stop()` or `warning()`. The default `msg` includes it's own formatted call, hence `call.` should be `FALSE` (default). If you insert your own `msg`, then you may want `call. = TRUE`.

Value

`invisible(arg)` or condition determined by argument `signal`.

See Also

Other check_dims: `check_ncol_equals()`

Examples

```
#--------------------------------------------------------------------------
# check_nrow*() examples
#--------------------------------------------------------------------------
library(bkcheck)

#--------------------------------------------------------------------------
# check_nrow_equals()
#--------------------------------------------------------------------------
f <- function(x) {
  x |>
    check_nrow_equals(1L)
}
f(data.frame(1))
try(f(data.frame(1:2)))

#--------------------------------------------------------------------------
# check_nrow_leq()
#--------------------------------------------------------------------------
f <- function(x) {
  x |>
    check_nrow_leq(2L)
}
f(matrix(1))
try(f(data.frame(1:3, 2:4, 3:5)))

#--------------------------------------------------------------------------
# check_nrow_geq()
#--------------------------------------------------------------------------
f <- function(x) {
  x |>
    check_data_frame() |>
    check_nrow_geq(1L)
}
f(data.frame(1))
try(f(data.frame()))
```

`check_numeric` *Check numeric vector*

Description

Check if `arg` is a numeric vector.

Usage

```
check_numeric(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_numeric(), check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#--------------------------------------------------------------------------
# check_numeric() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_numeric()
}

logical <- TRUE
integer <- 1:2
numeric <- c(1, 1.5)
character <- letters[1:4]
```

```
factor <- factor(character)

f(integer)
f(numeric)

try(f(logical))
try(f(character))
try(f(factor))
```

`check_numeric_or_logical` *Check numeric or logical*

Description

Check if `arg` is numeric or logical.

Usage

```
check_numeric_or_logical(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

`arg`	(object) The argument to check.
`signal`	(string: `c("error", "warning", "message")`) Must be one of `"error"`, `"warning"`, or `"message"`. `"error"` (default) calls `base::stop()` to signal an error message. `"warning"` calls `base::warning()` to signal a warning message. `"message"` calls `base::message()` to signal a message.
`msg`	(string or `NULL`: `NULL`) A string that replaces the default message. Set as `NULL` to use the default message.
`call.`	(Scalar logical: `c(FALSE, TRUE)`) Passed to `stop()` or `warning()`. The default `msg` includes it's own formatted call, hence `call.` should be `FALSE` (default). If you insert your own `msg`, then you may want `call. = TRUE`.

Value

`invisible(arg)` or condition determined by argument `signal`.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_numeric(), check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#--------------------------------------------------------------------------
# check_numeric_or_logical() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_numeric_or_logical()
}

bool <- c(TRUE, FALSE)
integer <- 1:2
numeric <- c(1, 1.5)
character <- letters[1:4]
factor <- factor(character)

f(bool)
f(integer)
f(numeric)

try(f(character))
try(f(factor))
```

---

check_probability *Check vector of probabilities*

Description

Check if arg is a vector of probabilities (numeric vector bounded by [0, 1]).

Usage

```
check_probability(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_numeric(), check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#--------------------------------------------------------------------------
# check_probability() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_probability()
}

logical <- TRUE
integer <- 1L
numeric <- c(0.5, 1)
numeric2 <- c(0.5, 1.5)
character <- letters[1:4]
```

```
factor <- factor(character)

f(integer)
f(numeric)

try(f(logical))
try(f(numeric2))
try(f(character))
try(f(factor))
```

---

check_range *Check* arg *is within a range*

Description

Check if all elements of arg lie within the interval bounded by lower and upper.

Usage

```
check_range(
  arg,
  lower,
  upper,
  lower_inclusive = TRUE,
  upper_inclusive = TRUE,
  signal = "error",
  msg = NULL,
  call. = FALSE
)
```

Arguments

arg	(Numeric) The argument to check.
lower	(Scalar numeric) The lower bound of the interval.
upper	(Scalar numeric) The upper bound of the interval. Must be greater than or equal to lower.
lower_inclusive	(Scalar logical: TRUE) Whether the lower bound is inclusive. TRUE makes the bound closed (arg >= lower). FALSE makes the bound open (arg > lower).
upper_inclusive	(Scalar logical: TRUE) Whether the upper bound is inclusive. TRUE makes the bound closed (arg <= upper). FALSE makes the bound open (arg < upper).

`signal` (string: `c("error", "warning", "message")`)
Must be one of `"error"`, `"warning"`, or `"message"`. `"error"` (default) calls `base::stop()` to signal an error message. `"warning"` calls `base::warning()` to signal a warning message. `"message"` calls `base::message()` to signal a message.

`msg` (string or `NULL`: `NULL`)
A string that replaces the default message. Set as `NULL` to use the default message.

`call.` (Scalar logical: `c(FALSE, TRUE)`)
Passed to `stop()` or `warning()`. The default `msg` includes it's own formatted call, hence `call.` should be `FALSE` (default). If you insert your own `msg`, then you may want `call. = TRUE`.

Value

`invisible(arg)` or condition determined by argument `signal`.

See Also

Other check_relation: `check_all_arg_in_x()`, `check_all_x_in_arg()`, `check_any_arg_in_x()`, `check_any_x_in_arg()`, `check_equals()`, `check_finite()`, `check_length_equals()`, `check_lt()`, `check_monotonic_decreasing()`, `check_names_in_arg()`, `check_scalar_arg_in_x()`, `check_strict_decreasing()`, `check_unique()`

Examples

```
#--------------------------------------------------------------------------
# check_range() examples
#--------------------------------------------------------------------------
library(bkcheck)

#--------------------------------------------------------------------------
# Closed interval [0, 1]
#--------------------------------------------------------------------------
f <- function(x) {
  x |>
    check_numeric() |>
    check_range(0, 1)
}

f(c(0, 0.5, 1))
try(f(c(0, 0.5, 1.5)))

#--------------------------------------------------------------------------
# Open interval (0, 1)
#--------------------------------------------------------------------------
g <- function(x) {
  x |>
    check_numeric() |>
    check_range(0, 1, lower_inclusive = FALSE, upper_inclusive = FALSE)
}
```

```
g(c(0.25, 0.5, 0.75))
try(g(c(0, 0.5, 1)))
```

`check_real` *Check real number vector*

Description

Check if `arg` is a numeric vector of real numbers. Every value must be finite and non-missing and the length must be greater than zero.

Usage

```
check_real(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

`arg`	(object) The argument to check.
`signal`	(string: `c("error", "warning", "message")`) Must be one of `"error"`, `"warning"`, or `"message"`. `"error"` (default) calls `base::stop()` to signal an error message. `"warning"` calls `base::warning()` to signal a warning message. `"message"` calls `base::message()` to signal a message.
`msg`	(string or `NULL`: `NULL`) A string that replaces the default message. Set as `NULL` to use the default message.
`call.`	(Scalar logical: `c(FALSE, TRUE)`) Passed to `stop()` or `warning()`. The default `msg` includes it's own formatted call, hence `call.` should be `FALSE` (default). If you insert your own `msg`, then you may want `call. = TRUE`.

Details

This is a shortcut for the chain `check_numeric()`, `check_valid()`, and `check_finite()`. Use `check_valid_numeric()` to allow infinite values.

Value

`invisible(arg)` or condition determined by argument `signal`.

See Also

check_valid_numeric()

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_numeric(), check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#--------------------------------------------------------------------------
# check_real() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_real()
}

integer <- 1:4
numeric <- c(1, 1.5)
character <- letters[1:4]

f(integer)
f(numeric)

try(f(character))
try(f(numeric(0)))
try(f(c(1, NA)))
try(f(c(1, Inf)))
```

---

check_scalar_arg_in_x *Check scalar* arg *in* x

Description

Check if scalar arg %in% x.

Usage

```
check_scalar_arg_in_x(arg, x, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
x	(Atomic vector) Scalar arg must match one value in x.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_relation: check_all_arg_in_x(), check_all_x_in_arg(), check_any_arg_in_x(), check_any_x_in_arg(), check_equals(), check_finite(), check_length_equals(), check_lt(), check_monotonic_decreasing(), check_names_in_arg(), check_range(), check_strict_decreasing(), check_unique()

Examples

```
#--------------------------------------------------------------------------
# check_scalar_arg_in_x() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  chr <- c("a", "b", "c")
  x |>
    check_scalar_arg_in_x(chr)
}

f("a")
try(f("d"))
```

check_scalar_character *Check scalar character*

Description

Check if arg is a scalar character vector.

Usage

```
check_scalar_character(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_numeric(), check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#----------------------------------------------------------------------------
# check_scalar_character() examples
#----------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_scalar_character()
}

f("a")

try(f(TRUE))
try(f(1L))
try(f(1))
try(f(c("a", "b")))
try(f(factor("a")))
```

`check_scalar_date` *Check scalar date*

Description

Check if `arg` is a scalar date vector.

Usage

```
check_scalar_date(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

`arg`	(object) The argument to check.
`signal`	(string: `c("error", "warning", "message")`) Must be one of `"error"`, `"warning"`, or `"message"`. `"error"` (default) calls `base::stop()` to signal an error message. `"warning"` calls `base::warning()` to signal a warning message. `"message"` calls `base::message()` to signal a message.
`msg`	(string or `NULL`: `NULL`) A string that replaces the default message. Set as `NULL` to use the default message.
`call.`	(Scalar logical: `c(FALSE, TRUE)`) Passed to `stop()` or `warning()`. The default `msg` includes it's own formatted call, hence `call.` should be `FALSE` (default). If you insert your own `msg`, then you may want `call. = TRUE`.

