Package: tidyassert (via r-universe)

September 6, 2024

Title Tidy assertions

Version 0.3.4

Description A re-imagining of stopifnot() with rlang::abort().
Provides an api for simple input (and output) validation with
consistent error messages.

License MIT + file LICENSE

URL https://github.com/qfes/tidyassert

BugReports https://github.com/qfes/tidyassert/issues

Depends R (>= 3.3.0)

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

Imports cli, rlang (>= 1.0.0)

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

RemoteUrl https://github.com/qfes/tidyassert

RemoteRef HEAD

RemoteSha 20a01266d84472497e0f22c65262083c87994694

Contents

`assert` *Assert*

Description

Raises an assertion error when `any(expr)` is false.

Usage

```
assert(
  expr,
  error_message = NULL,
  error_class = NULL,
  call = rlang::caller_call(),
  env = rlang::caller_env(),
  print_expr = NULL,
  ...
)
```

Arguments

expr	<expression> a logical expression to test.
error_message	<string> a message to be displayed when assertion fails.
error_class	<character> the class name/s for the error.
call	The execution environment of a currently running function, e.g. call = caller_env(). The corresponding function call is retrieved and mentioned in error messages as the source of the error. You only need to supply call when throwing a condition from a helper function which wouldn't be relevant to mention in the message. Can also be NULL or a defused function call to respectively not display any call or hard-code a code to display.
env	<environment> the environment for substituted dots and print_expr. Has no effect if dots and print_expr are already quosures.
print_expr	<expression> a diffused expression for altering the error message. Defaults to rlang::as_quosure(substitute(expr), rlang::caller_env())
...	<any> values used in evaluating glue expressions, for the error message.

---

assert_equal *Assert equal*

Description

Raises an assertion error when !all(a == b).

Usage

```
assert_equal(
  a,
  b,
  error_message = "{.arg a} must equal {.arg b}",
  error_class = NULL
)
```

Arguments

a	<any> any value
b	<any> any value
error_message	<string> a message to be displayed when assertion fails.
error_class	<character> the class name/s for the error.

See Also

Other logical-assertions: assert_greater_equal(), assert_greater(), assert_less_equal(), assert_less(), assert_not_equal(), assert_range()

assert_greater *Assert greater*

Description

Raises an assertion error when `!all(a > b)`

Usage

```
assert_greater(
  a,
  b,
  error_message = "{.arg a} must be greater than {.arg b}",
  error_class = NULL
)
```

Arguments

a	<any> any value
b	<any> any value
error_message	<string> a message to be displayed when assertion fails.
error_class	<character> the class name/s for the error.

See Also

Other logical-assertions: `assert_equal()`, `assert_greater_equal()`, `assert_less_equal()`, `assert_less()`, `assert_not_equal()`, `assert_range()`

assert_greater_equal *Assert greater equal*

Description

Raises an assertion error when `!all(a >= b)`

Usage

```
assert_greater_equal(
  a,
  b,
  error_message = "{.arg a} must be greater than or equal {.arg b}",
  error_class = NULL
)
```

Arguments

a	<any> any value
b	<any> any value
error_message	<string> a message to be displayed when assertion fails.
error_class	<character> the class name/s for the error.

See Also

Other logical-assertions: assert_equal(), assert_greater(), assert_less_equal(), assert_less(), assert_not_equal(), assert_range()

---

assert_has_names *Assert has names*

Description

Raises an assertion error when !rlang::has_name(obj, names).

Usage

```
assert_has_names(
  obj,
  names,
  error_message = "{.arg obj} must have names {.arg names}",
  error_class = NULL
)
```

Arguments

obj	<any> any value
names	<character> a vector of names.
error_message	<string> a message to be displayed when assertion fails.
error_class	<character> the class name/s for the error.

See Also

Other attribute-assertions: assert_is_named()

assert_inherits *Assert inherits*

Description

Raises an assertion error when `!inherits(obj, class)`.

Usage

```
assert_inherits(
  obj,
  class,
  error_message = "{.arg obj} must inherit {.cls {class}}",
  error_class = NULL
)
```

Arguments

`obj`	<any> any value
`class`	<string | character> the expected class(es)
`error_message`	<string> a message to be displayed when assertion fails.
`error_class`	<character> the class name/s for the error.

