Package 'quickcheck'

March 17, 2022

Title Property Based Testing

Version 0.1.2

Description Property based testing, inspired by
the original 'QuickCheck'. This package builds on
the property based testing framework provided by
'hedgehog' and is designed to seamlessly integrate with
'testthat'.

License MIT + file LICENSE

URL https://github.com/armcn/quickcheck,
https://armcn.github.io/quickcheck/

BugReports https://github.com/armcn/quickcheck/issues

Encoding UTF-8

RoxygenNote 7.1.1

Imports testthat (>= 3.0.0), hedgehog, purrr, tibble, data.table, hms,
stats, magrittr

Suggests knitr, rmarkdown, covr, dplyr

Config/testthat/edition 3

NeedsCompilation no

Author Andrew McNeil [aut, cre]

Maintainer Andrew McNeil <andrew.richard.mcneil@gmail.com>

Repository CRAN

Date/Publication 2022-03-17 13:10:02 UTC

R topics documented:

anything 2
any_atomic 3
any_data.table 4
any_data_frame 4
any_flat_homogeneous_list 5

any_flat_list . . . 6
any_list . . . 6
any_tibble . . . 7
any_undefined . . . 8
any_vector . . . 8
as_hedgehog . . . 9
character_ . . . 9
constant . . . 10
data.table_ . . . 11
data.table_of . . . 11
data_frame_ . . . 12
data_frame_of . . . 13
date_ . . . 14
double_ . . . 15
equal_length . . . 17
factor_ . . . 18
flat_list_of . . . 18
for_all . . . 19
from_hedgehog . . . 20
hms_ . . . 21
integer_ . . . 22
list_ . . . 23
list_of . . . 23
logical_ . . . 24
numeric_ . . . 24
one_of . . . 25
posixct_ . . . 26
repeat_test . . . 27
show_example . . . 28
tibble_ . . . 28
tibble_of . . . 29

Index **30**

`anything` *Any R object generator*

Description

Generate any R object. This doesn't actually generate any possible object, just the most common ones, namely atomic vectors, lists, data.frames, tibbles, data.tables, and undefined values like `NA`, `NULL`, `Inf`, and `NaN`.

Usage

```
anything(any_empty = TRUE, any_undefined = TRUE)
```

Arguments

any_empty	Whether empty vectors or data frames should be allowed.
any_undefined	Whether undefined values should be allowed.

Value

A quickcheck_generator object.

Examples

```
anything() %>% show_example()
```

---

any_atomic *Any atomic vector generator*

Description

Generate vectors of integer, double, character, logical, date, POSIXct, hms, or factors.

Usage

```
any_atomic(len = c(1L, 10L), any_na = FALSE)
```

Arguments

len	Length of the generated vectors. If len is a single number all vectors will have this length. If len is a numeric vector of length 2 it will produce vectors with lengths between a minimum and maximum, inclusive. For example len = c(1L,10L) would produce vectors with lengths between 1 and 10. To produce empty vectors set len = 0L or a range like len = c(0L,10L).
any_na	Whether NA values should be allowed.

Value

A quickcheck_generator object.

Examples

```
any_atomic() %>% show_example()
any_atomic(len = 10L, any_na = TRUE) %>% show_example()
```

any_data.table *Any data.table generator*

Description

Generate data.tables.

Usage

```
any_data.table(rows = c(1L, 10L), cols = c(1L, 10L), any_na = FALSE)
```

Arguments

rows	Number of rows of the generated data frame. If rows is a single number all data frames will have this number of rows. If rows is a numeric vector of length 2 it will produce data frames with rows between a minimum and maximum, inclusive. For example rows = c(1L,10L) would produce data frames with rows between 1 and 10. To produce empty tibbles set rows = 0L or a range like rows = c(0L,10L).
cols	Number of columns of the generated data frame. If cols is a single number all data frames will have this number of columns. If cols is a numeric vector of length 2 it will produce data frames with columns between a minimum and maximum, inclusive. For example cols = c(1L,10L) would produce data frames with columns between 1 and 10. To produce empty tibbles set cols = 0L or a range like cols = c(0L,10L).
any_na	Whether NA values should be allowed.

Value

A quickcheck_generator object.

Examples

```
any_data.table(rows = 3L, cols = 3L) %>% show_example()
```

any_data_frame *Any data frame generator*

Description

Generate data.frames.

