Package 'stbl'

May 23, 2024

Title Stabilize Function Arguments

Version 0.1.1

Description A set of consistent, opinionated functions to quickly check function arguments, coerce them to the desired configuration, or deliver informative error messages when that is not possible.

License MIT + file LICENSE

URL https://github.com/jonthegeek/stbl, https://jonthegeek.github.io/stbl/

BugReports https://github.com/jonthegeek/stbl/issues

Imports cli, glue, rlang (>= 1.1.0), vctrs

Suggests stringi, testthat (>= 3.0.0)

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

RoxygenNote 7.3.1

NeedsCompilation no

Author Jon Harmon [aut, cre, cph] (<https://orcid.org/0000-0003-4781-4346>)

Maintainer Jon Harmon <jonthegeek@gmail.com>

Repository CRAN

Date/Publication 2024-05-23 15:40:02 UTC

R topics documented:

object_type *Identify the class, type, etc of an object*

Description

Extract the class (or type) of an object for use in error messages.

Usage

```
object_type(x)
```

Arguments

x	An object to test.

Value

A length-1 character vector describing the class of the object.

Examples

```
object_type("a")
object_type(1L)
object_type(1.1)
object_type(mtcars)
object_type(rlang::quo(something))
```

stabilize_arg *Ensure an argument meets expectations*

Description

`stabilize_arg()` is used by other functions such as `stabilize_int()`. Use `stabilize_arg()` if the type-specific functions will not work for your use case, but you would still like to check things like size or whether the argument is NULL.

`stabilize_arg_scalar()` is optimized to check for length-1 vectors.

Usage

```
stabilize_arg(
  x,
  ...,
  allow_null = TRUE,
  allow_na = TRUE,
  min_size = NULL,
  max_size = NULL,
```

```
  x_arg = caller_arg(x),
  call = caller_env(),
  x_class = object_type(x)
)

stabilize_arg_scalar(
  x,
  ...,
  allow_null = TRUE,
  allow_zero_length = TRUE,
  allow_na = TRUE,
  x_arg = caller_arg(x),
  call = caller_env(),
  x_class = object_type(x)
)
```

Arguments

x	The argument to stabilize.
...	These dots are for future extensions and should be empty.
allow_null	Logical. Is NULL an acceptable value?
allow_na	Logical. Are NA values ok?
min_size	Integer. The minimum size of the object. Object size will be tested using vctrs::vec_size().
max_size	Integer. The maximum size of the object. Object size will be tested using vctrs::vec_size().
x_arg	Character. An argument name for x. The automatic value will work in most cases, or pass it through from higher-level functions to make error messages clearer in unexported functions.
call	The execution environment of the call. See the call argument of rlang::abort() for more information.
x_class	Character. The class name of x to use in error messages. Use this if you remove a special class from x before checking its coercion, but want the error message to match the original class.
allow_zero_length	Logical. Are zero-length vectors acceptable?

Value

x, unless one of the checks fails.

Examples

```
wrapper <- function(this_arg, ...) {
  stabilize_arg(this_arg, ...)
}
wrapper(1)
```

```
wrapper(NULL)
wrapper(NA)
try(wrapper(NULL, allow_null = FALSE))
try(wrapper(NA, allow_na = FALSE))
try(wrapper(1, min_size = 2))
try(wrapper(1:10, max_size = 5))
stabilize_arg_scalar("a")
stabilize_arg_scalar(1L)
try(stabilize_arg_scalar(1:10))
```

`stabilize_chr` *Ensure a character argument meets expectations*

Description

`to_chr()` checks whether an argument can be coerced to character without losing information, returning it silently if so. Otherwise an informative error message is signaled.

`stabilize_chr()` can check more details about the argument, but is slower than `to_chr()`.

`stabilize_chr_scalar()` and `to_chr_scalar()` are optimized to check for length-1 character vectors.

