Package 'checkthat'

May 8, 2026

Title Intuitive Unit Testing Tools for Data Manipulation

Version 0.1.0

Description Provides a lightweight data validation and testing toolkit for R. Its guiding philosophy is that adding code-based data checks to users' existing workflow should be both quick and intuitive. The suite of functions included therefore mirror the common data checks many users already perform by hand or by eye. Additionally, the 'checkthat' package is optimized to work within 'tidyverse' data manipulation pipelines.

License MIT + file LICENSE

URL https://github.com/iancero/checkthat, https://iancero.github.io/checkthat/

BugReports https://github.com/iancero/checkthat/issues

Depends R (>= 4.3)

Imports cli (>= 3.6.1), glue (>= 1.6.2), lifecycle (>= 1.0.3), purrr (>= 1.0.2), rlang (>= 1.1.1)

Suggests dplyr (>= 1.1.2), knitr (>= 1.43.0), lubridate (>= 1.9.2), rmarkdown (>= 2.23.0), testthat (>= 3.0.0), tidyr (>= 1.3.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.2.3

VignetteBuilder knitr

NeedsCompilation no

Author Ian Cero [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-2862-0450>)

Maintainer Ian Cero <ian_cero@urmc.rochester.edu>

Repository CRAN

Date/Publication 2023-09-12 20:30:05 UTC

Contents

`at_least` *Facilitate "At Least" Comparison on Logical Vectors*

Description

This function facilitates a comparison to check if at least a specified proportion or count of values in a logical vector evaluate to `TRUE`.

Usage

```
at_least(logical_vec, p = NULL, n = NULL, na.rm = FALSE)
```

Arguments

`logical_vec`	A logical vector.
`p`	Proportion value (0 to 1) to compare against.
`n`	Count value (integer) to compare against.
`na.rm`	Logical. Should missing values be removed before calculation?

Value

`TRUE` if the condition is met for at least the specified proportion or count, otherwise `FALSE`.

See Also

Other basic_quantifiers: `at_most()`, `exactly_equal()`, `less_than()`, `more_than()`

Examples

```
# Check if at least 50% of values are TRUE
at_least(c(TRUE, TRUE, FALSE), p = 0.5) # Returns TRUE
```

`at_most` *Facilitate "At Most" Comparison on Logical Vectors*

Description

This function facilitates a comparison to check if at most a specified proportion or count of values in a logical vector evaluate to `TRUE`.

Usage

```
at_most(logical_vec, p = NULL, n = NULL, na.rm = FALSE)
```

Arguments

`logical_vec`	A logical vector.
`p`	Proportion value (0 to 1) to compare against.
`n`	Count value (integer) to compare against.
`na.rm`	Logical. Should missing values be removed before calculation?

Value

`TRUE` if the condition is met for at most the specified proportion or count, otherwise `FALSE`.

See Also

Other basic_quantifiers: `at_least()`, `exactly_equal()`, `less_than()`, `more_than()`

Examples

```
# Check if at most 20% of values are TRUE
at_most(c(TRUE, FALSE, TRUE, TRUE), p = 0.2) # Returns TRUE
```

check_that *Check that assertions about a dataframe are true/false*

Description

This function allows you to test whether a set of assertions about a dataframe are true and to print the results of those tests. It is particularly useful for quality control and data validation.

Usage

```
check_that(.data, ..., print = TRUE, raise_error = TRUE, encourage = TRUE)
```

Arguments

`.data`	A dataframe to be tested.
`...`	One or more conditions to test on the dataframe. Each condition should be expressed as a logical expression that evaluates to a single `TRUE` or `FALSE` value (e.g., `all(x < 3)`, `!any(is.na(x))`).
`print`	Logical. If `TRUE`, the results of the tests will be printed.
`raise_error`	Logical. If `TRUE`, an error will be thrown if any test fails. If `FALSE`, the evaluation will continue even if tests fail. Disabling errors can sometimes be useful for debugging, but should generally be avoided in finalized checks/tests.
`encourage`	Logical. If `TRUE`, encouraging messages will be displayed for tests that pass.

Details

The `check_that()` function is designed to work with both base R's existing logical functions, as well as several new functions provided in the checkthat package (see See Also below).

In addition, it also provides a data pronoun, `.d`. This is a copy of the `.data` dataframe provided as the first argument and is useful for testing not only features of specific rows or columns, but of the entire dataframe, see examples.