Usage

```
any_data_frame(rows = c(1L, 10L), cols = c(1L, 10L), any_na = FALSE)
```

Arguments

rows	Number of rows of the generated data frame. If rows is a single number all data frames will have this number of rows. If rows is a numeric vector of length 2 it will produce data frames with rows between a minimum and maximum, inclusive. For example rows = c(1L,10L) would produce data frames with rows between 1 and 10. To produce empty tibbles set rows = 0L or a range like rows = c(0L,10L).
cols	Number of columns of the generated data frame. If cols is a single number all data frames will have this number of columns. If cols is a numeric vector of length 2 it will produce data frames with columns between a minimum and maximum, inclusive. For example cols = c(1L,10L) would produce data frames with columns between 1 and 10. To produce empty tibbles set cols = 0L or a range like cols = c(0L,10L).
any_na	Whether NA values should be allowed.

Value

A quickcheck_generator object.

Examples

```
any_data_frame(rows = 3L, cols = 3L) %>% show_example()
```

---

any_flat_homogeneous_list

Any flat homogeneous list generator

Description

Generate lists in which each element is an atomic scalar of the same class.

Usage

```
any_flat_homogeneous_list(len = c(1L, 10L), any_na = FALSE)
```

Arguments

len	Length of the generated vectors. If len is a single number all vectors will have this length. If len is a numeric vector of length 2 it will produce vectors with lengths between a minimum and maximum, inclusive. For example len = c(1L,10L) would produce vectors with lengths between 1 and 10. To produce empty vectors set len = 0L or a range like len = c(0L,10L).
any_na	Whether NA values should be allowed.

Value

A quickcheck_generator object.

Examples

```
any_flat_homogeneous_list() %>% show_example()
any_flat_homogeneous_list(len = 10L, any_na = TRUE) %>% show_example()
```

---

any_flat_list *Any flat list generator*

Description

Generate lists in which each element is an atomic scalar.

Usage

```
any_flat_list(len = c(1L, 10L), any_na = FALSE)
```

Arguments

len	Length of the generated vectors. If `len` is a single number all vectors will have this length. If `len` is a numeric vector of length 2 it will produce vectors with lengths between a minimum and maximum, inclusive. For example `len = c(1L,10L)` would produce vectors with lengths between 1 and 10. To produce empty vectors set `len = 0L` or a range like `len = c(0L,10L)`.
any_na	Whether `NA` values should be allowed.

Value

A `quickcheck_generator` object.

Examples

```
any_flat_list() %>% show_example()
any_flat_list(len = 10L, any_na = TRUE) %>% show_example()
```

---

any_list *Any list generator*

Description

Generate lists containing lists or atomic vectors.

Usage

```
any_list(len = c(1L, 10L), any_na = FALSE)
```

Arguments

len	Length of the generated vectors. If len is a single number all vectors will have this length. If len is a numeric vector of length 2 it will produce vectors with lengths between a minimum and maximum, inclusive. For example len = c(1L,10L) would produce vectors with lengths between 1 and 10. To produce empty vectors set len = 0L or a range like len = c(0L,10L).
any_na	Whether NA values should be allowed.

Value

A quickcheck_generator object.

Examples

```
any_list() %>% show_example()
any_list(len = 10L, any_na = TRUE) %>% show_example()
```

---

any_tibble *Any tibble generator*

Description

Generate tibbles.

Usage

```
any_tibble(rows = c(1L, 10L), cols = c(1L, 10L), any_na = FALSE)
```

Arguments

rows	Number of rows of the generated data frame. If rows is a single number all data frames will have this number of rows. If rows is a numeric vector of length 2 it will produce data frames with rows between a minimum and maximum, inclusive. For example rows = c(1L,10L) would produce data frames with rows between 1 and 10. To produce empty tibbles set rows = 0L or a range like rows = c(0L,10L).
cols	Number of columns of the generated data frame. If cols is a single number all data frames will have this number of columns. If cols is a numeric vector of length 2 it will produce data frames with columns between a minimum and maximum, inclusive. For example cols = c(1L,10L) would produce data frames with columns between 1 and 10. To produce empty tibbles set cols = 0L or a range like cols = c(0L,10L).
any_na	Whether NA values should be allowed.

Value

A quickcheck_generator object.

Examples

```
any_tibble(rows = 3L, cols = 3L) %>% show_example()
```

`any_undefined` *Any undefined value generator*

Description

Generate undefined values. In this case undefined values include `NA`, `NA_integer_`, `NA_real_`, `NA_character_`, `NA_complex_`, `NULL`, `-Inf`, `Inf`, and `NaN`. Values generated are always scalars.