Value

`invisible(arg)` or condition determined by argument `signal`.

See Also

Other check_type: `check_atomic()`, `check_character()`, `check_data_frame()`, `check_data_frame_or_matrix()`, `check_date()`, `check_date_or_datetime()`, `check_datetime()`, `check_double()`, `check_factor()`, `check_flag()`, `check_formula()`, `check_function()`, `check_function_or_numeric()`, `check_inherits()`, `check_integer()`, `check_list()`, `check_logical()`, `check_matrix()`, `check_named()`, `check_nonempty_string()`, `check_nonmissing()`, `check_numeric()`, `check_numeric_or_logical()`, `check_probability()`, `check_real()`, `check_scalar_character()`, `check_scalar_datetime()`, `check_scalar_double()`, `check_scalar_factor()`, `check_scalar_integer()`, `check_scalar_logical()`, `check_scalar_numeric()`, `check_scalar_probability()`, `check_scalar_whole_number()`, `check_scalar_whole_numeric()`, `check_string()`, `check_table()`, `check_valid()`, `check_valid_numeric()`, `check_whole_number()`, `check_whole_numeric()`

Examples

```
#-------------------------------------------------------------------------
# check_scalar_date() examples
#-------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_scalar_date()
}

f(Sys.Date())

try(f(c(Sys.Date(), Sys.Date() - 1)))
try(f(TRUE))
try(f(Sys.time()))
try(f(1L))
try(f(1))
try(f(factor("a")))
try(f("a"))
```

`check_scalar_datetime` *Check scalar datetime*

Description

Check if `arg` is a scalar datetime vector.

Usage

```
check_scalar_datetime(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_numeric(), check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#--------------------------------------------------------------------------
# check_scalar_datetime() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_scalar_datetime()
}

try(f(Sys.time()))

try(f(c(Sys.time(), Sys.time() - 1)))
try(f(TRUE))
```

```
try(f(Sys.Date()))
try(f(1L))
try(f(1))
try(f(factor("a")))
try(f("a"))
```

---

check_scalar_double *Check scalar double-precision*

Description

Check if arg is a scalar double-precision vector.

Usage

```
check_scalar_double(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(),

check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_numeric(), check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#------------------------------------------------------------------------------
# check_scalar_double() examples
#------------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_scalar_double()
}

logical <- TRUE
integer <- 1L
numeric <- 1
character <- letters[1]
factor <- factor(character)

f(numeric)

try(f(logical))
try(f(integer))
try(f(character))
try(f(factor))
```

---

check_scalar_factor *Check scalar factor*

Description

Check if arg is a scalar factor vector.

Usage

```
check_scalar_factor(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning()

to signal a warning message. "message" calls base::message() to signal a message.

msg (string or NULL: NULL)
A string that replaces the default message. Set as NULL to use the default message.

call. (Scalar logical: c(FALSE, TRUE))
Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_numeric(), check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#--------------------------------------------------------------------------
# check_scalar_factor() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_scalar_factor()
}

f(factor("a"))

try(f(TRUE))
try(f(1L))
try(f(1))
try(f("a"))
try(f(factor(c("a", "b"))))
```

check_scalar_integer *Check scalar integer*

Description

Check if arg is a scalar integer vector.

Usage

```
check_scalar_integer(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_numeric(), check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#--------------------------------------------------------------------------
# check_scalar_integer() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_scalar_integer()
}

logical <- TRUE
integer <- 1L
numeric <- 1
character <- letters[1]
factor <- factor(character)

f(integer)

try(f(logical))
try(f(numeric))
try(f(character))
try(f(factor))
```

`check_scalar_logical` *Check scalar logical*

Description

Check if `arg` is a scalar logical vector.

Usage

```
check_scalar_logical(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

`arg`	(object) The argument to check.
`signal`	(string: `c("error", "warning", "message")`) Must be one of `"error"`, `"warning"`, or `"message"`. `"error"` (default) calls `base::stop()` to signal an error message. `"warning"` calls `base::warning()` to signal a warning message. `"message"` calls `base::message()` to signal a message.
`msg`	(string or `NULL`: `NULL`) A string that replaces the default message. Set as `NULL` to use the default message.

`call.` (Scalar logical: `c(FALSE, TRUE)`)
Passed to `stop()` or `warning()`. The default `msg` includes it's own formatted call, hence `call.` should be `FALSE` (default). If you insert your own `msg`, then you may want `call. = TRUE`.

Value

`invisible(arg)` or condition determined by argument `signal`.

See Also

Other check_type: `check_atomic()`, `check_character()`, `check_data_frame()`, `check_data_frame_or_matrix()`, `check_date()`, `check_date_or_datetime()`, `check_datetime()`, `check_double()`, `check_factor()`, `check_flag()`, `check_formula()`, `check_function()`, `check_function_or_numeric()`, `check_inherits()`, `check_integer()`, `check_list()`, `check_logical()`, `check_matrix()`, `check_named()`, `check_nonempty_string()`, `check_nonmissing()`, `check_numeric()`, `check_numeric_or_logical()`, `check_probability()`, `check_real()`, `check_scalar_character()`, `check_scalar_date()`, `check_scalar_datetime()`, `check_scalar_double()`, `check_scalar_factor()`, `check_scalar_integer()`, `check_scalar_numeric()`, `check_scalar_probability()`, `check_scalar_whole_number()`, `check_scalar_whole_numeric()`, `check_string()`, `check_table()`, `check_valid()`, `check_valid_numeric()`, `check_whole_number()`, `check_whole_numeric()`

Examples

```
#------------------------------------------------------------------------------
# check_scalar_logical() examples
#------------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_scalar_logical()
}

logical <- TRUE
integer <- 1L
numeric <- 1
character <- letters[1]
factor <- factor(character)

f(logical)

try(f(integer))
try(f(numeric))
try(f(character))
try(f(factor))
```

`check_scalar_numeric` *Check scalar numeric*

Description

Check if `arg` is a scalar numeric vector.

Usage

```
check_scalar_numeric(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

`arg`	(object) The argument to check.
`signal`	(string: `c("error", "warning", "message")`) Must be one of `"error"`, `"warning"`, or `"message"`. `"error"` (default) calls `base::stop()` to signal an error message. `"warning"` calls `base::warning()` to signal a warning message. `"message"` calls `base::message()` to signal a message.
`msg`	(string or `NULL`: `NULL`) A string that replaces the default message. Set as `NULL` to use the default message.
`call.`	(Scalar logical: `c(FALSE, TRUE)`) Passed to `stop()` or `warning()`. The default `msg` includes it's own formatted call, hence `call.` should be `FALSE` (default). If you insert your own `msg`, then you may want `call. = TRUE`.

Value

`invisible(arg)` or condition determined by argument `signal`.

See Also

Other check_type: `check_atomic()`, `check_character()`, `check_data_frame()`, `check_data_frame_or_matrix()`, `check_date()`, `check_date_or_datetime()`, `check_datetime()`, `check_double()`, `check_factor()`, `check_flag()`, `check_formula()`, `check_function()`, `check_function_or_numeric()`, `check_inherits()`, `check_integer()`, `check_list()`, `check_logical()`, `check_matrix()`, `check_named()`, `check_nonempty_string()`, `check_nonmissing()`, `check_numeric()`, `check_numeric_or_logical()`, `check_probability()`, `check_real()`, `check_scalar_character()`, `check_scalar_date()`, `check_scalar_datetime()`, `check_scalar_double()`, `check_scalar_factor()`, `check_scalar_integer()`, `check_scalar_logical()`, `check_scalar_probability()`, `check_scalar_whole_number()`, `check_scalar_whole_numeric()`, `check_string()`, `check_table()`, `check_valid()`, `check_valid_numeric()`, `check_whole_number()`, `check_whole_numeric()`

Examples

```
#------------------------------------------------------------------------------
# check_scalar_numeric() examples
#------------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_scalar_numeric()
}

logical <- TRUE
integer <- 1L
numeric <- 1
character <- letters[1]
factor <- factor(character)

f(integer)
f(numeric)

try(f(logical))
try(f(character))
try(f(factor))
```

---

check_scalar_probability

Check scalar probability

Description

Check if `arg` is a scalar probability (numeric vector of length 1 bounded by [0, 1]).