Value

(invisibly) the original, unmodified `.data` dataframe.

See Also

`some_of`, `whenever`, `for_case`

Examples

```
example_data <- data.frame(x = 1:5, y = 6:10)

# Test a dataframe for specific conditions
example_data |>
  check_that(
    all(x > 0),
    !any(y < 5)
  )

# Use .d pronoun to test aspect of entire dataframe
example_data |>
  check_that(
    nrow(.d) == 5,
    "x" %in% names(.d)
  )
```

---

exactly_equal *Facilitate "Exactly Equal" Comparison on Logical Vectors*

Description

This function facilitates a comparison to check if the proportion or count of values in a logical vector is exactly equal to a specified value.

Usage

```
exactly_equal(logical_vec, p = NULL, n = NULL, na.rm = FALSE)
```

Arguments

logical_vec	A logical vector.
p	Proportion value (0 to 1) to compare against.
n	Count value (integer) to compare against.
na.rm	Logical. Should missing values be removed before calculation?

Value

TRUE if the proportion or count of values is exactly equal to the specified value, otherwise FALSE.

See Also

Other basic_quantifiers: at_least(), at_most(), less_than(), more_than()

Examples

```
# Check if all values are TRUE
exactly_equal(c(TRUE, TRUE, TRUE), p = 1.0) # Returns TRUE
```

`for_case` *Check if Logical Conditions Hold for a Specific Data Row*

Description

Designed as a helper function for `check_that()`, this function checks whether user-supplied logical conditions hold true for a specific data row.

Usage

```
for_case(case, ...)
```

Arguments

`case`	A row number or a logical vector identifying the specific data row(s) to check. If a logical vector, it must have exactly 1 TRUE element (i.e., that can be used to infer the row of interest).
`...`	A set of logical conditions to be checked.

Details

This function is useful for checking if certain logical conditions are met for a specific data row in your dataset. You can provide one or more logical conditions as arguments, and the function will evaluate them for the specified row.

If you provide a row number (`case`), the function will check the conditions for that specific row. If `case` is a logical vector, it will check the conditions for rows where `case` is TRUE. Note, when `case` is a logical vector, it must have exactly one TRUE element that can then be used to infer the row of interest. Internally, this is done with a call to `which()`.

If the specified `case` is not a valid count (i.e., a row number) or does not satisfy the condition `length(which(case)) == 1`, the function will throw an error.

Value

A logical value indicating whether ALL specified conditions hold true for the specified data row (i.e., `case`).

See Also

Other special quantifiers: `some_of()`, `whenever()`

Examples

```
# for_case is designed primarily as a helper function for check_that
sample_data <- data.frame(id = c(11, 22, 33), group = c("A", "B", "C"))

sample_data |>
  check_that(
    for_case(2, group == "B"), # case given as number
    for_case(id == 22, group == "B") # case given as logical vector
  )

# for_case will technically work with simple vectors too
backwards_letters <- rev(letters)
for_case(3, backwards_letters == "x") # TRUE
```

---

is_count *Check if a Numeric Value is a Count*

Description

This function checks if a numeric value is a count, meaning it is integer-like and non-negative.

Usage

```
is_count(x, include_zero = TRUE)
```

Arguments

x	Numeric value to check.
include_zero	Logical, whether to include zero as a valid count.

Value

TRUE if x is a count, otherwise FALSE.

See Also

is_proportion, is_integerlike, validate_count, validate_proportion

Examples

```
is_count(0) # TRUE
is_count(3) # TRUE
is_count(0, include_zero = FALSE) # FALSE
is_count(-1) # FALSE
is_count(1.5) # FALSE
```

is_integerlike *Check if a Numeric Value is Integer-like*

Description

This function checks if a numeric value is and integer-like scalar, meaning it is numeric and its length is 1.

Usage

```
is_integerlike(x)
```

Arguments

x	Numeric value to check.

Value

TRUE if x is integer-like, otherwise FALSE.

See Also

is_proportion, is_count, validate_proportion, validate_count

Examples

```
is_integerlike(3) # TRUE
is_integerlike(3.5) # FALSE
is_integerlike("3") # FALSE
is_integerlike(c(1, 2)) # FALSE
```

is_logical_vec *Check if a Vector is a Valid Logical Vector*

Description

This function checks if a given vector is a valid logical vector. A valid logical vector is one that contains only logical values (TRUE or FALSE), has a length of at least 1, and does not consist entirely of missing values (NA).

