Package: checkCLI (via r-universe)

May 21, 2026

Type Package

Title 'CLI' Messages for Checkmate Assertions and Checks

Version 1.0

Description Providing more beautiful and more meaningful return messages for checkmate assertions and checks helping users to better understand errors.

License MIT + file LICENSE

Imports checkmate, cli, glue, purrr, stringr

Encoding UTF-8

LazyData true

Suggests knitr, R6, rmarkdown, testthat (>= 3.0.0)

Depends R (>= 4.1.0)

Config/testthat/edition 3

RoxygenNote 7.3.3

VignetteBuilder knitr

Config/pak/sysreqs libicu-dev

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

Date/Publication 2026-01-21 12:00:55 UTC

RemoteUrl https://github.com/luizesser/checkcli

RemoteRef HEAD

RemoteSha 0a076af231e3dce4c69eb84d01be96a6dcd75d5f

Contents

`assert_cli` *Combine multiple CLI assertions*

Description

Combine multiple ‘check_*_cli()‘ expressions with ‘"or"‘ or ‘"and"‘ logic and assert them jointly.

Usage

```
assert_cli(..., combine = "or", .var.name = NULL, add = NULL)
```

Arguments

`...`	Expressions evaluating to the result of ‘check_*_cli()‘ calls.
`combine`	Either ‘"or"‘ or ‘"and"‘.
`.var.name`	Optional variable name(s) for the combined assertion.
`add`	Optional [checkmate::AssertCollection] to collect failures.

Value

Invisible ‘TRUE‘ on success, otherwise error or collected failures.

Examples

```
x <- 1L
assert_cli(
  check_int_cli(x),
  check_numeric_cli(x),
  combine = "or",
  .var.name = "x"
)
```

checkcli-containers *Container and data structure CLI assertions*

Description

These functions wrap container and structural checks from **checkmate** and either return the underlying check result (‘check_*_cli()‘) or raise a ‘cli::cli_abort()‘ error via [make_assertion()] (‘assert_*_cli()‘). They target common R data structures such as arrays, matrices, generic vectors, lists, data frames, factors, environments, functions, formulas, R6 objects, and raw vectors, while emitting consistent CLI-styled error messages.

Usage

```
check_array_cli(...)

assert_array_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_matrix_cli(...)

assert_matrix_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_vector_cli(...)

assert_vector_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_list_cli(...)

assert_list_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_data_frame_cli(...)

assert_data_frame_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_factor_cli(...)

assert_factor_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_environment_cli(...)

assert_environment_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_function_cli(...)

assert_function_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_formula_cli(...)
```

```
assert_formula_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_r6_cli(...)

assert_r6_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_raw_cli(...)

assert_raw_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)
```

Arguments

`...`	Additional arguments passed on to the corresponding **checkmate** function (e.g. ‘lower‘, ‘upper‘, ‘any.missing‘, ‘all.missing‘, ‘min.len‘, ‘null.ok‘).
`x`	Object to be checked. This is the value whose type, length, or other properties are validated.
`.var.name`	Character scalar used in error messages to refer to the checked object. Defaults to [checkmate::vname()], which tries to infer the variable name from the calling context.
`add`	Optional [checkmate::AssertCollection] to which assertion failures are added instead of triggering an immediate error. Defaults to ‘NULL‘, which causes a ‘cli::cli_abort()‘ on failure.

- Generic containers: - [check_array_cli()], [assert_array_cli()] for multi-dimensional arrays. - [check_matrix_cli()], [assert_matrix_cli()] for 2D matrices. - [check_vector_cli()], [assert_vector_cli()] for (atomic or list) vectors. - [check_list_cli()], [assert_list_cli()] for lists. - [check_data_frame_cli()], [assert_data_frame_cli()] for data frames. - Typed / special containers: - [check_factor_cli()], [assert_factor_cli()] for factor vectors. - [check_environment_cli()], [assert_environment_cli()] for environments. - [check_function_cli()], [assert_function_cli()] for functions. - [check_formula_cli()], [assert_formula_cli()] for formulas. - [check_r6_cli()], [assert_r6_cli()] for R6 objects. - [check_raw_cli()], [assert_raw_cli()] for raw vectors.