Usage

```
check_scalar_probability(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

`arg`	(object) The argument to check.
`signal`	(string: `c("error", "warning", "message")`) Must be one of `"error"`, `"warning"`, or `"message"`. `"error"` (default) calls `base::stop()` to signal an error message. `"warning"` calls `base::warning()` to signal a warning message. `"message"` calls `base::message()` to signal a message.
`msg`	(string or `NULL`: `NULL`) A string that replaces the default message. Set as `NULL` to use the default message.

`call.` (Scalar logical: `c(FALSE, TRUE)`)
Passed to `stop()` or `warning()`. The default `msg` includes it's own formatted call, hence `call.` should be `FALSE` (default). If you insert your own `msg`, then you may want `call. = TRUE`.

Value

`invisible(arg)` or condition determined by argument `signal`.

See Also

Other check_type: `check_atomic()`, `check_character()`, `check_data_frame()`, `check_data_frame_or_matrix()`, `check_date()`, `check_date_or_datetime()`, `check_datetime()`, `check_double()`, `check_factor()`, `check_flag()`, `check_formula()`, `check_function()`, `check_function_or_numeric()`, `check_inherits()`, `check_integer()`, `check_list()`, `check_logical()`, `check_matrix()`, `check_named()`, `check_nonempty_string()`, `check_nonmissing()`, `check_numeric()`, `check_numeric_or_logical()`, `check_probability()`, `check_real()`, `check_scalar_character()`, `check_scalar_date()`, `check_scalar_datetime()`, `check_scalar_double()`, `check_scalar_factor()`, `check_scalar_integer()`, `check_scalar_logical()`, `check_scalar_numeric()`, `check_scalar_whole_number()`, `check_scalar_whole_numeric()`, `check_string()`, `check_table()`, `check_valid()`, `check_valid_numeric()`, `check_whole_number()`, `check_whole_numeric()`

Examples

```
#--------------------------------------------------------------------------
# check_scalar_probability() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_scalar_probability()
}

logical <- TRUE
integer <- 1L
numeric <- 1
character <- letters[1]
factor <- factor(character)

f(integer)
f(numeric)

try(f(logical))
try(f(character))
try(f(factor))
```

check_scalar_whole_number *Check scalar whole number*

Description

Check if arg is a scalar whole number (to tolerance tol). Must be a numeric vector, length == 1, and not missing.

Usage

```
check_scalar_whole_number(
  arg,
  tol = 1e-08,
  signal = "error",
  msg = NULL,
  call. = FALSE
)
```

Arguments

arg	(object) The argument to check.
tol	(Scalar non-negative numeric: 1e-08) Relative differences smaller than tol are ignored, thus assumed to be a whole number.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_numeric(), check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#--------------------------------------------------------------------------
# check_scalar_whole_number() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_scalar_whole_number()
}

logical <- TRUE
integer <- 1L
numeric <- 1.000000001
character <- letters[1]
factor <- factor(character)

f(integer)
f(numeric)

try(f(logical))
try(f(character))
try(f(factor))
```

---

check_scalar_whole_numeric

Check scalar whole numeric

Description

Check if arg is a scalar whole number. Must be a numeric vector, length == 1, and not missing.

Usage

```
check_scalar_whole_numeric(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#--------------------------------------------------------------------------
# check_whole_numeric() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_scalar_whole_numeric()
}

logical <- TRUE
integer <- 1L
numeric <- 1
character <- letters[1]
factor <- factor(character)
```

```
f(integer)
f(numeric)

try(f(logical))
try(f(character))
try(f(factor))
```

`check_strict_decreasing` *Check strictly changing vector*

Description

Check if `arg` is a strictly increasing/decreasing numeric vector.

Usage

```
check_strict_decreasing(
  arg,
  na.rm = FALSE,
  signal = "error",
  msg = NULL,
  call. = FALSE
)

check_strict_increasing(
  arg,
  na.rm = FALSE,
  signal = "error",
  msg = NULL,
  call. = FALSE
)
```

Arguments

`arg`	(object) The argument to check.
`na.rm`	(scalar logical: `c(FALSE, TRUE)`) Whether or not to remove missing values before checking.
`signal`	(string: `c("error", "warning", "message")`) Must be one of `"error"`, `"warning"`, or `"message"`. `"error"` (default) calls `base::stop()` to signal an error message. `"warning"` calls `base::warning()` to signal a warning message. `"message"` calls `base::message()` to signal a message.

msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_relation: check_all_arg_in_x(), check_all_x_in_arg(), check_any_arg_in_x(), check_any_x_in_arg(), check_equals(), check_finite(), check_length_equals(), check_lt(), check_monotonic_decreasing(), check_names_in_arg(), check_range(), check_scalar_arg_in_x(), check_unique()

Examples

```
#--------------------------------------------------------------------------
# check_strict_*() examples
#--------------------------------------------------------------------------
library(bkcheck)

#--------------------------------------------------------------------------
# check_strict_decreasing()
#--------------------------------------------------------------------------
f <- function(x) {
  x |>
    check_numeric() |>
    check_strict_decreasing()
}

f(10:1)
try(f(c(3, 3, 2, 1)))
try(f(1:2))

#--------------------------------------------------------------------------
# check_strict_increasing()
#--------------------------------------------------------------------------
f <- function(x) {
  x |>
    check_numeric() |>
    check_strict_increasing()
}

f(1:10)
try(f(c(1, 2, 3, 3)))
try(f(2:1))
```

check_string *Check scalar character*

Description

Check if arg is a scalar character vector (a string).

Usage

```
check_string(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_nur check_table(), check_valid(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#--------------------------------------------------------------------------
# check_string() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_string()
}

f("a")

try(f(TRUE))
try(f(1L))
try(f(1))
try(f(c("a", "b")))
try(f(factor("a")))
```

`check_table` *Check table*

Description

Check if `arg` is a table.

Usage

```
check_table(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

`arg`	(object) The argument to check.
`signal`	(string: `c("error", "warning", "message")`) Must be one of `"error"`, `"warning"`, or `"message"`. `"error"` (default) calls `base::stop()` to signal an error message. `"warning"` calls `base::warning()` to signal a warning message. `"message"` calls `base::message()` to signal a message.
`msg`	(string or `NULL`: `NULL`) A string that replaces the default message. Set as `NULL` to use the default message.
`call.`	(Scalar logical: `c(FALSE, TRUE)`) Passed to `stop()` or `warning()`. The default `msg` includes it's own formatted call, hence `call.` should be `FALSE` (default). If you insert your own `msg`, then you may want `call. = TRUE`.

Value

`invisible(arg)` or condition determined by argument `signal`.

See Also

Other check_type: `check_atomic()`, `check_character()`, `check_data_frame()`, `check_data_frame_or_matrix()`, `check_date()`, `check_date_or_datetime()`, `check_datetime()`, `check_double()`, `check_factor()`, `check_flag()`, `check_formula()`, `check_function()`, `check_function_or_numeric()`, `check_inherits()`, `check_integer()`, `check_list()`, `check_logical()`, `check_matrix()`, `check_named()`, `check_nonempty_string()`, `check_nonmissing()`, `check_numeric()`, `check_numeric_or_logical()`, `check_probability()`, `check_real()`, `check_scalar_character()`, `check_scalar_date()`, `check_scalar_datetime()`, `check_scalar_double()`, `check_scalar_factor()`, `check_scalar_integer()`, `check_scalar_logical()`, `check_scalar_numeric()`, `check_scalar_probability()`, `check_scalar_whole_number()`, `check_scalar_whole_nu`
`check_string()`, `check_valid()`, `check_valid_numeric()`, `check_whole_number()`, `check_whole_numeric()`

Examples

```
#--------------------------------------------------------------------------
# check_table() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_table()
}

f(table(1))

try(f(list(1)))
try(f(TRUE))
try(f(1L))
try(f(1))
try(f("a"))
try(f(factor("a")))
try(f(NULL))
```

`check_unique` *Check unique*

Description

Check if `arg` is all unique.

Usage

```
check_unique(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_relation: check_all_arg_in_x(), check_all_x_in_arg(), check_any_arg_in_x(), check_any_x_in_arg(), check_equals(), check_finite(), check_length_equals(), check_lt(), check_monotonic_decreasing(), check_names_in_arg(), check_range(), check_scalar_arg_in_x(), check_strict_decreasing()

Examples

```
#--------------------------------------------------------------------------
# check_unique() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_unique()
}

f("a")

try(f(c(TRUE, TRUE)))
try(f(c(1L, 1L)))
try(f(c(1, 1)))
try(f(c("a", "b", "a")))
try(f(factor(c("a", "a"))))
```

check_valid *Check valid value*

Description

Check if arg contains no missing values (NA or NaN), is not zero-length, and is not NULL.

Usage

```
check_valid(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_nur check_string(), check_table(), check_valid_numeric(), check_whole_number(), check_whole_numeric()

Examples

```
#--------------------------------------------------------------------------
# check_valid() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_valid()
}

f(c(1, 2))
try(f(numeric(0)))
try(f(NULL))
try(f(c(1, NA)))
try(f(c(1, NaN)))
```

`check_valid_numeric` *Check valid numeric vector*

Description

Check if `arg` is a numeric vector with no missing values (`NA` or `NaN`) and length greater than zero.

Usage

```
check_valid_numeric(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

`arg`	(object) The argument to check.
`signal`	(string: `c("error", "warning", "message")`) Must be one of `"error"`, `"warning"`, or `"message"`. `"error"` (default) calls `base::stop()` to signal an error message. `"warning"` calls `base::warning()` to signal a warning message. `"message"` calls `base::message()` to signal a message.
`msg`	(string or `NULL`: `NULL`) A string that replaces the default message. Set as `NULL` to use the default message.
`call.`	(Scalar logical: `c(FALSE, TRUE)`) Passed to `stop()` or `warning()`. The default `msg` includes it's own formatted call, hence `call.` should be `FALSE` (default). If you insert your own `msg`, then you may want `call. = TRUE`.

Details

This is a shortcut for the chain `check_numeric()` and `check_valid()`. Infinite values are allowed. Use `check_real()` to also require finite values.

Value

`invisible(arg)` or condition determined by argument `signal`.