Usage

```
is_logical_vec(logical_vec)
```

Arguments

logical_vec	A vector to be evaluated.

Value

TRUE if logical_vec is a valid logical vector, otherwise FALSE.

Examples

```
# Check if a valid logical vector
is_logical_vec(c(TRUE, FALSE, TRUE)) # Returns TRUE

# Check if an empty vector
is_logical_vec(c()) # Returns FALSE

# Check if a vector with missing values
is_logical_vec(c(TRUE, FALSE, NA)) # Returns TRUE
is_logical_vec(c(NA, NA, NA)) # Returns FALSE
```

---

is_proportion *Check if a Numeric Value is a Proportion*

Description

This function checks if a numeric value is a proportion scalar, meaning it is numeric and within the range of 0 to 1 (inclusive).

Usage

```
is_proportion(x)
```

Arguments

x	Numeric value to check.

Value

TRUE if x is a proportion, otherwise FALSE.

See Also

is_integerlike, is_count, validate_proportion, validate_count

Examples

```
is_proportion(0.5) # TRUE
is_proportion(1.2) # FALSE
is_proportion(-0.2) # FALSE
```

less_than *Facilitate "Less Than" Comparison on Logical Vectors*

Description

This function facilitates a comparison to check if less than a specified proportion or count of values in a logical vector evaluate to TRUE.

Usage

```
less_than(logical_vec, p = NULL, n = NULL, na.rm = FALSE)
```

Arguments

logical_vec	A logical vector.
p	Proportion value (0 to 1) to compare against.
n	Count value (integer) to compare against.
na.rm	Logical. Should missing values be removed before calculation?

Value

TRUE if the condition is met for less than the specified proportion or count, otherwise FALSE.

See Also

Other basic_quantifiers: at_least(), at_most(), exactly_equal(), more_than()

Examples

```
# Check if less than 10% of values are TRUE
less_than(c(TRUE, FALSE, FALSE), p = 0.1) # Returns FALSE
```

more_than *Facilitate "More Than" Comparison on Logical Vectors*

Description

This function facilitates a comparison to check if more than a specified proportion or count of values in a logical vector evaluate to TRUE.

Usage

```
more_than(logical_vec, p = NULL, n = NULL, na.rm = FALSE)
```

Arguments

logical_vec	A logical vector.
p	Proportion value (0 to 1) to compare against.
n	Count value (integer) to compare against.
na.rm	Logical. Should missing values be removed before calculation?

Value

TRUE if the condition is met for more than the specified proportion or count, otherwise FALSE.

See Also

Other basic_quantifiers: at_least(), at_most(), exactly_equal(), less_than()

Examples

```
# Check if more than 70% of values are TRUE
more_than(c(TRUE, TRUE, FALSE, TRUE), p = 0.7) # Returns TRUE
```

---

prop *Calculate Proportion of TRUE Values in a Logical Vector*

Description

This function calculates the proportion of TRUE values in a logical vector.

Usage

```
prop(logical_vec, na.rm = FALSE)
```

Arguments

logical_vec	A logical vector.
na.rm	Logical. Should missing values be removed before calculation? Behaves similar to base::mean, removing missing values from both the numerator and denominator of the proportion calculation.

Value

The proportion of TRUE values in the logical vector.

Examples

```
prop(c(TRUE, TRUE, FALSE, TRUE)) # Returns 0.75
prop(c(TRUE, FALSE, TRUE, FALSE, NA), na.rm = TRUE) # Returns 0.5
```

some_of *Check if logical conditions are met some of the time in a logical vector*

Description

Designed as a helper function for `check_that()`, this function allows you to check that a certain percentage or count of TRUE values are observed in a logical vector. It is therefore a more flexible version of `all()` or `any()`.

Usage

```
some_of(logical_vec, ...)
```

Arguments

`logical_vec`	A logical vector to be checked.
`...`	A set of one or more frequency specifiers (e.g., `at_least = 5`, `at_most = .70`).

Details

This function is designed as a helper function for `check_that()`. It allows you to validate that a certain percentage or count of TRUE values are observed in a logical vector. It is therefore a more flexible version of `all()` or `any()`.

The named arguments in `...` should correspond to quantifiers (e.g., `at_least`, `at_most`) followed by a numeric value representing the criteria for that quantifier (either an integer count or proportion between zero and one). For example, `at_least = 2` checks if at least 2 TRUE values are present in `logical_vec`.