See Also

Other type-assertions: `assert_is_character()`, `assert_is_double()`, `assert_is_integerish()`, `assert_is_integer()`, `assert_is_logical()`, `assert_is_numeric()`, `assert_is_raw()`, `assert_is_typeof()`

assert_is_bool *Assert is bool*

Description

Raises an assertion error when `!rlang::is_bool(obj)`.

Usage

```
assert_is_bool(
  obj,
  error_message = "{.arg obj} must be a {.cls bool}",
  error_class = NULL
)
```

Arguments

obj	<any> any value
error_message	<string> a message to be displayed when assertion fails.
error_class	<character> the class name/s for the error.

See Also

Other scalar-assertions: assert_is_scalar_character(), assert_is_scalar_double(), assert_is_scalar_integeri
assert_is_scalar_integer(), assert_is_scalar_logical(), assert_is_scalar_numeric(),
assert_is_scalar_raw(), assert_is_string()

---

assert_is_character *Assert is character*

Description

Raises an assertion error when !rlang::is_character(obj).

Usage

```
assert_is_character(
  obj,
  error_message = "{.arg obj} must be a {.cls character} vector",
  error_class = NULL
)
```

Arguments

obj	<any> any value
error_message	<string> a message to be displayed when assertion fails.
error_class	<character> the class name/s for the error.

See Also

Other type-assertions: assert_inherits(), assert_is_double(), assert_is_integerish(),
assert_is_integer(), assert_is_logical(), assert_is_numeric(), assert_is_raw(), assert_is_typeof()

assert_is_double *Assert is double*

Description

Raises an assertion error when `!rlang::is_double(obj)`.

Usage

```
assert_is_double(
  obj,
  error_message = "{.arg obj} must be a {.cls double} vector",
  error_class = NULL
)
```

Arguments

obj	<any> any value
error_message	<string> a message to be displayed when assertion fails.
error_class	<character> the class name/s for the error.

See Also

Other type-assertions: assert_inherits(), assert_is_character(), assert_is_integerish(), assert_is_integer(), assert_is_logical(), assert_is_numeric(), assert_is_raw(), assert_is_typeof()

assert_is_integer *Assert is integer*

Description

Raises an assertion error when `!rlang::is_integer(obj)`.

Usage

```
assert_is_integer(
  obj,
  error_message = "{.arg obj} must be an {.cls integer} vector",
  error_class = NULL
)
```

Arguments

obj	<any> any value
error_message	<string> a message to be displayed when assertion fails.
error_class	<character> the class name/s for the error.

See Also

Other type-assertions: assert_inherits(), assert_is_character(), assert_is_double(), assert_is_integerish(), assert_is_logical(), assert_is_numeric(), assert_is_raw(), assert_is_typeof()

`assert_is_integerish` *Assert is integerish*

Description

Raises an assertion error when `!rlang::is_integerish(obj)`.

Usage

```
assert_is_integerish(
  obj,
  error_message = "{.arg obj} must be an {.cls integerish} vector",
  error_class = NULL
)
```

Arguments

`obj`	<any> any value
`error_message`	<string> a message to be displayed when assertion fails.
`error_class`	<character> the class name/s for the error.

See Also

Other type-assertions: assert_inherits(), assert_is_character(), assert_is_double(), assert_is_integer(), assert_is_logical(), assert_is_numeric(), assert_is_raw(), assert_is_typeof()

`assert_is_logical` *Assert is logical*

Description

Raises an assertion error when `!rlang::is_logical(obj)`.

Usage

```
assert_is_logical(
  obj,
  error_message = "{.arg obj} must be a {.cls logical} vector",
  error_class = NULL
)
```

Arguments

obj	<any> any value
error_message	<string> a message to be displayed when assertion fails.
error_class	<character> the class name/s for the error.

See Also

Other type-assertions: assert_inherits(), assert_is_character(), assert_is_double(), assert_is_integerish(), assert_is_integer(), assert_is_numeric(), assert_is_raw(), assert_is_typeof()

---

assert_is_named *Assert is named*

Description

Raises an assertion error when !rlang::is_named(obj).