See Also

`check_real()`

Other check_type: `check_atomic()`, `check_character()`, `check_data_frame()`, `check_data_frame_or_matrix()`, `check_date()`, `check_date_or_datetime()`, `check_datetime()`, `check_double()`, `check_factor()`, `check_flag()`, `check_formula()`, `check_function()`, `check_function_or_numeric()`, `check_inherits()`, `check_integer()`, `check_list()`, `check_logical()`, `check_matrix()`, `check_named()`, `check_nonempty_string()`, `check_nonmissing()`, `check_numeric()`, `check_numeric_or_logical()`, `check_probability()`, `check_real()`, `check_scalar_character()`, `check_scalar_date()`, `check_scalar_datetime()`, `check_scalar_double()`, `check_scalar_factor()`, `check_scalar_integer()`, `check_scalar_logical()`, `check_scalar_numeric()`, `check_scalar_probability()`, `check_scalar_whole_number()`, `check_scalar_whole_nur` `check_string()`, `check_table()`, `check_valid()`, `check_whole_number()`, `check_whole_numeric()`

Examples

```
#------------------------------------------------------------------------------
# check_valid_numeric() examples
#------------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_valid_numeric()
}

integer <- 1:4
numeric <- c(1, 1.5)
character <- letters[1:4]

f(integer)
f(numeric)
f(c(1, Inf))

try(f(character))
try(f(numeric(0)))
try(f(c(1, NA)))
```

check_whole_number *Check whole number vector*

Description

Check if arg is a whole number vector (to tolerance tol). Must be a numeric vector, length > 0, and no missing values.

Usage

```
check_whole_number(
  arg,
  tol = 1e-08,
  signal = "error",
  msg = NULL,
  call. = FALSE
)
```

Arguments

arg	(object) The argument to check.
tol	(Scalar non-negative numeric: 1e-08) Relative differences smaller than tol are ignored, thus assumed to be a whole number.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_nu
check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_numeric()

Examples

```
#--------------------------------------------------------------------------
# check_whole_number() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_whole_number()
}

logical <- TRUE
integer <- -4:0
numeric <- c(-1, 0, 0.99)
character <- letters[1:4]
factor <- factor(character)

f(integer)

try(f(logical))
try(f(numeric))
try(f(character))
try(f(factor))
```

---

check_whole_numeric *Check whole numeric vector*

Description

Check if `arg` is a vector of integers stored as numeric or integer. Must be a numeric vector, length > 0, and no missing values.

Usage

```
check_whole_numeric(arg, signal = "error", msg = NULL, call. = FALSE)
```

Arguments

arg	(object) The argument to check.
signal	(string: c("error", "warning", "message")) Must be one of "error", "warning", or "message". "error" (default) calls base::stop() to signal an error message. "warning" calls base::warning() to signal a warning message. "message" calls base::message() to signal a message.
msg	(string or NULL: NULL) A string that replaces the default message. Set as NULL to use the default message.
call.	(Scalar logical: c(FALSE, TRUE)) Passed to stop() or warning(). The default msg includes it's own formatted call, hence call. should be FALSE (default). If you insert your own msg, then you may want call. = TRUE.

Value

invisible(arg) or condition determined by argument signal.

See Also

Other check_type: check_atomic(), check_character(), check_data_frame(), check_data_frame_or_matrix(), check_date(), check_date_or_datetime(), check_datetime(), check_double(), check_factor(), check_flag(), check_formula(), check_function(), check_function_or_numeric(), check_inherits(), check_integer(), check_list(), check_logical(), check_matrix(), check_named(), check_nonempty_string(), check_nonmissing(), check_numeric(), check_numeric_or_logical(), check_probability(), check_real(), check_scalar_character(), check_scalar_date(), check_scalar_datetime(), check_scalar_double(), check_scalar_factor(), check_scalar_integer(), check_scalar_logical(), check_scalar_numeric(), check_scalar_probability(), check_scalar_whole_number(), check_scalar_whole_nur check_string(), check_table(), check_valid(), check_valid_numeric(), check_whole_number()

Examples

```
#--------------------------------------------------------------------------
# check_whole_numeric() examples
#--------------------------------------------------------------------------
library(bkcheck)

f <- function(x) {
  x |>
    check_whole_numeric()
}

logical <- TRUE
integer <- -4:0
numeric <- c(-1, 0, 0.99)
character <- letters[1:4]
factor <- factor(character)
```

```
f(integer)

try(f(logical))
try(f(numeric))
try(f(character))
try(f(factor))
```

`insert_value` *Insert a value into a vector*

Description

Replaces existing elements with your supplied value(s) at random locations.

Usage

```
insert_value(x, value, prob = 1, n = 1L, keep_nonfinite = TRUE, replace = TRUE)
```

Arguments

`x`	(Atomic vector) The vector of interest.
`value`	(Atomic vector) Value(s) to be inserted into `x`. Must be same type as `x` or some special value such as `NA/NaN/Inf`. Will be randomly sampled from for insertion into `x`.
`prob`	(Scalar numeric: `[0, 1]`) The probability that elements from `value` will be inserted into `x`.
`n`	(Scalar integer: `[1, Inf]`) The number of times `value` should be sampled from and then inserted into `x`.
`keep_nonfinite`	(Scalar logical: `c(TRUE, FALSE)`) Whether or not to protect non finite values from being replaced in `x`. `TRUE` (default) protects `NA/NaN/Inf` values from being replaced.
`replace`	(Scalar logical: `c(TRUE, FALSE)`) Whether or not to sample from `value` with replacement. `TRUE` (default) samples with replacement.

Value

vector

See Also

Other simulate: `rand_chars()`, `sim_character()`, `sim_data.frame()`, `sim_date()`, `sim_datetime()`, `sim_double()`, `sim_factor()`, `sim_integer()`, `sim_logical()`

Examples

```
#--------------------------------------------------------------------------
# insert_value() examples
#--------------------------------------------------------------------------
library(bkcheck)

x <- c(1, 2, 3)
insert_value(x = x, value = 10, prob = 1)
insert_value(x = x, value = 11:20, prob = 1)
insert_value(x = x, value = 11:20, n = 3, prob = 1)
insert_value(x = x, value = NA, prob = 1)
insert_value(x = x, value = NaN, prob = 1)
insert_value(x = x, value = Inf, prob = 1)
```

---

rand_chars *Random character strings*

Description

Generate a vector of random character strings.

Usage

```
rand_chars(
  length = 1L,
  nchars = 6L,
  as_raw = c(33:47, 91:95, 123:126, 49:57, 65:90, 97:122),
  replace = TRUE
)
```

Arguments

length	(Scalar integer: `[1, Inf]`) The number of elements in the character vector.
nchars	(Scalar integer: `[1, Inf]`) The number of characters in each string.
as_raw	(Integer vector: See 'Details') Integers to be converted to `raw` type and then back to character.
replace	(Scalar logical: `c(TRUE, FALSE)`) Whether or not characters can be repeated in each string. `TRUE` (default) samples with replacement.

Details

Below is the result of converting integers to `raw` type and then back to character.

- `33:47` - `!\"#$%&'()*+,-./`
- `91:95` - `[]^_`
- `123:126` - `{|}~`
- `49:57` - `123456789`
- `65:90` - `ABCDEFGHIJKLMNOPQRSTUVWXYZ`
- `97:122` - `abcdefghijklmnopqrstuvwxyz`

Value

character

See Also

Other simulate: `insert_value()`, `sim_character()`, `sim_data.frame()`, `sim_date()`, `sim_datetime()`, `sim_double()`, `sim_factor()`, `sim_integer()`, `sim_logical()`

Examples

```
#-------------------------------------------------------------------------
# rand_chars() examples
#-------------------------------------------------------------------------
library(bkcheck)

rand_chars()
rand_chars(10)
rand_chars(length = 6, nchars = 2)
```

`sim_character` *Simulate a character vector*

Description

Simulate a vector of random strings. Provides the option of inserting base::NA or returning base::NULL or zero-length objects.

Usage

```
sim_character(
  x = rand_chars(length = 10),
  length = c(1, 10),
  prob_na = 0,
  n_na = 1L,
  prob_null = 0,
```

```
  prob_zero_length = 0,
  names = FALSE,
  replace = TRUE
)
```

Arguments

x	(character) The vector to randomly sample from.
length	(integer vector of length 1 or 2) The number of elements in the simulated vector. If `length(length)==1L`, sets the number of elements to `length`. If `length(length)==2L`, randomly selects the number of elements from the interval `[min, max]`.
prob_na	(Scalar numeric: `[0, 1]`) The probability of inserting an `NA`.
n_na	(Scalar integer: `[0, Inf]`) The number of `NA`s to insert.
prob_null	(Scalar numeric: `[0, 1]`) The probability of returning a `NULL` object.
prob_zero_length	(Scalar numeric: `[0, 1]`) The probability of returning a zero-length vector.
names	(Scalar logical: `c(TRUE, FALSE)`) Whether or not the vector should have names. `TRUE` (default) sets names as `name_1`, `name_2`, `name_3`, ...
replace	(Scalar logical: `c(TRUE, FALSE)`) Whether or not elements from `x` can be repeated in the result. `TRUE` (default) samples with replacement.

Value

character

See Also

Other simulate: `insert_value()`, `rand_chars()`, `sim_data.frame()`, `sim_date()`, `sim_datetime()`, `sim_double()`, `sim_factor()`, `sim_integer()`, `sim_logical()`

Examples

```
#--------------------------------------------------------------------------
# sim_character() examples
#--------------------------------------------------------------------------
library(bkcheck)

sim_character()
sim_character(prob_na = 1)
sim_character(prob_null = 1)
```

```
sim_character(prob_zero_length = 1)
sim_character(names = TRUE)

sim_character(replace = FALSE)
sim_character(replace = TRUE)
```

`sim_data.frame` *Simulate a data frame*

Description

Simulate a data frame with different data types. Provides the option of inserting base::NA or returning base::NULL or zero-length data frame.