Note, specifying exactly 1 in an argument is ambiguous (e.g., `at_least = 1`). Because it could represent a count (n = 1) or a proportion (100%), this value is not allowed in `some_of()` and will throw an error. If you need to specify exactly 1 (either as a count or a proportion), please use a more specific quantifier function, such as `at_least(logical_vec, p = 1)` or `at_least(logical_vec, n = 1)`.

Value

A logical value indicating all conditions specified in `...` resolve to TRUE in the given `logical_vec`.

See Also

Other special quantifiers: `for_case()`, `whenever()`

Examples

```
logical_vec <- c(TRUE, FALSE, TRUE, FALSE, TRUE)

# Check if at least 2 TRUE values are present
some_of(logical_vec, at_least = 2) # TRUE

# Check if at most 2 TRUE values are present
some_of(logical_vec, at_most = 2) # FALSE

# Check if exactly 3 TRUE values are present
some_of(logical_vec, exactly_equal = 3) # TRUE

# Check if exactly 4 TRUE values are present
some_of(logical_vec, exactly_equal = 3) # FALSE

# Invalid usage: No specific quantifiers provided (error will be thrown)
try(some_of(logical_vec)) # Error
```

---

validate_count *Validate a Count Value*

Description

This function validates whether a numeric value is a valid count (integer of zero or greater).

Usage

```
validate_count(x, include_zero = TRUE)
```

Arguments

x	Numeric value to validate as a count.
include_zero	Logical, whether to include zero as a valid count.

Value

TRUE if x is a valid count, otherwise it throws an error.

See Also

is_count, is_proportion, validate_proportion, is_integerlike

Examples

```
validate_count(0) # TRUE
validate_count(3) # TRUE
try(validate_count(0, include_zero = FALSE)) # Error: Not a valid count
try(validate_count(-1)) # Error: Not a valid count value.
```

validate_logical_vec *Validate a Logical Vector*

Description

Validates a logical vector to ensure it meets specific criteria:

- Must have a length of at least 1.
- Must be a logical-type vector.
- If all values are NA, it will raise a warning.

Usage

```
validate_logical_vec(logical_vec)
```

Arguments

logical_vec	Logical vector to validate.

Value

TRUE if the logical vector is valid, otherwise it throws an error.

See Also

is_proportion, is_count, validate_proportion, validate_count

Examples

```
validate_logical_vec(c(TRUE, FALSE, TRUE)) # TRUE
try(validate_logical_vec(c())) # Error
validate_logical_vec(c(NA, NA)) # Warning
```

validate_proportion *Validate a Proportion Value*

Description

This function validates whether a numeric value is a valid proportion scalar (ranging from 0 to 1, inclusive).

Usage

```
validate_proportion(x)
```

Arguments

x	Numeric value to validate as a proportion.

Value

TRUE if x is a valid proportion, otherwise it throws an error.

See Also

is_proportion, is_count, validate_count, is_integerlike

Examples

```
validate_proportion(0.5) # TRUE
try(validate_proportion(1.2)) # Error
```

---

whenever *Whenever one condition is true, check other logical conditions also hold*

Description

Designed as a helper function for check_that(), this function checks that whenever a certain condition is observed, other expected conditions hold as well.

Usage

```
whenever(is_observed, then_expect, ...)
```

Arguments

is_observed	A logical vector indicating the when the observed cases of interest.
then_expect	A logical vector indicating the conditions to be checked for those observed cases in is_observed.
...	A set of qualifying logical conditions (e.g., at_least = .50) to be checked in conjunction with then_expect.

Details

This function is designed as a helper function for check_that(). It is useful for checking, whenever an event or condition of interest (is_observed) is true, that certain logical conditions (then_expect) also hold true. You can provide additional qualifiers (...) to clarify how often then_expect must resolve to TRUE.

Value

A logical value indicating whether all specified conditions in then_expect hold true, whenever is_observed is TRUE.

See Also

Other special quantifiers: `for_case()`, `some_of()`

Examples

```
# whenever() is designed to work with check_that()
df <- data.frame(x = 1:5, y = 6:10)

df |>
  check_that(
    whenever(is_observed = x > 3, then_expect = y > 8),
    whenever(x %in% 2:3, y > 6, at_least = .50) # qualifying condition
  )

# whenever() can also work outside check_that()
x <- 1:5
y <- 6:10

whenever(x > 3, y > 9, at_least = 1 / 2) # TRUE
```

Index