Usage

```
stabilize_chr(
  x,
  ...,
  allow_null = TRUE,
  allow_na = TRUE,
  min_size = NULL,
  max_size = NULL,
  regex = NULL,
  x_arg = caller_arg(x),
  call = caller_env(),
  x_class = object_type(x)
)

stabilize_chr_scalar(
  x,
  ...,
  allow_null = TRUE,
  allow_zero_length = TRUE,
  allow_na = TRUE,
  regex = NULL,
  x_arg = caller_arg(x),
  call = caller_env(),
  x_class = object_type(x)
)
```

```
to_chr(
  x,
  allow_null = TRUE,
  x_arg = caller_arg(x),
  call = caller_env(),
  x_class = object_type(x)
)

to_chr_scalar(
  x,
  allow_null = TRUE,
  allow_zero_length = TRUE,
  x_arg = caller_arg(x),
  call = caller_env(),
  x_class = object_type(x)
)
```

Arguments

x	The argument to stabilize.
...	These dots are for future extensions and should be empty.
allow_null	Logical. Is NULL an acceptable value?
allow_na	Logical. Are NA values ok?
min_size	Integer. The minimum size of the object. Object size will be tested using vctrs::vec_size().
max_size	Integer. The maximum size of the object. Object size will be tested using vctrs::vec_size().
regex	Character scalar. An optional regex pattern to compare the value(s) of x against. If a complex regex pattern throws an error, try installing the stringi package with install.packages("stringi").
x_arg	Character. An argument name for x. The automatic value will work in most cases, or pass it through from higher-level functions to make error messages clearer in unexported functions.
call	The execution environment of the call. See the call argument of rlang::abort() for more information.
x_class	Character. The class name of x to use in error messages. Use this if you remove a special class from x before checking its coercion, but want the error message to match the original class.
allow_zero_length	Logical. Are zero-length vectors acceptable?

Details

These functions have two important differences from base::as.character():

- `lists` and `data.frames` are *not* coerced to character. In base R, such objects are coerced to character representations of their elements. For example, `as.character(list(1:3))` returns "1:10". In the unlikely event that this is the expected behavior, use `as.character()` instead.
- `NULL` values can be rejected as part of the call to this function (with `allow_null = FALSE`).

Value

The argument as a character vector.

Examples

```
to_chr("a")
to_chr(letters)
to_chr(1:10)
to_chr(1 + 0i)
to_chr(NULL)
try(to_chr(NULL, allow_null = FALSE))

to_chr_scalar("a")
try(to_chr_scalar(letters))

stabilize_chr(letters)
stabilize_chr(1:10)
stabilize_chr(NULL)
try(stabilize_chr(NULL, allow_null = FALSE))
try(stabilize_chr(c("a", NA), allow_na = FALSE))
try(stabilize_chr(letters, min_size = 50))
try(stabilize_chr(letters, max_size = 20))
try(stabilize_chr(c("hide", "find", "find", "hide"), regex = "hide"))

stabilize_chr_scalar(TRUE)
stabilize_chr_scalar("TRUE")
try(stabilize_chr_scalar(c(TRUE, FALSE, TRUE)))
stabilize_chr_scalar(NULL)
try(stabilize_chr_scalar(NULL, allow_null = FALSE))
```

---

stabilize_fct *Ensure a factor argument meets expectations*

Description

`to_fct()` checks whether an argument can be coerced to a factor without losing information, returning it silently if so. Otherwise an informative error message is signaled.

`stabilize_fct()` can check more details about the argument, but is slower than `to_fct()`.

`stabilize_fct_scalar()` and `to_fct_scalar()` are optimized to check for length-1 factors.

Usage

```
stabilize_fct(
  x,
  ...,
  allow_null = TRUE,
  allow_na = TRUE,
  min_size = NULL,
  max_size = NULL,
  levels = NULL,
  to_na = character(),
  x_arg = caller_arg(x),
  call = caller_env(),
  x_class = object_type(x)
)

stabilize_fct_scalar(
  x,
  ...,
  allow_null = TRUE,
  allow_zero_length = TRUE,
  allow_na = TRUE,
  levels = NULL,
  to_na = character(),
  x_arg = caller_arg(x),
  call = caller_env(),
  x_class = object_type(x)
)

to_fct(
  x,
  allow_null = TRUE,
  levels = NULL,
  to_na = character(),
  x_arg = caller_arg(x),
  call = caller_env(),
  x_class = object_type(x)
)

to_fct_scalar(
  x,
  allow_null = TRUE,
  allow_zero_length = TRUE,
  levels = NULL,
  to_na = character(),
  x_arg = caller_arg(x),
  call = caller_env(),
  x_class = object_type(x)
)
```

Arguments

x	The argument to stabilize.
...	These dots are for future extensions and should be empty.
allow_null	Logical. Is NULL an acceptable value?
allow_na	Logical. Are NA values ok?
min_size	Integer. The minimum size of the object. Object size will be tested using vctrs::vec_size().
max_size	Integer. The maximum size of the object. Object size will be tested using vctrs::vec_size().
levels	Character. Expected levels. If NULL (default), the levels will be computed by base::factor().
to_na	Character. Values to coerce to NA.
x_arg	Character. An argument name for x. The automatic value will work in most cases, or pass it through from higher-level functions to make error messages clearer in unexported functions.
call	The execution environment of the call. See the call argument of rlang::abort() for more information.
x_class	Character. The class name of x to use in error messages. Use this if you remove a special class from x before checking its coercion, but want the error message to match the original class.
allow_zero_length	Logical. Are zero-length vectors acceptable?