Usage

```
sim_data.frame(
  x = list(logical = sim_logical(length = n), integer = sim_integer(length = n), double =
     sim_double(length = n), factor = sim_factor(length = n), character =
     sim_character(length = n), date = sim_date(length = n), datetime =
     sim_datetime(length = n)),
  n = 10L,
  prob_null = 0,
  prob_zero_length = 0
)
```

Arguments

`x`	(named list) Will be converted to `data.frame`. All list elements must be equal length vectors. Use the `bkcheck::sim_*()` functions to fill in list elements.
`n`	(Scalar integer: `[0, Inf]`) A convenience argument to set the number of rows when using `bkcheck::sim_*()` in `x`.
`prob_null`	(Scalar numeric: `[0, 1]`) The probability of returning a `NULL` object.
`prob_zero_length`	(Scalar numeric: `[0, 1]`) The probability of returning a zero-row and zero-column data frame: `data.frame()`.

Value

data.frame

See Also

Other simulate: `insert_value()`, `rand_chars()`, `sim_character()`, `sim_date()`, `sim_datetime()`, `sim_double()`, `sim_factor()`, `sim_integer()`, `sim_logical()`

Examples

```
#---------------------------------------------------------------------------
# sim_data.frame() examples
#---------------------------------------------------------------------------
library(bkcheck)

sim_data.frame()
sim_data.frame(x = list(var1 = sim_double(length = 10)))
sim_data.frame(x = list(logical = sim_logical(length = 10)))
sim_data.frame(prob_null = 1)
sim_data.frame(prob_zero_length = 1)
```

`sim_date` *Simulate a date vector*

Description

Simulate a vector of random date values (class `Date`). Provides the option of inserting `bkcheck:::NA_Date_` or returning base::NULL or zero-length dates.

Usage

```
sim_date(
  x = seq(from = as.Date(Sys.Date() - 30), to = as.Date(Sys.Date() + 30), by = "day"),
  length = c(1, 10),
  prob_na = 0,
  n_na = 1L,
  prob_null = 0,
  prob_zero_length = 0,
  names = FALSE,
  replace = TRUE
)
```

Arguments

`x`	(class `Date` vector) The vector to randomly sample from.
`length`	(integer vector of length 1 or 2) The number of elements in the simulated vector. If `length(length)==1L`, sets the number of elements to `length`. If `length(length)==2L`, randomly selects the number of elements from the interval `[min, max]`.

prob_na (Scalar numeric: [0, 1])
The probability of inserting an NA.

n_na (Scalar integer: [0, Inf])
The number of NAs to insert.

prob_null (Scalar numeric: [0, 1])
The probability of returning a NULL object.

prob_zero_length (Scalar numeric: [0, 1])
The probability of returning a zero-length vector.

names (Scalar logical: c(TRUE, FALSE))
Whether or not the vector should have names. TRUE (default) sets names as name_1, name_2, name_3, ...

replace (Scalar logical: c(TRUE, FALSE))
Whether or not elements from x can be repeated in the result. TRUE (default) samples with replacement.

Value

Date vector

See Also

Other simulate: insert_value(), rand_chars(), sim_character(), sim_data.frame(), sim_datetime(), sim_double(), sim_factor(), sim_integer(), sim_logical()

Examples

```
#---------------------------------------------------------------------------
# sim_date() examples
#---------------------------------------------------------------------------
library(bkcheck)

sim_date()
sim_date(prob_na = 1)
sim_date(prob_null = 1)
sim_date(prob_zero_length = 1)
sim_date(names = TRUE)
```

---

sim_datetime *Simulate a datetime vector*

Description

Simulate a vector of random datetime values (class POSIXct). Provides the option of inserting NAs or returning base::NULL or zero-length datetimes.

Usage

```
sim_datetime(
  x = seq(from = as.POSIXct(Sys.Date() - 5), to = as.POSIXct(Sys.Date() + 5), by =
    "hour"),
  length = c(1, 10),
  prob_na = 0,
  n_na = 1L,
  prob_null = 0,
  prob_zero_length = 0,
  names = FALSE,
  replace = TRUE
)
```

Arguments

x	(class POSIXct vector) The vector to randomly sample from.
length	(integer vector of length 1 or 2) The number of elements in the simulated vector. If length(length)==1L, sets the number of elements to length. If length(length)==2L, randomly selects the number of elements from the interval [min, max].
prob_na	(Scalar numeric: [0, 1]) The probability of inserting an NA.
n_na	(Scalar integer: [0, Inf]) The number of NAs to insert.
prob_null	(Scalar numeric: [0, 1]) The probability of returning a NULL object.
prob_zero_length	(Scalar numeric: [0, 1]) The probability of returning a zero-length vector.
names	(Scalar logical: c(TRUE, FALSE)) Whether or not the vector should have names. TRUE (default) sets names as name_1, name_2, name_3, ...
replace	(Scalar logical: c(TRUE, FALSE)) Whether or not elements from x can be repeated in the result. TRUE (default) samples with replacement.

Value

POSIXct vector

See Also

Other simulate: insert_value(), rand_chars(), sim_character(), sim_data.frame(), sim_date(), sim_double(), sim_factor(), sim_integer(), sim_logical()

Examples

```
#--------------------------------------------------------------------------
# sim_datetime() examples
#--------------------------------------------------------------------------
library(bkcheck)

sim_datetime()
sim_datetime(prob_na = 1)
sim_datetime(prob_null = 1)
sim_datetime(prob_zero_length = 1)
sim_datetime(names = TRUE)
```

`sim_double` *Simulate a double-precision vector*

Description

Simulate a vector of random double-precision values (`base::numeric()`). Provides the option of inserting zeros, base::NA, base::NaN, and base::Inf values or returning base::NULL or zero-length objects.

Usage

```
sim_double(
  x = runif(n = 10L, min = -10, max = 10),
  length = c(1L, 10L),
  prob_zero = 1,
  n_zero = 1L,
  prob_na = 0,
  n_na = 1L,
  prob_nan = 0,
  n_nan = 1L,
  prob_inf = 0,
  n_inf = 1L,
  prob_null = 0,
  prob_zero_length = 0,
  names = FALSE,
  replace = TRUE
)
```

Arguments

`x`	(Double-precision vector) The vector to randomly sample from.

`length` (integer vector of length 1 or 2)
The number of elements in the simulated vector. If `length(length)==1L`, sets the number of elements to `length`. If `length(length)==2L`, randomly selects the number of elements from the interval `[min, max]`.

`prob_zero` (Scalar numeric: `[0, 1]`)
The probability of inserting a `0`.

`n_zero` (Scalar integer: `[0, Inf]`)
The number of `0`s to insert.

`prob_na` (Scalar numeric: `[0, 1]`)
The probability of inserting an `NA`.

`n_na` (Scalar integer: `[0, Inf]`)
The number of `NA`s to insert.

`prob_nan` (Scalar numeric: `[0, 1]`)
The probability of inserting a `NaN`.

`n_nan` (Scalar integer: `[0, Inf]`)
The number of `NaN`s to insert.

`prob_inf` (Scalar numeric: `[0, 1]`)
The probability of inserting an `Inf`.

`n_inf` (Scalar integer: `[0, Inf]`)
The number of `Inf`s to insert.

`prob_null` (Scalar numeric: `[0, 1]`)
The probability of returning a `NULL` object.

`prob_zero_length` (Scalar numeric: `[0, 1]`)
The probability of returning a zero-length vector.

`names` (Scalar logical: `c(TRUE, FALSE)`)
Whether or not the vector should have names. `TRUE` (default) sets names as `name_1`, `name_2`, `name_3`, ...

`replace` (Scalar logical: `c(TRUE, FALSE)`)
Whether or not elements from `x` can be repeated in the result. `TRUE` (default) samples with replacement.

Details

The arguments list contains many possible conditions. Combining more than one set of conditions may not necessarily result in the specified behavior. For example, `sim_double(prob_null=0.9, prob_zero_length=0.9)` does not result in `NULL` and zero-length outcomes each with probability 0.9.

Value

Double-precision vector

See Also

Other simulate: `insert_value()`, `rand_chars()`, `sim_character()`, `sim_data.frame()`, `sim_date()`, `sim_datetime()`, `sim_factor()`, `sim_integer()`, `sim_logical()`

Examples

```
#--------------------------------------------------------------------------
# sim_double() examples
#--------------------------------------------------------------------------
library(bkcheck)

sim_double()
sim_double(prob_na = 1)
sim_double(length = 4, prob_na = 1, prob_nan = 1, prob_inf = 1)
sim_double(prob_null = 1)
sim_double(prob_zero_length = 1)
sim_double(names = TRUE)
```

`sim_factor` *Simulate a factor vector*

Description

Simulate a vector of random factor values. Provides the option of inserting base::NA or returning base::NULL or zero-length factors.

Usage

```
sim_factor(
  x = factor(month.name),
  length = c(1, 10),
  prob_na = 0,
  n_na = 1L,
  prob_null = 0,
  prob_zero_length = 0,
  names = FALSE,
  replace = TRUE
)
```

Arguments

`x`	(factor) The vector to randomly sample from.
`length`	(integer vector of length 1 or 2) The number of elements in the simulated vector. If `length(length)==1L`, sets the number of elements to `length`. If `length(length)==2L`, randomly selects the number of elements from the interval `[min, max]`.
`prob_na`	(Scalar numeric: `[0, 1]`) The probability of inserting an `NA`.
`n_na`	(Scalar integer: `[0, Inf]`) The number of `NA`s to insert.

prob_null (Scalar numeric: [0, 1])
The probability of returning a NULL object.

prob_zero_length (Scalar numeric: [0, 1])
The probability of returning a zero-length vector.

names (Scalar logical: c(TRUE, FALSE))
Whether or not the vector should have names. TRUE (default) sets names as name_1, name_2, name_3, ...

replace (Scalar logical: c(TRUE, FALSE))
Whether or not elements from x can be repeated in the result. TRUE (default) samples with replacement.