In all ‘assert_*_cli()‘ functions, ‘x‘ is the object being checked and ‘.var.name‘ is used only for error message construction; the return value is always ‘x‘ (invisibly) on success.

Value

- ‘check_*_cli()‘ functions return ‘TRUE‘ on success or a character vector describing the failure, exactly like the corresponding **checkmate** checks. - ‘assert_*_cli()‘ functions return ‘x‘ invisibly on success and either: - raise a ‘cli::cli_abort()‘ error with bullet-style messages, or - push messages into an [checkmate::AssertCollection] if ‘add‘ is supplied.

See Also

[checkmate::check_array()], [checkmate::check_matrix()], [checkmate::check_vector()], [checkmate::check_list()], [checkmate::check_data_frame()], [checkmate::check_factor()], [checkmate::check_environment()], [checkmate::check_function()], [checkmate::check_formula()], [checkmate::check_raw()], [checkmate::check_r6()], [make_assertion()], [assert_cli()]

Other checkCLI: checkcli-datetime-os, checkcli-files, checkcli-names-sets, checkcli-scalars

Examples

```
# Arrays and matrices:

# 3D array (e.g. raster stack, simulation outputs)
arr <- array(1:12, dim = c(2, 2, 3))
check_array_cli(arr)
try(check_array_cli(list(1, 2, 3)))              # failure: not an array

assert_array_cli(arr)
try(assert_array_cli(list(1, 2, 3)))         # cli-styled error

# Simple 2x2 matrix
mat <- matrix(1:4, nrow = 2)
check_matrix_cli(mat)
try(check_matrix_cli(1:4))                       # failure: not a matrix

assert_matrix_cli(mat)
try(assert_matrix_cli(1:4))

# Vectors, lists, and data frames:

# Generic vector (atomic or list)
v <- 1:5
check_vector_cli(v)
try(check_vector_cli(mean))                      # failure: function, not vector

assert_vector_cli(v)
try(assert_vector_cli(mean))

# Lists
lst <- list(a = 1, b = 2)
check_list_cli(lst)
try(check_list_cli(1:3))                         # failure: atomic, not list

assert_list_cli(lst)
try(assert_list_cli(1:3))

# Data frames (e.g. species-by-site table)
df <- data.frame(x = 1:3, y = c("a", "b", "c"))
check_data_frame_cli(df)
try(check_data_frame_cli(list(x = 1:3)))         # failure

assert_data_frame_cli(df)
try(assert_data_frame_cli(list(x = 1:3)))

# Factors:

f <- factor(c("forest", "savanna", "forest"))
check_factor_cli(f)
try(check_factor_cli(c("forest", "savanna")))    # failure: not a factor
```

```
assert_factor_cli(f)
try(assert_factor_cli(c("forest", "savanna")))

# Environments and functions:

env <- new.env()
env$x <- 1
check_environment_cli(env)
try(check_environment_cli(list(x = 1)))          # failure

assert_environment_cli(env)
try(assert_environment_cli(list(x = 1)))

fun <- function(x) x * 2
check_function_cli(fun)
try(check_function_cli(1:3))                     # failure

assert_function_cli(fun)
try(assert_function_cli(1:3))

# Formulas:

fo <- abundance ~ temperature + precipitation
check_formula_cli(fo)
try(check_formula_cli("abundance ~ temperature")) # failure

assert_formula_cli(fo)
try(assert_formula_cli("abundance ~ temperature"))

# R6 objects:

if (requireNamespace("R6", quietly = TRUE)) {
  Model <- R6::R6Class("Model", public = list(fit = function(x) x))
  m <- Model$new()

  check_r6_cli(m)
  try(check_r6_cli(list()))                      # failure

  assert_r6_cli(m)
  try(assert_r6_cli(list()))
}

# Raw vectors:

r <- charToRaw("abc")
check_raw_cli(r)
try(check_raw_cli(c(1L, 2L)))                    # failure

assert_raw_cli(r)
try(assert_raw_cli(c(1L, 2L)))
```

checkcli-datetime-os *Date, time, and OS CLI assertions*

Description

These functions wrap **checkmate** checks for dates, POSIXct date-times, and operating systems, returning the underlying check result (‘check_*_cli()‘) or raising a ‘cli::cli_abort()‘ error via [make_assertion()] (‘assert_*_cli()‘). They are useful in user-facing workflows that depend on temporal objects (e.g., time series, observation timestamps) or that only support specific operating systems, while emitting consistent CLI-styled error messages.