Details

These functions have important differences from base::as.factor() and base::factor():

- Values are never silently coerced to NA unless they are explicitly supplied in the to_na argument.
- NULL values can be rejected as part of the call to this function (with allow_null = FALSE).

Value

The argument as a factor.

Examples

```
to_fct("a")
to_fct(1:10)
to_fct(NULL)
try(to_fct(letters[1:5], levels = c("a", "c"), to_na = "b"))

to_fct_scalar("a")
try(to_fct_scalar(letters))

stabilize_fct(letters)
```

```
try(stabilize_fct(NULL, allow_null = FALSE))
try(stabilize_fct(c("a", NA), allow_na = FALSE))
try(stabilize_fct(c("a", "b", "c"), min_size = 5))
try(stabilize_fct(c("a", "b", "c"), max_size = 2))

stabilize_fct_scalar("a")
try(stabilize_fct_scalar(letters))
try(stabilize_fct_scalar("c", levels = c("a", "b")))
```

`stabilize_int` *Ensure an integer argument meets expectations*

Description

`to_int()` checks whether an argument can be coerced to integer without losing information, returning it silently if so. Otherwise an informative error message is signaled.

`stabilize_int()` can check more details about the argument, but is slower than `to_int()`.

`stabilize_int_scalar()` and `to_int_scalar()` are optimized to check for length-1 integer vectors.

Usage

```
stabilize_int(
  x,
  ...,
  allow_null = TRUE,
  allow_na = TRUE,
  coerce_character = TRUE,
  coerce_factor = TRUE,
  min_size = NULL,
  max_size = NULL,
  min_value = NULL,
  max_value = NULL,
  x_arg = caller_arg(x),
  call = caller_env(),
  x_class = object_type(x)
)

stabilize_int_scalar(
  x,
  ...,
  allow_null = TRUE,
  allow_zero_length = TRUE,
  allow_na = TRUE,
  coerce_character = TRUE,
  coerce_factor = TRUE,
  min_value = NULL,
```

```
  max_value = NULL,
  x_arg = caller_arg(x),
  call = caller_env(),
  x_class = object_type(x)
)

to_int(
  x,
  allow_null = TRUE,
  coerce_character = TRUE,
  coerce_factor = TRUE,
  x_arg = caller_arg(x),
  call = caller_env(),
  x_class = object_type(x)
)

to_int_scalar(
  x,
  allow_null = TRUE,
  allow_zero_length = TRUE,
  coerce_character = TRUE,
  coerce_factor = TRUE,
  x_arg = caller_arg(x),
  call = caller_env(),
  x_class = object_type(x)
)
```

Arguments

x	The argument to stabilize.
...	These dots are for future extensions and should be empty.
allow_null	Logical. Is NULL an acceptable value?
allow_na	Logical. Are NA values ok?
coerce_character	Logical. Should character vectors such as "1" and "2.0" be coerced to integer?
coerce_factor	Logical. Should factors with values such as "1" and "2.0" be coerced to integer? Note that this function uses the character value from the factor, while `as.integer()` uses the integer index of the factor.
min_size	Integer. The minimum size of the object. Object size will be tested using `vctrs::vec_size()`.
max_size	Integer. The maximum size of the object. Object size will be tested using `vctrs::vec_size()`.
min_value	Integer scalar. The lowest allowed value for `x`. If `NULL` (default) values are not checked.
max_value	Integer scalar. The highest allowed value for `x`. If `NULL` (default) values are not checked.

x_arg — Character. An argument name for x. The automatic value will work in most cases, or pass it through from higher-level functions to make error messages clearer in unexported functions.

call — The execution environment of the call. See the call argument of rlang::abort() for more information.

x_class — Character. The class name of x to use in error messages. Use this if you remove a special class from x before checking its coercion, but want the error message to match the original class.

allow_zero_length — Logical. Are zero-length vectors acceptable?

Value

The argument as an integer.