Usage

```
any_undefined()
```

Value

A `quickcheck_generator` object.

Examples

```
any_undefined() %>% show_example()
```

`any_vector` *Any vector generator*

Description

Generate atomic vectors or lists.

Usage

```
any_vector(len = c(1L, 10L), any_na = FALSE)
```

Arguments

`len`	Length of the generated vectors. If `len` is a single number all vectors will have this length. If `len` is a numeric vector of length 2 it will produce vectors with lengths between a minimum and maximum, inclusive. For example `len = c(1L,10L)` would produce vectors with lengths between 1 and 10. To produce empty vectors set `len = 0L` or a range like `len = c(0L,10L)`.
`any_na`	Whether `NA` values should be allowed.

Value

A `quickcheck_generator` object.

Examples

```
any_vector() %>% show_example()
any_vector(len = 10L, any_na = TRUE) %>% show_example()
```

---

as_hedgehog *Convert a quickcheck generator to a hedgehog generator*

Description

Convert a quickcheck generator to a hedgehog generator

Usage

```
as_hedgehog(generator)
```

Arguments

generator	A quickcheck_generator object.

Value

A quickcheck_generator object.

Examples

```
is_even <-
  function(a) a %% 2L == 0L
gen_powers_of_two <-
  integer_bounded(1L, 10L, len = 1L) %>%
    as_hedgehog() %>%
    hedgehog::gen.with(function(a) 2 ^ a)
for_all(
  a = from_hedgehog(gen_powers_of_two),
  property = function(a) is_even(a) %>% testthat::expect_true()
)
```

---

character_ *Character generators*

Description

A set of generators for character vectors.

Usage

```
character_(len = c(1L, 10L), any_na = FALSE, any_empty = FALSE)

character_letters(len = c(1L, 10L), any_na = FALSE, any_empty = FALSE)

character_numbers(len = c(1L, 10L), any_na = FALSE, any_empty = FALSE)

character_alphanumeric(len = c(1L, 10L), any_na = FALSE, any_empty = FALSE)
```

Arguments

len	Length of the generated vectors. If len is a single number all vectors will have this length. If len is a numeric vector of length 2 it will produce vectors with lengths between a minimum and maximum, inclusive. For example len = c(1L,10L) would produce vectors with lengths between 1 and 10. To produce empty vectors set len = 0L or a range like len = c(0L,10L).
any_na	Whether NA values should be allowed.
any_empty	Whether empty character values should be allowed.

Value

A quickcheck_generator object.

Examples

```
character_() %>% show_example()
character_(len = 10L, any_na = TRUE) %>% show_example()
character_(len = 10L, any_empty = TRUE) %>% show_example()
```

---

constant *Generate the same value every time*

Description

Generate the same value every time

Usage

```
constant(a)
```

Arguments

a	Any R object

Value

A quickcheck_generator object.

Examples

```
constant(NULL) %>% show_example()
```

`data.table_` *data.table generators*

Description

Construct data.table generators in a similar way to `data.table::data.table`.

Usage

```
data.table_(..., rows = c(1L, 10L))
```

Arguments

`...`	A set of name-value pairs with the values being vector generators.
`rows`	Number of rows of the generated data frame. If `rows` is a single number all data frames will have this number of rows. If `rows` is a numeric vector of length 2 it will produce data frames with rows between a minimum and maximum, inclusive. For example `rows = c(1L,10L)` would produce data frames with rows between 1 and 10. To produce empty tibbles set `rows = 0L` or a range like `rows = c(0L,10L)`.

Value

A `quickcheck_generator` object.

Examples

```
data.table_(a = integer_()) %>% show_example()
data.table_(a = integer_(), b = character_(), rows = 5L) %>% show_example()
```

`data.table_of` *data.table generator with randomized columns*

Description

data.table generator with randomized columns

Usage

```
data.table_of(..., rows = c(1L, 10L), cols = c(1L, 10L))
```

Arguments

`...`	A set of unnamed generators. The generated data.tables will be built with random combinations of these generators.
`rows`	Number of rows of the generated data frame. If `rows` is a single number all data frames will have this number of rows. If `rows` is a numeric vector of length 2 it will produce data frames with rows between a minimum and maximum, inclusive. For example `rows = c(1L,10L)` would produce data frames with rows between 1 and 10. To produce empty tibbles set `rows = 0L` or a range like `rows = c(0L,10L)`.
`cols`	Number of columns of the generated data frame. If `cols` is a single number all data frames will have this number of columns. If `cols` is a numeric vector of length 2 it will produce data frames with columns between a minimum and maximum, inclusive. For example `cols = c(1L,10L)` would produce data frames with columns between 1 and 10. To produce empty tibbles set `cols = 0L` or a range like `cols = c(0L,10L)`.