Usage

```
check_date_cli(...)

assert_date_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_posixct_cli(...)

assert_posixct_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_os_cli(...)

assert_os_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)
```

Arguments

`...`	Additional arguments passed on to the corresponding **checkmate** function (e.g. ‘lower‘, ‘upper‘, ‘any.missing‘, ‘all.missing‘, ‘min.len‘, ‘null.ok‘).
`x`	Object to be checked. This is the value whose type, length, or other properties are validated.
`.var.name`	Character scalar used in error messages to refer to the checked object. Defaults to [checkmate::vname()], which tries to infer the variable name from the calling context.
`add`	Optional [checkmate::AssertCollection] to which assertion failures are added instead of triggering an immediate error. Defaults to ‘NULL‘, which causes a ‘cli::cli_abort()‘ on failure. - Dates: - [check_date_cli()], [assert_date_cli()] ensure that ‘x‘ is a ‘Date‘ vector and can enforce additional constraints such as length, bounds, or missingness. - POSIXct date-times: - [check_posixct_cli()], [assert_posixct_cli()] ensure that ‘x‘ is a ‘POSIXct‘ vector, suitable for time stamps with time zones. - Operating system: - [check_os_cli()], [assert_os_cli()] check that the current OS matches a given specification (e.g. ‘"windows"‘, ‘"unix"‘). In all ‘assert_*_cli()‘ functions, ‘x‘ is the object being checked and ‘.var.name‘ is used only for error message construction; the return value is always ‘x‘ (invisibly) on success.

Value

- ‘check_*_cli()‘ functions return ‘TRUE‘ on success or a character vector describing the failure, exactly like the corresponding **checkmate** checks. - ‘assert_*_cli()‘ functions return ‘x‘ invisibly on success and either: - raise a ‘cli::cli_abort()‘ error with bullet-style messages, or - push messages into an [checkmate::AssertCollection] if ‘add‘ is supplied.

See Also

[checkmate::check_date()], [checkmate::check_posixct()], [checkmate::check_os()], [make_assertion()], [assert_cli()]

Other checkCLI: checkcli-containers, checkcli-files, checkcli-names-sets, checkcli-scalars

Examples

```
# Dates:

# Valid Date vector
d <- as.Date(c("2020-01-01", "2020-02-01"))
check_date_cli(d)

# Character input is not a Date
try(check_date_cli(c("2020-01-01", "2020-02-01")))   # failure

assert_date_cli(d)
try(assert_date_cli(c("2020-01-01", "2020-02-01")))

# Single observation date
obs_date <- Sys.Date()
assert_date_cli(obs_date)

# POSIXct date-times:

# Valid POSIXct vector
t <- as.POSIXct(c("2020-01-01 12:00:00", "2020-01-02 08:30:00"), tz = "UTC")
check_posixct_cli(t)

# Plain Date is not POSIXct
try(check_posixct_cli(as.Date("2020-01-01")))        # failure

assert_posixct_cli(t)
try(assert_posixct_cli(as.Date("2020-01-01")))

# Typical use in logging or time series
ts_times <- Sys.time() + 0:9
assert_posixct_cli(ts_times)

# Operating system:

# Check that the current OS is one of the supported ones
# (e.g., skip functions not available on Windows)
try(check_os_cli("unix"))                      # TRUE on Linux/macOS, failure on Windows
```

```
# Assert OS:
try(assert_os_cli("unix"))
```

---

checkcli-files *File, directory, and path CLI assertions*

Description

These functions wrap filesystem-related checks from **checkmate** and either return the underlying check result (‘check_*_cli()‘) or raise a ‘cli::cli_abort()‘ error via [make_assertion()] (‘assert_*_cli()‘). They are useful for validating input and output locations in user-facing functions, ensuring that required files and directories exist (or can be created) and that paths are safe to write to, while emitting consistent CLI-styled error messages.