Value

factor

See Also

Other simulate: insert_value(), rand_chars(), sim_character(), sim_data.frame(), sim_date(), sim_datetime(), sim_double(), sim_integer(), sim_logical()

Examples

```
#---------------------------------------------------------------------------
# sim_factor() examples
#---------------------------------------------------------------------------
library(bkcheck)

sim_factor()
sim_factor(prob_na = 1)
sim_factor(prob_null = 1)
sim_factor(prob_zero_length = 1)
sim_factor(names = TRUE)

sim_factor(factor(rand_chars(10)), replace = FALSE)
sim_factor(factor(rand_chars(10)), replace = TRUE)
```

---

sim_integer *Simulate an integer vector*

Description

Simulate a vector of random integer values. Provides the option of inserting zeros, base::NA, base::NaN, and base::Inf values or returning base::NULL or zero-length objects.

Usage

```
sim_integer(
  x = -10:10,
  length = c(1, 10),
  prob_zero = 1,
  n_zero = 1L,
  prob_na = 0,
  n_na = 1L,
  prob_nan = 0,
  n_nan = 1L,
  prob_inf = 0,
  n_inf = 1L,
  prob_null = 0,
  prob_zero_length = 0,
  names = FALSE,
  replace = TRUE
)
```

Arguments

x	(integer) The vector to randomly sample from.
length	(integer vector of length 1 or 2) The number of elements in the simulated vector. If `length(length)==1L`, sets the number of elements to `length`. If `length(length)==2L`, randomly selects the number of elements from the interval `[min, max]`.
prob_zero	(Scalar numeric: `[0, 1]`) The probability of inserting a `0`.
n_zero	(Scalar integer: `[0, Inf]`) The number of `0`s to insert.
prob_na	(Scalar numeric: `[0, 1]`) The probability of inserting an `NA`.
n_na	(Scalar integer: `[0, Inf]`) The number of `NA`s to insert.
prob_nan	(Scalar numeric: `[0, 1]`) The probability of inserting a `NaN`.
n_nan	(Scalar integer: `[0, Inf]`) The number of `NaN`s to insert.
prob_inf	(Scalar numeric: `[0, 1]`) The probability of inserting an `Inf`.
n_inf	(Scalar integer: `[0, Inf]`) The number of `Inf`s to insert.
prob_null	(Scalar numeric: `[0, 1]`) The probability of returning a `NULL` object.

prob_zero_length
: (Scalar numeric: `[0, 1]`)
The probability of returning a zero-length vector.

names
: (Scalar logical: `c(TRUE, FALSE)`)
Whether or not the vector should have names. `TRUE` (default) sets names as `name_1`, `name_2`, `name_3`, ...

replace
: (Scalar logical: `c(TRUE, FALSE)`)
Whether or not elements from `x` can be repeated in the result. `TRUE` (default) samples with replacement.

Details

Note that inserting base::NaN and base::Inf coerces to double storage type.

Value

integer

See Also

Other simulate: `insert_value()`, `rand_chars()`, `sim_character()`, `sim_data.frame()`, `sim_date()`, `sim_datetime()`, `sim_double()`, `sim_factor()`, `sim_logical()`

Examples

```
#---------------------------------------------------------------------------
# sim_integer() examples
#---------------------------------------------------------------------------
library(bkcheck)

sim_integer()
sim_integer(prob_na = 1)
sim_integer(prob_na = 1, prob_nan = 1, prob_inf = 1)
sim_integer(prob_null = 1)
sim_integer(prob_zero_length = 1)
sim_integer(names = TRUE)
```

---

sim_logical *Simulate a logical vector*

Description

Simulate a vector of random logical values. Provides the option of inserting base::NA or returning base::NULL or zero-length logical.

Usage

```
sim_logical(
  prob_true = 0.5,
  length = c(1, 10),
  prob_na = 0,
  n_na = 1L,
  prob_null = 0,
  prob_zero_length = 0,
  names = FALSE
)
```

Arguments

prob_true	(Scalar numeric: [0, 1]) The probability of sampling TRUE. The probability of sampling FALSE is 1 - prob_true.
length	(integer vector of length 1 or 2) The number of elements in the simulated vector. If length(length)==1L, sets the number of elements to length. If length(length)==2L, randomly selects the number of elements from the interval [min, max].
prob_na	(Scalar numeric: [0, 1]) The probability of inserting an NA.
n_na	(Scalar integer: [0, Inf]) The number of NAs to insert.
prob_null	(Scalar numeric: [0, 1]) The probability of returning a NULL object.
prob_zero_length	(Scalar numeric: [0, 1]) The probability of returning a zero-length vector.
names	(Scalar logical: c(TRUE, FALSE)) Whether or not the vector should have names. TRUE (default) sets names as name_1, name_2, name_3, ...

Value

logical

See Also

Other simulate: insert_value(), rand_chars(), sim_character(), sim_data.frame(), sim_date(), sim_datetime(), sim_double(), sim_factor(), sim_integer()

Examples

```
#--------------------------------------------------------------------------
# sim_logical() examples
#--------------------------------------------------------------------------
```

```
library(bkcheck)

sim_logical()
sim_logical(prob_na = 1)
sim_logical(prob_null = 1)
sim_logical(prob_zero_length = 1)
sim_logical(names = TRUE)
```

Index

∗ **check_dims**
check_ncol_equals, 46
check_nrow_equals, 51
∗ **check_path**
check_file_exists, 22
∗ **check_relation**
check_all_arg_in_x, 3
check_all_x_in_arg, 4
check_any_arg_in_x, 5
check_any_x_in_arg, 7
check_equals, 20
check_finite, 24
check_length_equals, 34
check_lt, 38
check_monotonic_decreasing, 42
check_names_in_arg, 45
check_range, 57
check_scalar_arg_in_x, 60
check_strict_decreasing, 78
check_unique, 82
∗ **check_type**
check_atomic, 8
check_character, 9
check_data_frame, 11
check_data_frame_or_matrix, 12
check_date, 14
check_date_or_datetime, 15
check_datetime, 17
check_double, 18
check_factor, 21
check_flag, 25
check_formula, 27
check_function, 28
check_function_or_numeric, 29
check_inherits, 31
check_integer, 32
check_list, 35
check_logical, 37
check_matrix, 40
check_named, 44
check_nonempty_string, 48
check_nonmissing, 49
check_numeric, 52
check_numeric_or_logical, 54
check_probability, 55
check_real, 59
check_scalar_character, 62
check_scalar_date, 63
check_scalar_datetime, 64
check_scalar_double, 66
check_scalar_factor, 67
check_scalar_integer, 69
check_scalar_logical, 70
check_scalar_numeric, 72
check_scalar_probability, 73
check_scalar_whole_number, 75
check_scalar_whole_numeric, 76
check_string, 80
check_table, 81
check_valid, 84
check_valid_numeric, 85
check_whole_number, 87
check_whole_numeric, 88
∗ **simulate**
insert_value, 90
rand_chars, 91
sim_character, 92
sim_data.frame, 94
sim_date, 95
sim_datetime, 96
sim_double, 98
sim_factor, 100
sim_integer, 101
sim_logical, 103

base::dir.exists(), *23*
base::file.exists(), *23*
base::Inf, *98*, *101*, *103*
base::inherits(), *31*

base::message(), 3, 4, 6–8, 10, 11, 13, 14, 16–18, 20, 21, 23–25, 27, 28, 30, 31, 33, 34, 36, 37, 39, 41, 42, 44, 45, 47, 48, 50, 51, 53, 54, 56, 58, 59, 61–63, 65, 66, 68–70, 72, 73, 75, 77, 78, 80, 81, 83–85, 87, 89
base::NA, 92, 94, 98, 100, 101, 103
base::NaN, 98, 101, 103
base::NULL, 92, 94–96, 98, 100, 101, 103
base::numeric(), 98
base::stop(), 3, 4, 6–8, 10, 11, 13, 14, 16–18, 20, 21, 23–25, 27, 28, 30–32, 34, 36, 37, 39, 41, 42, 44, 45, 47, 48, 50, 51, 53, 54, 56, 58, 59, 61–63, 65–67, 69, 70, 72, 73, 75, 77, 78, 80, 81, 83–85, 87, 89
base::warning(), 3, 4, 6–8, 10, 11, 13, 14, 16–18, 20, 21, 23–25, 27, 28, 30–32, 34, 36, 37, 39, 41, 42, 44, 45, 47, 48, 50, 51, 53, 54, 56, 58, 59, 61–63, 65–67, 69, 70, 72, 73, 75, 77, 78, 80, 81, 83–85, 87, 89