Value

A `quickcheck_generator` object.

Examples

```
data.table_of(logical_(), date_()) %>% show_example()
data.table_of(any_atomic(), rows = 10L, cols = 5L) %>% show_example()
```

`data_frame_` *Data frame generators*

Description

Construct data frame generators in a similar way to `base::data.frame`.

Usage

```
data_frame_(..., rows = c(1L, 10L))
```

Arguments

`...`	A set of name-value pairs with the values being vector generators.
`rows`	Number of rows of the generated data frame. If `rows` is a single number all data frames will have this number of rows. If `rows` is a numeric vector of length 2 it will produce data frames with rows between a minimum and maximum, inclusive. For example `rows = c(1L,10L)` would produce data frames with rows between 1 and 10. To produce empty tibbles set `rows = 0L` or a range like `rows = c(0L,10L)`.

Value

A quickcheck_generator object.

Examples

```
data_frame_(a = integer_()) %>% show_example()
data_frame_(a = integer_(), b = character_(), rows = 5L) %>% show_example()
```

---

data_frame_of *Data frame generator with randomized columns*

Description

Data frame generator with randomized columns

Usage

```
data_frame_of(..., rows = c(1L, 10L), cols = c(1L, 10L))
```

Arguments

...	A set of unnamed generators. The generated data frames will be built with random combinations of these generators.
rows	Number of rows of the generated data frame. If rows is a single number all data frames will have this number of rows. If rows is a numeric vector of length 2 it will produce data frames with rows between a minimum and maximum, inclusive. For example rows = c(1L,10L) would produce data frames with rows between 1 and 10. To produce empty tibbles set rows = 0L or a range like rows = c(0L,10L).
cols	Number of columns of the generated data frame. If cols is a single number all data frames will have this number of columns. If cols is a numeric vector of length 2 it will produce data frames with columns between a minimum and maximum, inclusive. For example cols = c(1L,10L) would produce data frames with columns between 1 and 10. To produce empty tibbles set cols = 0L or a range like cols = c(0L,10L).

Value

A quickcheck_generator object.

Examples

```
data_frame_of(logical_(), date_()) %>% show_example()
data_frame_of(any_atomic(), rows = 10L, cols = 5L) %>% show_example()
```

date_ *Date generators*

Description

A set of generators for date vectors.

Usage

```
date_(len = c(1L, 10L), any_na = FALSE)

date_bounded(left, right, len = c(1L, 10L), any_na = FALSE)

date_left_bounded(left, len = c(1L, 10L), any_na = FALSE)

date_right_bounded(right, len = c(1L, 10L), any_na = FALSE)
```

Arguments

len	Length of the generated vectors. If `len` is a single number all vectors will have this length. If `len` is a numeric vector of length 2 it will produce vectors with lengths between a minimum and maximum, inclusive. For example `len = c(1L,10L)` would produce vectors with lengths between 1 and 10. To produce empty vectors set `len = 0L` or a range like `len = c(0L,10L)`.
any_na	Whether `NA` values should be allowed.
left	The minimum possible value for generated numbers, inclusive.
right	The maximum possible value for generated numbers, inclusive.

Value

A `quickcheck_generator` object.

Examples

```
date_() %>% show_example()
date_bounded(
  left = as.Date("2020-01-01"),
  right = as.Date("2020-01-10")
) %>% show_example()
date_(len = 10L, any_na = TRUE) %>% show_example()
```

`double_` *Double generators*

Description

A set of generators for double vectors.