Usage

```
check_file_cli(...)

assert_file_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_file_exists_cli(...)

assert_file_exists_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_directory_cli(...)

assert_directory_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_directory_exists_cli(...)

assert_directory_exists_cli(
  x,
  ...,
  .var.name = checkmate::vname(x),
  add = NULL
)

check_path_for_output_cli(...)

assert_path_for_output_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)
```

Arguments

...	Additional arguments passed on to the corresponding **checkmate** function (e.g. ‘lower‘, ‘upper‘, ‘any.missing‘, ‘all.missing‘, ‘min.len‘, ‘null.ok‘).

x	Object to be checked. This is the value whose type, length, or other properties are validated.
.var.name	Character scalar used in error messages to refer to the checked object. Defaults to [checkmate::vname()], which tries to infer the variable name from the calling context.
add	Optional [checkmate::AssertCollection] to which assertion failures are added instead of triggering an immediate error. Defaults to 'NULL', which causes a 'cli::cli_abort()' on failure. - Files: - [check_file_cli()], [assert_file_cli()] validate file paths with properties such as existence, readability, and writability. - [check_file_exists_cli()], [assert_file_exists_cli()] specifically check that a file exists. - Directories: - [check_directory_cli()], [assert_directory_cli()] wrap [checkmate::checkDirectory()] for directory-like paths (often used for input). - [check_directory_exists_cli()], [assert_directory_exists_cli()] ensure that a directory already exists. - Output paths: - [check_path_for_output_cli()], [assert_path_for_output_cli()] ensure that a path is suitable for writing output (e.g., directory exists, file does not unexpectedly overwrite unless allowed). In all 'assert_*_cli()' functions, 'x' is the object being checked and '.var.name' is used only for error message construction; the return value is always 'x' (invisibly) on success.

Value

- 'check_*_cli()' functions return 'TRUE' on success or a character vector describing the failure, exactly like the corresponding **checkmate** checks. - 'assert_*_cli()' functions return 'x' invisibly on success and either: - raise a 'cli::cli_abort()' error with bullet-style messages, or - push messages into an [checkmate::AssertCollection] if 'add' is supplied.

See Also

[checkmate::checkFile()], [checkmate::check_file_exists()], [checkmate::checkDirectory()], [checkmate::check_directory_exists()], [checkmate::check_path_for_output()], [make_assertion()], [assert_cli()]

Other checkCLI: checkcli-containers, checkcli-datetime-os, checkcli-names-sets, checkcli-scalars

Examples

```
# Files:

# Create a temporary file for demonstration
f <- tempfile(fileext = ".csv")
write.csv(data.frame(x = 1:3), f, row.names = FALSE)

# Check that f is a readable file
check_file_cli(f, access = "r")
check_file_exists_cli(f)

# Clearly invalid path
bad_file <- file.path(tempdir(), "does_not_exist.csv")
try(check_file_cli(bad_file, access = "r"))          # failure
```

```
try(check_file_exists_cli(bad_file))                # failure

assert_file_cli(f, access = "r")
try(assert_file_cli(bad_file, access = "r"))

assert_file_exists_cli(f)
try(assert_file_exists_cli(bad_file))

# Directories:

d <- tempdir()

# Generic directory check (using checkDirectory)
check_directory_cli(d)

# Directory must exist
check_directory_exists_cli(d)

bad_dir <- file.path(tempdir(), "no_such_dir_xyz")
try(check_directory_cli(bad_dir))                   # typically failure
try(check_directory_exists_cli(bad_dir))            # failure

assert_directory_cli(d)
try(assert_directory_cli(bad_dir))

assert_directory_exists_cli(d)
try(assert_directory_exists_cli(bad_dir))

# Output paths:

# Valid output path in an existing directory
out_file <- file.path(tempdir(), "results.rds")
check_path_for_output_cli(out_file)

# Directory part does not exist
bad_out <- file.path(tempdir(), "no_such_dir", "results.rds")
try(check_path_for_output_cli(bad_out))             # failure

assert_path_for_output_cli(out_file)
try(assert_path_for_output_cli(bad_out))

# Combine with other assertions in a pipeline:
out_file2 <- file.path(tempdir(), "summary.csv")
assert_directory_exists_cli(dirname(out_file2))
assert_path_for_output_cli(out_file2)
```

---

checkcli-names-sets *Names, sets, and class CLI assertions*

Description

These functions wrap **checkmate** checks related to names, sets, permutations, choices, and class membership, returning the check result (‘check_*_cli()‘) or raising a ‘cli::cli_abort()‘ error via [make_assertion()] (‘assert_*_cli()‘). They are useful for validating input names, restricting values to a fixed set, checking set relationships, or ensuring that objects inherit from one or more expected classes, while emitting consistent CLI-styled error messages.