Usage

```
assert_is_named(
  obj,
  error_message = "{.arg obj} must be named",
  error_class = NULL
)
```

Arguments

obj	<any> any value
error_message	<string> a message to be displayed when assertion fails.
error_class	<character> the class name/s for the error.

See Also

Other attribute-assertions: assert_has_names()

assert_is_numeric *Assert is numeric*

Description

Raises an assertion error when `!is.numeric(obj)`.

Usage

```
assert_is_numeric(
  obj,
  error_message = "{.arg obj} must be a {.cls numeric} vector",
  error_class = NULL
)
```

Arguments

`obj`	<any> any value
`error_message`	<string> a message to be displayed when assertion fails.
`error_class`	<character> the class name/s for the error.

See Also

Other type-assertions: `assert_inherits()`, `assert_is_character()`, `assert_is_double()`, `assert_is_integerish()`, `assert_is_integer()`, `assert_is_logical()`, `assert_is_raw()`, `assert_is_typeof()`

assert_is_raw *Assert is raw*

Description

Raises an assertion error when `!rlang::is_raw(obj)`.

Usage

```
assert_is_raw(
  obj,
  error_message = "{.arg obj} must be a {.cls raw} vector",
  error_class = NULL
)
```

Arguments

`obj`	<any> any value
`error_message`	<string> a message to be displayed when assertion fails.
`error_class`	<character> the class name/s for the error.

See Also

Other type-assertions: assert_inherits(), assert_is_character(), assert_is_double(), assert_is_integerish(), assert_is_integer(), assert_is_logical(), assert_is_numeric(), assert_is_typeof()

`assert_is_scalar_character`

Assert is scalar character

Description

Raises an assertion error when `!rlang::is_scalar_character(obj)`.

Usage

```
assert_is_scalar_character(
  obj,
  error_message = "{.arg obj} must be a scalar {.cls character}",
  error_class = NULL
)
```

Arguments

obj	<any> any value
error_message	<string> a message to be displayed when assertion fails.
error_class	<character> the class name/s for the error.

See Also

Other scalar-assertions: assert_is_bool(), assert_is_scalar_double(), assert_is_scalar_integerish(), assert_is_scalar_integer(), assert_is_scalar_logical(), assert_is_scalar_numeric(), assert_is_scalar_raw(), assert_is_string()

`assert_is_scalar_double`

Assert is scalar double

Description

Raises an assertion error when `!rlang::is_scalar_double(obj)`.

Usage

```
assert_is_scalar_double(
  obj,
  error_message = "{.arg obj} must be a scalar {.cls double}",
  error_class = NULL
)
```

Arguments

obj	<any> any value
error_message	<string> a message to be displayed when assertion fails.
error_class	<character> the class name/s for the error.

See Also

Other scalar-assertions: assert_is_bool(), assert_is_scalar_character(), assert_is_scalar_integerish(), assert_is_scalar_integer(), assert_is_scalar_logical(), assert_is_scalar_numeric(), assert_is_scalar_raw(), assert_is_string()

---

assert_is_scalar_integer

Assert is scalar integer

Description

Raises an assertion error when !rlang::is_scalar_integer(obj).

Usage

```
assert_is_scalar_integer(
  obj,
  error_message = "{.arg obj} must be a scalar {.cls integer}",
  error_class = NULL
)
```

Arguments

obj	<any> any value
error_message	<string> a message to be displayed when assertion fails.
error_class	<character> the class name/s for the error.

See Also

Other scalar-assertions: assert_is_bool(), assert_is_scalar_character(), assert_is_scalar_double(), assert_is_scalar_integerish(), assert_is_scalar_logical(), assert_is_scalar_numeric(), assert_is_scalar_raw(), assert_is_string()

assert_is_scalar_integerish

Assert is scalar integerish

Description

Raises an assertion error when `!rlang::is_scalar_integerish(obj)`.

Usage

```
assert_is_scalar_integerish(
  obj,
  error_message = "{.arg obj} must be a scalar {.cls integerish}",
  error_class = NULL
)
```

Arguments

`obj`	<any> any value
`error_message`	<string> a message to be displayed when assertion fails.
`error_class`	<character> the class name/s for the error.