Usage

```
double_(
  len = c(1L, 10L),
  any_na = FALSE,
  any_nan = FALSE,
  any_inf = FALSE,
  big_dbl = FALSE
)

double_bounded(
  left,
  right,
  len = c(1L, 10L),
  any_na = FALSE,
  any_nan = FALSE,
  any_inf = FALSE
)

double_left_bounded(
  left,
  len = c(1L, 10L),
  any_na = FALSE,
  any_nan = FALSE,
  any_inf = FALSE,
  big_dbl = FALSE
)

double_right_bounded(
  right,
  len = c(1L, 10L),
  any_na = FALSE,
  any_nan = FALSE,
  any_inf = FALSE,
  big_dbl = FALSE
)

double_positive(
  len = c(1L, 10L),
  any_na = FALSE,
```

```
  any_nan = FALSE,
  any_inf = FALSE,
  big_dbl = FALSE
)

double_negative(
  len = c(1L, 10L),
  any_na = FALSE,
  any_nan = FALSE,
  any_inf = FALSE,
  big_dbl = FALSE
)

double_fractional(
  len = c(1L, 10L),
  any_na = FALSE,
  any_nan = FALSE,
  any_inf = FALSE,
  big_dbl = FALSE
)

double_whole(
  len = c(1L, 10L),
  any_na = FALSE,
  any_nan = FALSE,
  any_inf = FALSE,
  big_dbl = FALSE
)
```

Arguments

len	Length of the generated vectors. If len is a single number all vectors will have this length. If len is a numeric vector of length 2 it will produce vectors with lengths between a minimum and maximum, inclusive. For example len = c(1L,10L) would produce vectors with lengths between 1 and 10. To produce empty vectors set len = 0L or a range like len = c(0L,10L).
any_na	Whether NA values should be allowed.
any_nan	Whether NaN values should be allowed.
any_inf	Whether Inf/-Inf values should be allowed.
big_dbl	Should doubles near the maximum size be included? This may cause problems because if the result of a computation results in a double larger than the maximum it will return Inf.
left	The minimum possible value for generated numbers, inclusive.
right	The maximum possible value for generated numbers, inclusive.

Value

A quickcheck_generator object.

Examples

```
double_() %>% show_example()
double_(big_dbl = TRUE) %>% show_example()
double_bounded(left = -5, right = 5) %>% show_example()
double_(len = 10L, any_na = TRUE) %>% show_example()
double_(len = 10L, any_nan = TRUE, any_inf = TRUE) %>% show_example()
```

---

equal_length *Equal length vector generator*

Description

Generates equal length vectors contained in a list.

Usage

```
equal_length(..., len = c(1L, 10L))
```

Arguments

...	A set of named or unnamed vector generators.
len	Length of the generated vectors. If `len` is a single number all vectors will have this length. If `len` is a numeric vector of length 2 it will produce vectors with lengths between a minimum and maximum, inclusive. For example `len = c(1L,10L)` would produce vectors with lengths between 1 and 10. To produce empty vectors set `len = 0L` or a range like `len = c(0L,10L)`.

Value

A `quickcheck_generator` object.

Examples

```
equal_length(integer_(), double_()) %>% show_example()
equal_length(a = logical_(), b = character_(), len = 5L) %>% show_example()
```

factor_ *Factor generator*

Description

A generator for factor vectors.

Usage

```
factor_(len = c(1L, 10L), any_na = FALSE)
```

Arguments

len	Length of the generated vectors. If len is a single number all vectors will have this length. If len is a numeric vector of length 2 it will produce vectors with lengths between a minimum and maximum, inclusive. For example len = c(1L,10L) would produce vectors with lengths between 1 and 10. To produce empty vectors set len = 0L or a range like len = c(0L,10L).
any_na	Whether NA values should be allowed.

Value

A quickcheck_generator object.

Examples

```
factor_() %>% show_example()
factor_(len = 10L, any_na = TRUE) %>% show_example()
```

flat_list_of *Variable length flat list generator*

Description

Generate flat lists with all values coming from a single generator. In a flat list all items will be scalars.

Usage

```
flat_list_of(generator, len = c(1L, 10L))
```

Arguments

generator	A quickcheck_generator object.
len	Length of the generated vectors. If len is a single number all vectors will have this length. If len is a numeric vector of length 2 it will produce vectors with lengths between a minimum and maximum, inclusive. For example len = c(1L,10L) would produce vectors with lengths between 1 and 10. To produce empty vectors set len = 0L or a range like len = c(0L,10L).

Value

A quickcheck_generator object.

Examples

```
flat_list_of(integer_(), len = 10L) %>% show_example()
```

---

for_all *Test properties of a function*

Description

Test properties of a function

Usage

```
for_all(
  ...,
  property,
  tests = getOption("quickcheck.tests", 100L),
  shrinks = getOption("quickcheck.shrinks", 100L),
  discards = getOption("quickcheck.discards", 100L)
)
```

Arguments

...	Named generators
property	A function which takes values from from the generator and calls an expectation on it. This function must have parameters matching the generator names.
tests	The number of tests to run.
shrinks	The maximum number of shrinks to run when shrinking a value to find the smallest counterexample.
discards	The maximum number of discards to permit when running the property.