Examples

```
to_int(1:10)
to_int("1")
to_int(1 + 0i)
to_int(NULL)
try(to_int(c(1, 2, 3.1, 4, 5.2)))
try(to_int("1", coerce_character = FALSE))
try(to_int(c("1", "2", "3.1", "4", "5.2")))

to_int_scalar("1")
try(to_int_scalar(1:10))

stabilize_int(1:10)
stabilize_int("1")
stabilize_int(1 + 0i)
stabilize_int(NULL)
try(stabilize_int(NULL, allow_null = FALSE))
try(stabilize_int(c(1, NA), allow_na = FALSE))
try(stabilize_int(letters))
try(stabilize_int("1", coerce_character = FALSE))
try(stabilize_int(factor(c("1", "a"))))
try(stabilize_int(factor("1"), coerce_factor = FALSE))
try(stabilize_int(1:10, min_value = 3))
try(stabilize_int(1:10, max_value = 7))

stabilize_int_scalar(1L)
stabilize_int_scalar("1")
try(stabilize_int_scalar(1:10))
stabilize_int_scalar(NULL)
try(stabilize_int_scalar(NULL, allow_null = FALSE))
```

stabilize_lgl *Ensure a logical argument meets expectations*

Description

`to_lgl()` checks whether an argument can be coerced to logical without losing information, returning it silently if so. Otherwise an informative error message is signaled.

`stabilize_lgl()` can check more details about the argument, but is slower than `to_lgl()`.

`stabilize_lgl_scalar()` and `to_lgl_scalar()` are optimized to check for length-1 logical vectors.

Usage

```
stabilize_lgl(
  x,
  ...,
  allow_null = TRUE,
  allow_na = TRUE,
  min_size = NULL,
  max_size = NULL,
  x_arg = caller_arg(x),
  call = caller_env(),
  x_class = object_type(x)
)

stabilize_lgl_scalar(
  x,
  ...,
  allow_null = TRUE,
  allow_zero_length = TRUE,
  allow_na = TRUE,
  x_arg = caller_arg(x),
  call = caller_env(),
  x_class = object_type(x)
)

to_lgl(
  x,
  allow_null = TRUE,
  x_arg = caller_arg(x),
  call = caller_env(),
  x_class = object_type(x)
)

to_lgl_scalar(
  x,
```

```
  allow_null = TRUE,
  allow_zero_length = TRUE,
  x_arg = caller_arg(x),
  call = caller_env(),
  x_class = object_type(x)
)
```

Arguments

x	The argument to stabilize.
...	These dots are for future extensions and should be empty.
allow_null	Logical. Is NULL an acceptable value?
allow_na	Logical. Are NA values ok?
min_size	Integer. The minimum size of the object. Object size will be tested using vctrs::vec_size().
max_size	Integer. The maximum size of the object. Object size will be tested using vctrs::vec_size().
x_arg	Character. An argument name for x. The automatic value will work in most cases, or pass it through from higher-level functions to make error messages clearer in unexported functions.
call	The execution environment of the call. See the call argument of rlang::abort() for more information.
x_class	Character. The class name of x to use in error messages. Use this if you remove a special class from x before checking its coercion, but want the error message to match the original class.
allow_zero_length	Logical. Are zero-length vectors acceptable?

Value

The argument as a logical vector.

Examples

```
to_lgl(TRUE)
to_lgl("TRUE")
to_lgl(1:10)
to_lgl(NULL)
try(to_lgl(NULL, allow_null = FALSE))
try(to_lgl(letters))
try(to_lgl(list(TRUE)))

to_lgl_scalar("TRUE")
try(to_lgl_scalar(c(TRUE, FALSE)))

stabilize_lgl(c(TRUE, FALSE, TRUE))
stabilize_lgl("true")
stabilize_lgl(NULL)
```

```
try(stabilize_lgl(NULL, allow_null = FALSE))
try(stabilize_lgl(c(TRUE, NA), allow_na = FALSE))
try(stabilize_lgl(letters))
try(stabilize_lgl(c(TRUE, FALSE, TRUE), min_size = 5))
try(stabilize_lgl(c(TRUE, FALSE, TRUE), max_size = 2))

stabilize_lgl_scalar(TRUE)
stabilize_lgl_scalar("TRUE")
try(stabilize_lgl_scalar(c(TRUE, FALSE, TRUE)))
stabilize_lgl_scalar(NULL)
try(stabilize_lgl_scalar(NULL, allow_null = FALSE))
```

Index