See Also

Other scalar-assertions: `assert_is_bool()`, `assert_is_scalar_character()`, `assert_is_scalar_double()`, `assert_is_scalar_integer()`, `assert_is_scalar_logical()`, `assert_is_scalar_numeric()`, `assert_is_scalar_raw()`, `assert_is_string()`

assert_is_scalar_logical

Assert is scalar logical

Description

Raises an assertion error when `!rlang::is_scalar_logical(obj)`.

Usage

```
assert_is_scalar_logical(
  obj,
  error_message = "{.arg obj} must be a scalar {.cls logical}",
  error_class = NULL
)
```

Arguments

obj	<any> any value
error_message	<string> a message to be displayed when assertion fails.
error_class	<character> the class name/s for the error.

See Also

Other scalar-assertions: assert_is_bool(), assert_is_scalar_character(), assert_is_scalar_double(), assert_is_scalar_integerish(), assert_is_scalar_integer(), assert_is_scalar_numeric(), assert_is_scalar_raw(), assert_is_string()

---

assert_is_scalar_numeric

Assert is scalar numeric

Description

Raises an assertion error when `!is.numeric(obj) || length(obj) != 1L`.

Usage

```
assert_is_scalar_numeric(
  obj,
  error_message = "{.arg obj} must be a scalar {.cls numeric}",
  error_class = NULL
)
```

Arguments

obj	<any> any value
error_message	<string> a message to be displayed when assertion fails.
error_class	<character> the class name/s for the error.

See Also

Other scalar-assertions: assert_is_bool(), assert_is_scalar_character(), assert_is_scalar_double(), assert_is_scalar_integerish(), assert_is_scalar_integer(), assert_is_scalar_logical(), assert_is_scalar_raw(), assert_is_string()

assert_is_scalar_raw *Assert is scalar raw*

Description

Raises an assertion error when `!rlang::is_scalar_raw(obj)`.

Usage

```
assert_is_scalar_raw(
  obj,
  error_message = "{.arg obj} must be a scalar {.cls raw}",
  error_class = NULL
)
```

Arguments

obj	<any> any value
error_message	<string> a message to be displayed when assertion fails.
error_class	<character> the class name/s for the error.

See Also

Other scalar-assertions: `assert_is_bool()`, `assert_is_scalar_character()`, `assert_is_scalar_double()`, `assert_is_scalar_integerish()`, `assert_is_scalar_integer()`, `assert_is_scalar_logical()`, `assert_is_scalar_numeric()`, `assert_is_string()`

assert_is_string *Assert is string*

Description

Raises an assertion error when `!rlang::is_string(obj)`.

Usage

```
assert_is_string(
  obj,
  error_message = "{.arg obj} must be a {.cls string}",
  error_class = NULL
)
```

Arguments

obj	<any> any value
error_message	<string> a message to be displayed when assertion fails.
error_class	<character> the class name/s for the error.

See Also

Other scalar-assertions: assert_is_bool(), assert_is_scalar_character(), assert_is_scalar_double(), assert_is_scalar_integerish(), assert_is_scalar_integer(), assert_is_scalar_logical(), assert_is_scalar_numeric(), assert_is_scalar_raw()

---

assert_is_typeof *Assert typeof*

Description

Raises an assertion error when typeof(obj) != type.

Usage

```
assert_is_typeof(
  obj,
  type,
  error_message = "{.arg obj} must be of type {.cls {type}}",
  error_class = NULL
)
```

Arguments

obj	<any> any value
type	<string> the expected type
error_message	<string> a message to be displayed when assertion fails.
error_class	<character> the class name/s for the error.

See Also

Other type-assertions: assert_inherits(), assert_is_character(), assert_is_double(), assert_is_integerish(), assert_is_integer(), assert_is_logical(), assert_is_numeric(), assert_is_raw()

assert_less *Assert less*

Description

Raises an assertion error when `!all(a < b)`

Usage

```
assert_less(
  a,
  b,
  error_message = "{.arg a} must be less than {.arg b}",
  error_class = NULL
)
```

Arguments

a	<any> any value
b	<any> any value
error_message	<string> a message to be displayed when assertion fails.
error_class	<character> the class name/s for the error.