Value

A testthat expectation object.

Examples

```
for_all(
  a = numeric_(len = 1L),
  b = numeric_(len = 1L),
  property = function(a, b) testthat::expect_equal(a + b, b + a)
)
```

`from_hedgehog` *Convert a hedgehog generator to a quickcheck generator*

Description

Convert a hedgehog generator to a quickcheck generator

Usage

```
from_hedgehog(generator)
```

Arguments

`generator`	A `hedgehog.internal.gen` object.

Value

A `quickcheck_generator` object.

Examples

```
is_even <-
  function(a) a %% 2L == 0L

gen_powers_of_two <-
  hedgehog::gen.element(1:10) %>% hedgehog::gen.with(function(a) 2 ^ a)

for_all(
  a = from_hedgehog(gen_powers_of_two),
  property = function(a) is_even(a) %>% testthat::expect_true()
)
```

hms_ *hms generators*

Description

A set of generators for hms vectors.

Usage

```
hms_(len = c(1L, 10L), any_na = FALSE)

hms_bounded(left, right, len = c(1L, 10L), any_na = FALSE)

hms_left_bounded(left, len = c(1L, 10L), any_na = FALSE)

hms_right_bounded(right, len = c(1L, 10L), any_na = FALSE)
```

Arguments

len	Length of the generated vectors. If len is a single number all vectors will have this length. If len is a numeric vector of length 2 it will produce vectors with lengths between a minimum and maximum, inclusive. For example len = c(1L,10L) would produce vectors with lengths between 1 and 10. To produce empty vectors set len = 0L or a range like len = c(0L,10L).
any_na	Whether NA values should be allowed.
left	The minimum possible value for generated numbers, inclusive.
right	The maximum possible value for generated numbers, inclusive.

Value

A quickcheck_generator object.

Examples

```
hms_() %>% show_example()
hms_bounded(
  left = hms::as_hms("00:00:00"),
  right = hms::as_hms("12:00:00")
) %>% show_example()
hms_(len = 10L, any_na = TRUE) %>% show_example()
```

integer_ *Integer generators*

Description

A set of generators for integer vectors.

Usage

```
integer_(len = c(1L, 10L), any_na = FALSE, big_int = FALSE)

integer_bounded(left, right, len = c(1L, 10L), any_na = FALSE)

integer_left_bounded(left, len = c(1L, 10L), any_na = FALSE, big_int = FALSE)

integer_right_bounded(right, len = c(1L, 10L), any_na = FALSE, big_int = FALSE)

integer_positive(len = c(1L, 10L), any_na = FALSE, big_int = FALSE)

integer_negative(len = c(1L, 10L), any_na = FALSE, big_int = FALSE)
```

Arguments

len	Length of the generated vectors. If len is a single number all vectors will have this length. If len is a numeric vector of length 2 it will produce vectors with lengths between a minimum and maximum, inclusive. For example len = c(1L,10L) would produce vectors with lengths between 1 and 10. To produce empty vectors set len = 0L or a range like len = c(0L,10L).
any_na	Whether NA values should be allowed.
big_int	Should integers near the maximum size be included? This may cause problems because if the result of a computation results in an integer larger than the maximum it will be silently coerced it to a double.
left	The minimum possible value for generated numbers, inclusive.
right	The maximum possible value for generated numbers, inclusive.

Value

A quickcheck_generator object.

Examples

```
integer_() %>% show_example()
integer_(big_int = TRUE) %>% show_example()
integer_bounded(left = -5L, right = 5L) %>% show_example()
integer_(len = 10L, any_na = TRUE) %>% show_example()
```

list_ *List generator*

Description

Generate lists with contents corresponding to the values generated by the input generators.

Usage

```
list_(...)
```

Arguments

...	A set of named or unnamed generators.

Value

A quickcheck_generator object.

Examples

```
list_(integer_(), logical_()) %>% show_example()
list_(a = any_vector(), b = any_vector()) %>% show_example()
```

list_of *Variable length list generator*

Description

Generate lists with all values coming from a single generator.

Usage

```
list_of(generator, len = c(1L, 10L))
```

Arguments

generator	A quickcheck_generator object.
len	Length of the generated vectors. If len is a single number all vectors will have this length. If len is a numeric vector of length 2 it will produce vectors with lengths between a minimum and maximum, inclusive. For example len = c(1L,10L) would produce vectors with lengths between 1 and 10. To produce empty vectors set len = 0L or a range like len = c(0L,10L).