Usage

```
check_names_cli(...)

assert_names_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_subset_cli(...)

assert_subset_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_permutation_cli(...)

assert_permutation_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_choice_cli(...)

assert_choice_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_set_equal_cli(...)

assert_set_equal_cli(x, y, ..., .var.name = checkmate::vname(x), add = NULL)

check_disjunct_cli(...)

assert_disjunct_cli(x, y, ..., .var.name = checkmate::vname(x), add = NULL)

check_class_cli(...)

assert_class_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_multi_class_cli(...)

assert_multi_class_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)
```

Arguments

...	Additional arguments passed on to the corresponding **checkmate** function (e.g. ‘lower‘, ‘upper‘, ‘any.missing‘, ‘all.missing‘, ‘min.len‘, ‘null.ok‘).
x	Object to be checked. This is the value whose type, length, or other properties are validated.

`.var.name` Character scalar used in error messages to refer to the checked object. Defaults to [checkmate::vname()], which tries to infer the variable name from the calling context.

`add` Optional [checkmate::AssertCollection] to which assertion failures are added instead of triggering an immediate error. Defaults to ‘NULL‘, which causes a ‘cli::cli_abort()‘ on failure.

- Names: - [check_names_cli()], [assert_names_cli()] check that object names satisfy constraints such as being non-missing, unique, or matching a pattern. - Set membership and subsets: - [check_subset_cli()], [assert_subset_cli()] ensure that all elements of ‘x‘ are contained in ‘choices‘. - [check_choice_cli()], [assert_choice_cli()] ensure that scalar ‘x‘ is one of a set of allowed values (‘choices‘). - [check_permutation_cli()], [assert_permutation_cli()] ensure that ‘x‘ is a permutation of integers ‘1:n‘. - Set relationships: - [check_set_equal_cli()], [assert_set_equal_cli()] check that two sets contain the same elements (order ignored). - [check_disjunct_cli()], [assert_disjunct_cli()] check that two sets are disjoint. - Class membership: - [check_class_cli()], [assert_class_cli()] ensure that ‘x‘ inherits from at least one of the given classes. - [check_multi_class_cli()], [assert_multi_class_cli()] are similar but designed for objects that may have multiple acceptable classes.

In all ‘assert_*_cli()‘ functions, ‘x‘ is the object being checked and ‘.var.name‘ is used only for error message construction; the return value is always ‘x‘ (invisibly) on success.

`y` Second object used in set-relationship checks. For functions such as ‘check_set_equal_cli()‘ and ‘check_disjunct_cli()‘, this is the object against which ‘x‘ is compared (e.g., another vector of labels or IDs).

Value

- ‘check_*_cli()‘ functions return ‘TRUE‘ on success or a character vector describing the failure, exactly like the corresponding **checkmate** checks. - ‘assert_*_cli()‘ functions return ‘x‘ invisibly on success and either: - raise a ‘cli::cli_abort()‘ error with bullet-style messages, or - push messages into an [checkmate::AssertCollection] if ‘add‘ is supplied.

See Also

[checkmate::check_names()], [checkmate::check_subset()], [checkmate::check_choice()], [checkmate::check_permutation()], [checkmate::check_set_equal()], [checkmate::check_disjunct()], [checkmate::check_class()], [checkmate::check_multi_class()], [make_assertion()], [assert_cli()]

Other checkCLI: checkcli-containers, checkcli-datetime-os, checkcli-files, checkcli-scalars