check_all_arg_in_x, 3
check_all_arg_in_x(), 5–7, 20, 25, 34, 39, 43, 46, 58, 61, 79, 83
check_all_x_in_arg, 4
check_all_x_in_arg(), 4, 6, 7, 20, 25, 34, 39, 43, 46, 58, 61, 79, 83
check_any_arg_in_x, 5
check_any_arg_in_x(), 4, 5, 7, 20, 25, 34, 39, 43, 46, 58, 61, 79, 83
check_any_x_in_arg, 7
check_any_x_in_arg(), 4–6, 20, 25, 34, 39, 43, 46, 58, 61, 79, 83
check_atomic, 8
check_atomic(), 10, 12, 13, 15–17, 19, 22, 26, 27, 29–31, 33, 36, 37, 41, 44, 49, 50, 53, 55, 56, 60, 62, 64–66, 68, 69, 71, 72, 74, 76, 77, 80, 82, 84, 86, 88, 89
check_character, 9
check_character(), 9, 12, 13, 15–17, 19, 22, 26, 27, 29–31, 33, 36, 37, 41, 44, 49, 50, 53, 55, 56, 60, 62, 64–66, 68, 69, 71, 72, 74, 76, 77, 80, 82, 84, 86, 88, 89
check_data_frame, 11
check_data_frame(), 9, 10, 13, 15–17, 19, 22, 26, 27, 29–31, 33, 36, 37, 41, 44, 49, 50, 53, 55, 56, 60, 62, 64–66, 68, 69, 71, 72, 74, 76, 77, 80, 82, 84, 86, 88, 89
check_data_frame_or_matrix, 12
check_data_frame_or_matrix(), 9, 10, 12, 15–17, 19, 22, 26, 27, 29–31, 33, 36, 37, 41, 44, 49, 50, 53, 55, 56, 60, 62, 64–66, 68, 69, 71, 72, 74, 76, 77, 80, 82, 84, 86, 88, 89
check_date, 14
check_date(), 9, 10, 12, 13, 16, 17, 19, 22, 26, 27, 29–31, 33, 36, 37, 41, 44, 49, 50, 53, 55, 56, 60, 62, 64–66, 68, 69, 71, 72, 74, 76, 77, 80, 82, 84, 86, 88, 89
check_date_or_datetime, 15
check_date_or_datetime(), 9, 10, 12, 13, 15, 17, 19, 22, 26, 27, 29–31, 33, 36, 37, 41, 44, 49, 50, 53, 55, 56, 60, 62, 64–66, 68, 69, 71, 72, 74, 76, 77, 80, 82, 84, 86, 88, 89
check_datetime, 17
check_datetime(), 9, 10, 12, 13, 15, 16, 19, 22, 26, 27, 29–31, 33, 36, 37, 41, 44, 49, 50, 53, 55, 56, 60, 62, 64–66, 68, 69, 71, 72, 74, 76, 77, 80, 82, 84, 86, 88, 89
check_dir_exists *(*check_file_exists*)*, 22
check_double, 18
check_double(), 9, 10, 12, 13, 15–17, 22, 26, 27, 29–31, 33, 36, 37, 41, 44, 49, 50, 53, 55, 56, 60, 62, 64–66, 68, 69, 71, 72, 74, 76, 77, 80, 82, 84, 86, 88, 89
check_equals, 20
check_equals(), 4–7, 25, 34, 39, 43, 46, 58, 61, 79, 83
check_factor, 21
check_factor(), 9, 10, 12, 13, 15–17, 19, 26, 27, 29–31, 33, 36, 37, 41, 44, 49, 50, 53, 55, 56, 60, 62, 64–66, 68, 69, 71, 72, 74, 76, 77, 80, 82, 84, 86, 88, 89
check_file_exists, 22
check_finite, 24
check_finite(), 4–7, 20, 34, 39, 43, 46, 58, 61, 79, 83
check_flag, 25

check_flag(), *9, 10*, *12, 13*, *15–17*, *19*, *22*, *27*, *29–31*, *33*, *36, 37*, *41*, *44*, *49, 50*, *53*, *55, 56*, *60*, *62*, *64–66*, *68, 69*, *71, 72*, *74*, *76, 77*, *80*, *82*, *84*, *86*, *88, 89*
check_formula, 27
check_formula(), *9, 10*, *12, 13*, *15–17*, *19*, *22*, *26*, *29–31*, *33*, *36, 37*, *41*, *44*, *49, 50*, *53*, *55, 56*, *60*, *62*, *64–66*, *68, 69*, *71, 72*, *74*, *76, 77*, *80*, *82*, *84*, *86*, *88, 89*
check_function, 28
check_function(), *9, 10*, *12, 13*, *15–17*, *19*, *22*, *26, 27*, *30, 31*, *33*, *36, 37*, *41*, *44*, *49, 50*, *53*, *55, 56*, *60*, *62*, *64–66*, *68, 69*, *71, 72*, *74*, *76, 77*, *80*, *82*, *84*, *86*, *88, 89*
check_function_or_numeric, 29
check_function_or_numeric(), *9, 10*, *12, 13*, *15–17*, *19*, *22*, *26, 27*, *29*, *31*, *33*, *36, 37*, *41*, *44*, *49, 50*, *53*, *55, 56*, *60*, *62*, *64–66*, *68, 69*, *71, 72*, *74*, *76, 77*, *80*, *82*, *84*, *86*, *88, 89*
check_geq *(*check_lt*)*, 38
check_gt *(*check_lt*)*, 38
check_inherits, 31
check_inherits(), *9, 10*, *12, 13*, *15–17*, *19*, *22*, *26, 27*, *29, 30*, *33*, *36, 37*, *41*, *44*, *49, 50*, *53*, *55, 56*, *60*, *62*, *64–66*, *68, 69*, *71, 72*, *74*, *76, 77*, *80*, *82*, *84*, *86*, *88, 89*
check_integer, 32
check_integer(), *9, 10*, *12, 13*, *15–17*, *19*, *22*, *26, 27*, *29–31*, *36, 37*, *41*, *44*, *49, 50*, *53*, *55, 56*, *60*, *62*, *64–66*, *68, 69*, *71, 72*, *74*, *76, 77*, *80*, *82*, *84*, *86*, *88, 89*
check_length_equals, 34
check_length_equals(), *4–7*, *20*, *25*, *39*, *43*, *46*, *58*, *61*, *79*, *83*
check_length_geq *(*check_length_equals*)*, 34
check_length_leq *(*check_length_equals*)*, 34
check_leq *(*check_lt*)*, 38
check_list, 35
check_list(), *9, 10*, *12, 13*, *15–17*, *19*, *22*, *26, 27*, *29–31*, *33*, *37*, *41*, *44*, *49, 50*, *53*, *55, 56*, *60*, *62*, *64–66*, *68, 69*, *71, 72*, *74*, *76, 77*, *80*, *82*, *84*, *86*, *88, 89*
check_logical, 37
check_logical(), *9, 10*, *12, 13*, *15–17*, *19*, *22*, *26, 27*, *29–31*, *33*, *36*, *41*, *44*, *49, 50*, *53*, *55, 56*, *60*, *62*, *64–66*, *68, 69*, *71, 72*, *74*, *76, 77*, *80*, *82*, *84*, *86*, *88, 89*
check_lt, 38
check_lt(), *4–7*, *20*, *25*, *34*, *43*, *46*, *58*, *61*, *79*, *83*
check_matrix, 40
check_matrix(), *9, 10*, *12, 13*, *15–17*, *19*, *22*, *26, 27*, *29–31*, *33*, *36, 37*, *44*, *49, 50*, *53*, *55, 56*, *60*, *62*, *64–66*, *68, 69*, *71, 72*, *74*, *76, 77*, *80*, *82*, *84*, *86*, *88, 89*
check_monotonic_decreasing, 42
check_monotonic_decreasing(), *4–7*, *20*, *25*, *34*, *39*, *46*, *58*, *61*, *79*, *83*
check_monotonic_increasing *(*check_monotonic_decreasing*)*, 42
check_named, 44
check_named(), *9, 10*, *12, 13*, *15–17*, *19*, *22*, *26, 27*, *29–31*, *33*, *36, 37*, *41*, *49, 50*, *53*, *55, 56*, *60*, *62*, *64–66*, *68, 69*, *71, 72*, *74*, *76, 77*, *80*, *82*, *84*, *86*, *88, 89*
check_names_in_arg, 45
check_names_in_arg(), *4–7*, *20*, *25*, *34*, *39*, *43*, *58*, *61*, *79*, *83*
check_ncol_equals, 46
check_ncol_equals(), *51*
check_ncol_geq *(*check_ncol_equals*)*, 46
check_ncol_leq *(*check_ncol_equals*)*, 46
check_nonempty_string, 48
check_nonempty_string(), *9, 10*, *12, 13*, *15–17*, *19*, *22*, *26, 27*, *29–31*, *33*, *36, 37*, *41*, *44*, *50*, *53*, *55, 56*, *60*, *62*, *64–66*, *68, 69*, *71, 72*, *74*, *76, 77*, *80*, *82*, *84*, *86*, *88, 89*
check_nonmissing, 49
check_nonmissing(), *9, 10*, *12, 13*, *15–17*, *19*, *22*, *26, 27*, *29–31*, *33*, *36, 37*, *41*, *44*, *48, 49*, *53*, *55, 56*, *60*, *62*, *64–66*, *68, 69*, *71, 72*, *74*, *76, 77*, *80*, *82*, *84*, *86*, *88, 89*
check_nrow_equals, 51
check_nrow_equals(), *47*
check_nrow_geq *(*check_nrow_equals*)*, 51