Value

A quickcheck_generator object.

Examples

```
list_of(integer_(), len = 10L) %>% show_example()
```

`logical_` *Logical generator*

Description

A generator for logical vectors.

Usage

```
logical_(len = c(1L, 10L), any_na = FALSE)
```

Arguments

`len`	Length of the generated vectors. If `len` is a single number all vectors will have this length. If `len` is a numeric vector of length 2 it will produce vectors with lengths between a minimum and maximum, inclusive. For example `len = c(1L,10L)` would produce vectors with lengths between 1 and 10. To produce empty vectors set `len = 0L` or a range like `len = c(0L,10L)`.
`any_na`	Whether `NA` values should be allowed.

Value

A `quickcheck_generator` object.

Examples

```
logical_() %>% show_example()
logical_(len = 10L, any_na = TRUE) %>% show_example()
```

`numeric_` *Numeric generators*

Description

A set of generators for numeric vectors. Numeric vectors can be either integer or double vectors.

Usage

```
numeric_(len = c(1L, 10L), any_na = FALSE, big_num = FALSE)

numeric_bounded(left, right, len = c(1L, 10L), any_na = FALSE)

numeric_left_bounded(left, len = c(1L, 10L), any_na = FALSE, big_num = FALSE)

numeric_right_bounded(right, len = c(1L, 10L), any_na = FALSE, big_num = FALSE)

numeric_positive(len = c(1L, 10L), any_na = FALSE, big_num = FALSE)

numeric_negative(len = c(1L, 10L), any_na = FALSE, big_num = FALSE)
```

Arguments

len	Length of the generated vectors. If len is a single number all vectors will have this length. If len is a numeric vector of length 2 it will produce vectors with lengths between a minimum and maximum, inclusive. For example len = c(1L,10L) would produce vectors with lengths between 1 and 10. To produce empty vectors set len = 0L or a range like len = c(0L,10L).
any_na	Whether NA values should be allowed.
big_num	Should integers or doubles near the maximum size be included? This may cause problems because if the result of a computation results in a number larger than the maximum an integer will be silently coerced to a double and a double will return Inf.
left	The minimum possible value for generated numbers, inclusive.
right	The maximum possible value for generated numbers, inclusive.

Value

A quickcheck_generator object.

Examples

```
numeric_() %>% show_example()
numeric_(big_num = TRUE) %>% show_example()
numeric_bounded(left = -5L, right = 5L) %>% show_example()
numeric_(len = 10L, any_na = TRUE) %>% show_example()
```

---

one_of *Randomly choose between generators*

Description

Randomly choose between generators

Usage

```
one_of(..., prob = NULL)
```

Arguments

...	A set of unnamed generators.
prob	A vector of probability weights for obtaining the elements of the vector being sampled.

Value

A quickcheck_generator object.

Examples

```
one_of(integer_(), character_()) %>% show_example()
one_of(constant(NULL), logical_(), prob = c(0.1, 0.9)) %>% show_example()
```

---

posixct_ *POSIXct generators*

Description

A set of generators for POSIXct vectors.

Usage

```
posixct_(len = c(1L, 10L), any_na = FALSE)

posixct_bounded(left, right, len = c(1L, 10L), any_na = FALSE)

posixct_left_bounded(left, len = c(1L, 10L), any_na = FALSE)

posixct_right_bounded(right, len = c(1L, 10L), any_na = FALSE)
```

Arguments

len	Length of the generated vectors. If len is a single number all vectors will have this length. If len is a numeric vector of length 2 it will produce vectors with lengths between a minimum and maximum, inclusive. For example len = c(1L,10L) would produce vectors with lengths between 1 and 10. To produce empty vectors set len = 0L or a range like len = c(0L,10L).
any_na	Whether NA values should be allowed.
left	The minimum possible value for generated numbers, inclusive.
right	The maximum possible value for generated numbers, inclusive.

Value

A quickcheck_generator object.

Examples

```
posixct_() %>% show_example()
posixct_bounded(
  left = as.POSIXct("2020-01-01 00:00:00"),
  right = as.POSIXct("2021-01-01 00:00:00")
) %>% show_example()
posixct_(len = 10L, any_na = TRUE) %>% show_example()
```

---

repeat_test *Repeatedly test properties of a function*

Description

Repeatedly test properties of a function

Usage

```
repeat_test(property, tests = getOption("quickcheck.tests", 100L))
```

Arguments

property	A function with no parameters which includes an expectation.
tests	The number of tests to run.