Examples

```
## Names:

x <- c(a = "a", b = "b")

# Check that all names are non-missing and unique
check_names_cli(x, type = "named")
```

```
y <- c("a", "b")
try(check_names_cli(y, type = "named"))        # failure: no names

assert_names_cli(x, type = "named")
try(assert_names_cli(y, type = "named"))  # cli-styled error

## Subsets and choices:

allowed <- c("rf", "xgboost", "nn")

# Subset: every element of x must be in allowed
check_subset_cli(c("rf", "nn"), choices = allowed)
try(check_subset_cli(c("rf", "svm"), choices = allowed))   # failure

assert_subset_cli(c("rf", "nn"), choices = allowed)
try(assert_subset_cli(c("rf", "svm"), choices = allowed))

# Choice: scalar x must be one of allowed
check_choice_cli("rf", choices = allowed)
try(check_choice_cli("svm", choices = allowed))            # failure

assert_choice_cli("xgboost", choices = allowed)
try(assert_choice_cli("svm", choices = allowed))

## Permutations:

# Valid permutation of 1:5
check_permutation_cli(5:1, 1:5)

# Not a permutation: duplicates or missing values
try(check_permutation_cli(c(1, 2, 2, 4, 5), 1:5))              # failure

assert_permutation_cli(c(1, 2, 3, 4), 1:4)
try(assert_permutation_cli(c(1, 3, 4, 4), 1:4))

## Set relationships:

a <- c("sp1", "sp2", "sp3")
b <- c("sp3", "sp2", "sp1")
c <- c("sp1", "sp4")

# Equality of sets (order ignored)
check_set_equal_cli(a, b)
try(check_set_equal_cli(a, c))                 # failure

assert_set_equal_cli(a, b)
try(assert_set_equal_cli(a, c))

# Disjointness
d <- c("x", "y")
e <- c("z", "w")
```

```
check_disjunct_cli(d, e)
try(check_disjunct_cli(d, c("y", "z")))        # failure: "y" is shared

assert_disjunct_cli(d, e)
try(assert_disjunct_cli(d, c("y", "z")))

## Class and multi-class:

df <- data.frame(x = 1:3)

# Single/inheritance class checks
check_class_cli(df, "data.frame")
try(check_class_cli(df, "matrix"))             # failure

assert_class_cli(df, "data.frame")
try(assert_class_cli(df, "matrix"))

# Multi-class: object must inherit from at least one allowed class
allowed_classes <- c("data.frame", "tbl_df")
check_multi_class_cli(df, allowed_classes)

# A plain numeric vector will fail this multi-class check
v <- 1:3
try(check_multi_class_cli(v, allowed_classes)) # failure

assert_multi_class_cli(df, allowed_classes)
try(assert_multi_class_cli(v, allowed_classes))
```

---

checkcli-scalars *Scalar and atomic CLI assertions*

Description

These functions wrap scalar and atomic checks from **checkmate** and either return the underlying check result (‘check_*_cli()‘) or raise a ‘cli::cli_abort()‘ error via [make_assertion()] (‘assert_*_cli()‘). They are intended for validating basic building blocks of user input (numbers, flags, strings, etc.) in a consistent way, while preserving the semantics and arguments of the underlying **checkmate** checks.

Usage

```
check_atomic_cli(...)

assert_atomic_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_atomic_vector_cli(...)

assert_atomic_vector_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)
```

```
check_scalar_cli(...)

assert_scalar_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_scalar_na_cli(...)

assert_scalar_na_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_integer_cli(...)

assert_integer_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_integerish_cli(...)

assert_integerish_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_double_cli(...)

assert_double_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_complex_cli(...)

assert_complex_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_count_cli(...)

assert_count_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_string_cli(...)

assert_string_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_flag_cli(...)

assert_flag_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_int_cli(...)

assert_int_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_numeric_cli(...)

assert_numeric_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_number_cli(...)

assert_number_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)
```

```
check_logical_cli(...)

assert_logical_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_character_cli(...)

assert_character_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_null_cli(...)

assert_null_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_true_cli(...)

assert_true_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)

check_false_cli(...)

assert_false_cli(x, ..., .var.name = checkmate::vname(x), add = NULL)
```