check_nrow_leq *(*check_nrow_equals*)*, 51
check_numeric, 52
check_numeric(), *9, 10*, *12, 13*, *15–17*, *19*, *22*, *26, 27*, *29–31*, *33*, *36, 37*, *41*, *44*, *49, 50*, *55, 56*, *60*, *62*, *64–66*, *68, 69*, *71, 72*, *74*, *76, 77*, *80*, *82*, *84*, *86*, *88, 89*
check_numeric_or_logical, 54
check_numeric_or_logical(), *9, 10*, *12, 13*, *15–17*, *19*, *22*, *26, 27*, *29–31*, *33*, *36, 37*, *41*, *44*, *49, 50*, *53*, *56*, *60*, *62*, *64–66*, *68, 69*, *71, 72*, *74*, *76, 77*, *80*, *82*, *84*, *86*, *88, 89*
check_probability, 55
check_probability(), *9, 10*, *12, 13*, *15–17*, *19*, *22*, *26, 27*, *29–31*, *33*, *36, 37*, *41*, *44*, *49, 50*, *53*, *55*, *60*, *62*, *64–66*, *68, 69*, *71, 72*, *74*, *76, 77*, *80*, *82*, *84*, *86*, *88, 89*
check_range, 57
check_range(), *4–7*, *20*, *25*, *34*, *39*, *43*, *46*, *61*, *79*, *83*
check_real, 59
check_real(), *9, 10*, *12, 13*, *15–17*, *19*, *22*, *26, 27*, *29–31*, *33*, *36, 37*, *41*, *44*, *49, 50*, *53*, *55, 56*, *62*, *64–66*, *68, 69*, *71, 72*, *74*, *76, 77*, *80*, *82*, *84*, *86*, *88, 89*
check_scalar_arg_in_x, 60
check_scalar_arg_in_x(), *4–7*, *20*, *25*, *34*, *39*, *43*, *46*, *58*, *79*, *83*
check_scalar_character, 62
check_scalar_character(), *9, 10*, *12, 13*, *15–17*, *19*, *22*, *26, 27*, *29–31*, *33*, *36, 37*, *41*, *44*, *49, 50*, *53*, *55, 56*, *60*, *64–66*, *68, 69*, *71, 72*, *74*, *76, 77*, *80*, *82*, *84*, *86*, *88, 89*
check_scalar_date, 63
check_scalar_date(), *9, 10*, *12, 13*, *15–17*, *19*, *22*, *26, 27*, *29–31*, *33*, *36, 37*, *41*, *44*, *49, 50*, *53*, *55, 56*, *60*, *62*, *65, 66*, *68, 69*, *71, 72*, *74*, *76, 77*, *80*, *82*, *84*, *86*, *88, 89*
check_scalar_datetime, 64
check_scalar_datetime(), *9, 10*, *12, 13*, *15–17*, *19*, *22*, *26, 27*, *29–31*, *33*, *36, 37*, *41*, *44*, *49, 50*, *53*, *55, 56*, *60*, *62*, *64*, *66*, *68, 69*, *71, 72*, *74*, *76, 77*, *80*, *82*, *84*, *86*, *88, 89*
check_scalar_double, 66
check_scalar_double(), *9, 10*, *12, 13*, *15–17*, *19*, *22*, *26, 27*, *29, 30*, *32, 33*, *36, 37*, *41*, *44*, *49, 50*, *53*, *55, 56*, *60*, *62*, *64, 65*, *68, 69*, *71, 72*, *74*, *76, 77*, *80*, *82*, *84*, *86*, *88, 89*
check_scalar_factor, 67
check_scalar_factor(), *9, 10*, *12, 13*, *15–17*, *19*, *22*, *26, 27*, *29, 30*, *32, 33*, *36, 37*, *41*, *44*, *49, 50*, *53*, *55, 56*, *60*, *62*, *64, 65*, *67*, *69*, *71, 72*, *74*, *76, 77*, *80*, *82*, *84*, *86*, *88, 89*
check_scalar_integer, 69
check_scalar_integer(), *9, 10*, *12, 13*, *15–17*, *19*, *22*, *26, 27*, *29, 30*, *32, 33*, *36, 37*, *41*, *44*, *49, 50*, *53*, *55, 56*, *60*, *62*, *64, 65*, *67, 68*, *71, 72*, *74*, *76, 77*, *80*, *82*, *84*, *86*, *88, 89*
check_scalar_logical, 70
check_scalar_logical(), *9, 10*, *12, 13*, *15–17*, *19*, *22*, *26, 27*, *29, 30*, *32, 33*, *36, 37*, *41*, *44*, *49, 50*, *53*, *55, 56*, *60*, *62*, *64, 65*, *67–69*, *72*, *74*, *76, 77*, *80*, *82*, *84*, *86*, *88, 89*
check_scalar_numeric, 72
check_scalar_numeric(), *9, 10*, *12, 13*, *15, 16*, *18, 19*, *22*, *26, 27*, *29, 30*, *32, 33*, *36*, *38*, *41*, *44*, *49, 50*, *53*, *55, 56*, *60*, *62*, *64, 65*, *67–69*, *71*, *74*, *76, 77*, *80*, *82*, *84*, *86*, *88, 89*
check_scalar_probability, 73
check_scalar_probability(), *9, 10*, *12, 13*, *15, 16*, *18, 19*, *22*, *26, 27*, *29, 30*, *32, 33*, *36*, *38*, *41*, *44*, *49, 50*, *53*, *55, 56*, *60*, *62*, *64, 65*, *67–69*, *71, 72*, *76, 77*, *80*, *82*, *84*, *86*, *88, 89*
check_scalar_whole_number, 75
check_scalar_whole_number(), *9, 10*, *12, 13*, *15, 16*, *18, 19*, *22*, *26, 27*, *29, 30*, *32, 33*, *36*, *38*, *41*, *44*, *49, 50*, *53*, *55, 56*, *60*, *62*, *64, 65*, *67–69*, *71, 72*, *74*, *77*, *80*, *82*, *84*, *86*, *88, 89*
check_scalar_whole_numeric, 76
check_scalar_whole_numeric(), *9, 10*, *12, 13*, *15, 16*, *18, 19*, *22*, *26, 27*, *29, 30*, *32, 33*, *36*, *38*, *41*, *44*, *49, 50*, *53*, *55, 56*, *60*, *62*, *64, 65*, *67–69*, *71, 72*, *74*, *76*, *80*, *82*, *84*, *86*, *88, 89*

check_strict_decreasing, 78
check_strict_decreasing(), *4–7*, *20*, *25*, *34*, *39*, *43*, *46*, *58*, *61*, *83*
check_strict_increasing (check_strict_decreasing), 78
check_string, 80
check_string(), *9, 10*, *12, 13*, *15, 16*, *18, 19*, *22*, *26, 27*, *29, 30*, *32, 33*, *36*, *38*, *41*, *44*, *48–50*, *53*, *55, 56*, *60*, *62*, *64, 65*, *67–69*, *71, 72*, *74*, *76, 77*, *82*, *84*, *86*, *88, 89*
check_table, 81
check_table(), *9, 10*, *12, 13*, *15, 16*, *18, 19*, *22*, *26, 27*, *29, 30*, *32, 33*, *36*, *38*, *41*, *44*, *49, 50*, *53*, *55, 56*, *60*, *62*, *64, 65*, *67–69*, *71, 72*, *74*, *76, 77*, *80*, *84*, *86*, *88, 89*
check_unique, 82
check_unique(), *4–7*, *20*, *25*, *34*, *39*, *43*, *46*, *58*, *61*, *79*
check_valid, 84
check_valid(), *9, 10*, *12, 13*, *15, 16*, *18, 19*, *22*, *26, 27*, *29, 30*, *32, 33*, *36*, *38*, *41*, *44*, *49, 50*, *53*, *55, 56*, *60*, *62*, *64, 65*, *67–69*, *71, 72*, *74*, *76, 77*, *80*, *82*, *86*, *88, 89*
check_valid_numeric, 85
check_valid_numeric(), *9, 10*, *12, 13*, *15, 16*, *18, 19*, *22*, *26, 27*, *29, 30*, *32, 33*, *36*, *38*, *41*, *44*, *49, 50*, *53*, *55, 56*, *59, 60*, *62*, *64, 65*, *67–69*, *71, 72*, *74*, *76, 77*, *80*, *82*, *84*, *88, 89*
check_whole_number, 87
check_whole_number(), *9, 10*, *12, 13*, *15, 16*, *18, 19*, *22*, *26, 27*, *29, 30*, *32, 33*, *36*, *38*, *41*, *44*, *49, 50*, *53*, *55, 56*, *60*, *62*, *64, 65*, *67–69*, *71, 72*, *74*, *76, 77*, *80*, *82*, *84*, *86*, *89*
check_whole_numeric, 88
check_whole_numeric(), *9, 10*, *12, 13*, *15, 16*, *18, 19*, *22*, *26, 27*, *29, 30*, *32, 33*, *36*, *38*, *41*, *44*, *49, 50*, *53*, *55, 56*, *60*, *62*, *64, 65*, *67–69*, *71, 72*, *74*, *76, 77*, *80*, *82*, *84*, *86*, *88*

insert_value, 90
insert_value(), *92, 93*, *95–97*, *99*, *101*, *103, 104*

rand_chars, 91
rand_chars(), *90*, *93*, *95–97*, *99*, *101*, *103, 104*

sim_character, 92
sim_character(), *90*, *92*, *95–97*, *99*, *101*, *103, 104*
sim_data.frame, 94
sim_data.frame(), *90*, *92, 93*, *96, 97*, *99*, *101*, *103, 104*
sim_date, 95
sim_date(), *90*, *92, 93*, *95*, *97*, *99*, *101*, *103, 104*
sim_datetime, 96
sim_datetime(), *90*, *92, 93*, *95, 96*, *99*, *101*, *103, 104*
sim_double, 98
sim_double(), *90*, *92, 93*, *95–97*, *101*, *103, 104*
sim_factor, 100
sim_factor(), *90*, *92, 93*, *95–97*, *99*, *103, 104*
sim_integer, 101
sim_integer(), *90*, *92, 93*, *95–97*, *99*, *101*, *104*
sim_logical, 103
sim_logical(), *90*, *92, 93*, *95–97*, *99*, *101*, *103*