See Also

Other logical-assertions: `assert_equal()`, `assert_greater_equal()`, `assert_greater()`, `assert_less_equal()`, `assert_not_equal()`, `assert_range()`

assert_less_equal *Assert less equal*

Description

Raises an assertion error when `!all(a <= b)`

Usage

```
assert_less_equal(
  a,
  b,
  error_message = "{.arg a} must be less than or equal {.arg b}",
  error_class = NULL
)
```

Arguments

a	<any> any value
b	<any> any value
error_message	<string> a message to be displayed when assertion fails.
error_class	<character> the class name/s for the error.

See Also

Other logical-assertions: assert_equal(), assert_greater_equal(), assert_greater(), assert_less(), assert_not_equal(), assert_range()

---

assert_not_equal *Assert not equal*

Description

Raises an assertion error when !all(a != b)

Usage

```
assert_not_equal(
  a,
  b,
  error_message = "{.arg a} must not equal {.arg b}",
  error_class = NULL
)
```

Arguments

a	<any> any value
b	<any> any value
error_message	<string> a message to be displayed when assertion fails.
error_class	<character> the class name/s for the error.

See Also

Other logical-assertions: assert_equal(), assert_greater_equal(), assert_greater(), assert_less_equal(), assert_less(), assert_range()

assert_not_na *Assert not na*

Description

Raises an assertion error when `anyNA(obj)`

Usage

```
assert_not_na(
  obj,
  error_message = "{.arg obj} must be not contain {.code NA}",
  error_class = NULL
)
```

Arguments

`obj`	<any> any object
`error_message`	<string> a message to be displayed when assertion fails.
`error_class`	<character> the class name/s for the error.

See Also

Other assertions: `assert_not_null()`

assert_not_null *Assert not null*

Description

Raises an assertion error when `is.null(obj)`

Usage

```
assert_not_null(
  obj,
  error_message = "{.arg obj} must be not {.code NULL}",
  error_class = NULL
)
```

Arguments

`obj`	<any> any object
`error_message`	<string> a message to be displayed when assertion fails.
`error_class`	<character> the class name/s for the error.

See Also

Other assertions: assert_not_na()

---

assert_range *Assert range*

Description

Raises an assertion error when !all(a >= min & a <= max)

Usage

```
assert_range(
  obj,
  min,
  max,
  error_message = "{.arg obj} must be within [{.arg min}, {.arg max}]",
  error_class = NULL
)
```

Arguments

obj	<any> any object
min	<any> the minimum value
max	<any> the maximum value
error_message	<string> a message to be displayed when assertion fails.
error_class	<character> the class name/s for the error.

See Also

Other logical-assertions: assert_equal(), assert_greater_equal(), assert_greater(), assert_less_equal(), assert_less(), assert_not_equal()

---

warn_if_not *Assert (soft)*

Description

Raises an assertion warning when any(expr) is false.

Raises an assertion warning when all(expr) is true.

Usage

```
warn_if_not(
  expr,
  warn_message = NULL,
  warn_class = NULL,
  call = rlang::caller_call(),
  env = rlang::caller_env(),
  print_expr = NULL,
  ...
)

warn_if(
  expr,
  warn_message = NULL,
  warn_class = NULL,
  call = rlang::caller_call(),
  env = rlang::caller_env(2L),
  print_expr = NULL,
  ...
)
```

Arguments

`expr`	`<expression>` a logical expression to test.
`warn_message`	`<string>` a message to be displayed when assertion fails.
`warn_class`	`<character>` the class name/s for the warning.
`call`	The execution environment of a currently running function, e.g. `call = caller_env()`. The corresponding function call is retrieved and mentioned in error messages as the source of the error. You only need to supply `call` when throwing a condition from a helper function which wouldn't be relevant to mention in the message. Can also be `NULL` or a defused function call to respectively not display any call or hard-code a code to display.
`env`	`<environment>` the environment for substituted `dots` and `print_expr`. Has no effect if `dots` and `print_expr` are already quosures.
`print_expr`	`<expression>` a diffused expression for altering the error message. Defaults to `rlang::as_quosure(substitute(expr), rlang::caller_env())`
`...`	`<any>` values used in evaluating glue expressions, for the error message.

Index