Arguments

`...`	Additional arguments passed on to the corresponding **checkmate** function (e.g. ‘lower‘, ‘upper‘, ‘any.missing‘, ‘all.missing‘, ‘min.len‘, ‘null.ok‘).
`x`	Object to be checked. This is the value whose type, length, or other properties are validated.
`.var.name`	Character scalar used in error messages to refer to the checked object. Defaults to [checkmate::vname()], which tries to infer the variable name from the calling context.
`add`	Optional [checkmate::AssertCollection] to which assertion failures are added instead of triggering an immediate error. Defaults to ‘NULL‘, which causes a ‘cli::cli_abort()‘ on failure. - Generic atomic containers: - [check_atomic_cli()], [assert_atomic_cli()] for atomic vectors of any storage mode. - [check_atomic_vector_cli()], [assert_atomic_vector_cli()] for atomic vectors with additional constraints (e.g., length, missingness). - Generic scalars: - [check_scalar_cli()], [assert_scalar_cli()] for length-one atomic values. - [check_scalar_na_cli()], [assert_scalar_na_cli()] for a single ‘NA‘ value. - Integer and numeric: - [check_integer_cli()], [assert_integer_cli()] for integer vectors. - [check_integerish_cli()], [assert_integerish_cli()] for integer-like numerics (e.g., ‘1‘, ‘2.0‘). - [check_int_cli()], [assert_int_cli()] for integer vectors (thin wrapper around [checkmate::check_int()]). - [check_numeric_cli()], [assert_numeric_cli()] for numeric vectors of any type. - [check_number_cli()], [assert_number_cli()] for numeric scalars (length-one). - [check_double_cli()], [assert_double_cli()] for double vectors. - [check_count_cli()], [assert_count_cli()] for non-negative integer counts. - [check_complex_cli()], [assert_complex_cli()] for complex vectors.

- Logical and character: - [check_flag_cli()], [assert_flag_cli()] for single logical flags. - [check_logical_cli()], [assert_logical_cli()] for logical vectors. - [check_true_cli()], [assert_true_cli()] for conditions that must be 'TRUE'. - [check_false_cli()], [assert_false_cli()] for conditions that must be 'FALSE'. - [check_string_cli()], [assert_string_cli()] for length-one character strings. - [check_character_cli()], [assert_character_cli()] for character vectors.

- Special values: - [check_null_cli()], [assert_null_cli()] for 'NULL' values.

In all 'assert_*_cli()' functions, 'x' is the object being checked and '.var.name' is used only for error message construction; the return value is always 'x' (invisibly) on success.

Value

- 'check_*_cli()' functions return 'TRUE' on success or a character vector describing the failure, exactly like the corresponding **checkmate** checks. - 'assert_*_cli()' functions return 'x' invisibly on success and either: - raise a 'cli::cli_abort()' error with bullet-style messages, or - push messages into an [checkmate::AssertCollection] if 'add' is supplied.

See Also

[checkmate::check_atomic()], [checkmate::check_scalar()], [checkmate::check_integer()], [checkmate::check_integerish()], [checkmate::check_int()], [checkmate::check_numeric()], [checkmate::check_number()], [checkmate::check_double()], [checkmate::check_complex()], [checkmate::check_count()], [checkmate::check_string()], [checkmate::check_flag()], [checkmate::check_logical()], [checkmate::check_character()], [checkmate::check_null()], [checkmate::check_true()], [checkmate::check_false()], [make_assertion()], [assert_cli()]

Other checkCLI: checkcli-containers, checkcli-datetime-os, checkcli-files, checkcli-names-sets

Examples

```
# Generic atomic and scalar:

# Atomic vs list
check_atomic_cli(1:3)
try(check_atomic_cli(list(1, 2)))                  # failure: not atomic

assert_atomic_cli(1:3)
try(assert_atomic_cli(list(1, 2)))

# Atomic vector vs scalar
check_atomic_vector_cli(1:3)
try(check_atomic_vector_cli(matrix(1:4, 2)))       # failure

assert_atomic_vector_cli(1:3)

# Scalars
check_scalar_cli(1L)
try(check_scalar_cli(1:3))                         # failure: length > 1

assert_scalar_cli("id")
```

```
try(assert_scalar_cli(c("a", "b")))

check_scalar_na_cli(NA)
try(check_scalar_na_cli(c(NA, NA)))                # failure

assert_scalar_na_cli(NA)

# Integer and numeric:

# Integer vectors
check_integer_cli(1:5)
try(check_integer_cli(c(1, 2, 3.5)))               # failure

assert_integer_cli(1:3)
try(assert_integer_cli(c(1, 2, 2.5)))

# Integerish (numeric but whole)
check_integerish_cli(c(1, 2, 3))
try(check_integerish_cli(c(1, 2.5)))               # failure

assert_integerish_cli(c(1, 2, 3))

# `check_int` is a thin wrapper for integer vectors
check_int_cli(1L)
try(check_int_cli(1.5))                            # failure

assert_int_cli(1L)
try(assert_int_cli(1.5))

# Numeric vs number
check_numeric_cli(c(0.1, 0.2))
try(check_numeric_cli("a"))                        # failure

assert_numeric_cli(c(1, 2, 3))

check_number_cli(3.14)
try(check_number_cli(c(1, 2)))                     # failure: not scalar

assert_number_cli(0)
try(assert_number_cli(c(0, 1)))

# Double and count
check_double_cli(c(0.1, 0.2))
try(check_double_cli("a"))                         # failure

count_vals <- c(0L, 10L)
check_count_cli(0L)
try(check_count_cli(-1L))                          # failure: negative

assert_count_cli(10L)
try(assert_count_cli(-5L))

# Complex, character, and logical:
```

```
# Complex
check_complex_cli(1 + 2i)
try(check_complex_cli(1:3))                          # failure

assert_complex_cli(1 + 2i)
try(assert_complex_cli(1:3))

# String vs character vector
check_string_cli("species_id")
try(check_string_cli(c("a", "b")))                   # failure: length > 1

assert_string_cli("ok")
try(assert_string_cli(c("a", "b")))

check_character_cli(c("a", "b"))
try(check_character_cli(1:3))                        # failure

assert_character_cli(c("a", "b"))

# Logical flags, TRUE/FALSE, and NULL:

# Single logical flag
check_flag_cli(TRUE)
try(check_flag_cli(c(TRUE, FALSE)))                  # failure

assert_flag_cli(FALSE)
try(assert_flag_cli(c(TRUE, FALSE)))

# Logical vectors
check_logical_cli(c(TRUE, FALSE, TRUE))
try(check_logical_cli(c(1, 0, 1)))                   # failure

assert_logical_cli(c(TRUE, FALSE))

# Conditions that must be TRUE / FALSE
check_true_cli(1 < 2)
try(check_true_cli(FALSE))                           # failure

assert_true_cli(1 == 1)
try(assert_true_cli(1 == 2))

check_false_cli(FALSE)
try(check_false_cli(TRUE))                           # failure

assert_false_cli(1 > 2)
try(assert_false_cli(1 < 2))

# NULL checks
check_null_cli(NULL)
try(check_null_cli(1))                               # failure

assert_null_cli(NULL)
```

```
try(assert_null_cli("not null"))
```

`fix_braced_list` *Fix braced lists in messages*

Description

Rewrite brace-enclosed lists such as `{'a','b','c'}` into `{a}, {b}, {c}` to improve how they are rendered by **cli**.

Usage

```
fix_braced_list(msg)
```

Arguments

`msg`	A single character string with a message.

Value

A modified message string.

Examples

```
fix_braced_list("allowed values: {'a','b','c'}")
```

`fmt_bullet_cli` *Format CLI bullets*

Description

Convert error messages into a named vector suitable for ‘cli::cli_abort()‘.

Usage

```
fmt_bullet_cli(res, cli_bullet = "i")
```

Arguments

`res`	A character vector of messages (typically from a ‘check_*‘ call).
`cli_bullet`	Single character bullet type (‘"i"‘, ‘"x"‘, etc.).

Value

A named character vector where names are bullet types.

Examples

```
fmt_bullet_cli("Something went wrong")
```

---

make_assertion *Make CLI-style assertion*

Description

Internal helper used by all ‘assert_*_cli()‘ functions.

Usage

```
make_assertion(x, res, var.name, collection)
```

Arguments

x	The object being checked.
res	Result of a ‘checkmate::check_*()‘ call (‘TRUE‘ or message).
var.name	Name of the variable for error messages.
collection	Optional [checkmate::AssertCollection] to collect failures.

Value

Invisibly returns ‘x‘ on success or raises a ‘cli::cli_abort()‘ error.

Examples

```
# Typically used via higher-level wrappers:
make_assertion(1L, checkmate::check_int(1L), "x", NULL)
```

---

sanitize_cli *Sanitize CLI message*

Description

Escape braces in error messages so that **cli** does not treat them as inline formatting.

Usage

```
sanitize_cli(res)
```

Arguments

res	A character vector of error messages.

Value

A character vector with braces escaped for use in ‘cli‘ messages.

Examples

```
sanitize_cli("Value {x} is invalid")
```

Index