Value

A testthat expectation object.

Examples

```
repeat_test(
  property = function() {
    num <- stats::runif(1, min = 0, max = 10)
    testthat::expect_true(num >= 0 && num <= 10)
  }
)
```

show_example *Show an example output of a generator*

Description

Show an example output of a generator

Usage

```
show_example(generator)
```

Arguments

generator	A quickcheck_generator object.

Value

An example output produced by the generator.

Examples

```
logical_() %>% show_example()
```

tibble_ *Tibble generators*

Description

Construct tibble generators in a similar way to tibble::tibble.

Usage

```
tibble_(..., rows = c(1L, 10L))
```

Arguments

...	A set of name-value pairs with the values being vector generators.
rows	Number of rows of the generated data frame. If rows is a single number all data frames will have this number of rows. If rows is a numeric vector of length 2 it will produce data frames with rows between a minimum and maximum, inclusive. For example rows = c(1L,10L) would produce data frames with rows between 1 and 10. To produce empty tibbles set rows = 0L or a range like rows = c(0L,10L).

Value

A quickcheck_generator object.

Examples

```
tibble_(a = integer_()) %>% show_example()
tibble_(a = integer_(), b = character_(), rows = 5L) %>% show_example()
```

`tibble_of` *Random tibble generator*

Description

Random tibble generator

Usage

```
tibble_of(..., rows = c(1L, 10L), cols = c(1L, 10L))
```

Arguments

`...`	A set of unnamed generators. The generated tibbles will be built with random combinations of these generators.
`rows`	Number of rows of the generated data frame. If `rows` is a single number all data frames will have this number of rows. If `rows` is a numeric vector of length 2 it will produce data frames with rows between a minimum and maximum, inclusive. For example `rows = c(1L,10L)` would produce data frames with rows between 1 and 10. To produce empty tibbles set `rows = 0L` or a range like `rows = c(0L,10L)`.
`cols`	Number of columns of the generated data frame. If `cols` is a single number all data frames will have this number of columns. If `cols` is a numeric vector of length 2 it will produce data frames with columns between a minimum and maximum, inclusive. For example `cols = c(1L,10L)` would produce data frames with columns between 1 and 10. To produce empty tibbles set `cols = 0L` or a range like `cols = c(0L,10L)`.

Value

A `quickcheck_generator` object.

Examples

```
tibble_of(logical_(), date_()) %>% show_example()
tibble_of(any_atomic(), rows = 10L, cols = 5L) %>% show_example()
```

Index

any_atomic, 3
any_data.table, 4
any_data_frame, 4
any_flat_homogeneous_list, 5
any_flat_list, 6
any_list, 6
any_tibble, 7
any_undefined, 8
any_vector, 8
anything, 2
as_hedgehog, 9

character_, 9
character_alphanumeric *(character_)*, 9
character_letters *(character_)*, 9
character_numbers *(character_)*, 9
constant, 10

data.table_, 11
data.table_of, 11
data_frame_, 12
data_frame_of, 13
date_, 14
date_bounded *(date_)*, 14
date_left_bounded *(date_)*, 14
date_right_bounded *(date_)*, 14
double_, 15
double_bounded *(double_)*, 15
double_fractional *(double_)*, 15
double_left_bounded *(double_)*, 15
double_negative *(double_)*, 15
double_positive *(double_)*, 15
double_right_bounded *(double_)*, 15
double_whole *(double_)*, 15

equal_length, 17

factor_, 18
flat_list_of, 18
for_all, 19
from_hedgehog, 20

hms_, 21
hms_bounded *(hms_)*, 21
hms_left_bounded *(hms_)*, 21
hms_right_bounded *(hms_)*, 21

integer_, 22
integer_bounded *(integer_)*, 22
integer_left_bounded *(integer_)*, 22
integer_negative *(integer_)*, 22
integer_positive *(integer_)*, 22
integer_right_bounded *(integer_)*, 22

list_, 23
list_of, 23
logical_, 24

numeric_, 24
numeric_bounded *(numeric_)*, 24
numeric_left_bounded *(numeric_)*, 24
numeric_negative *(numeric_)*, 24
numeric_positive *(numeric_)*, 24
numeric_right_bounded *(numeric_)*, 24

one_of, 25

posixct_, 26
posixct_bounded *(posixct_)*, 26
posixct_left_bounded *(posixct_)*, 26
posixct_right_bounded *(posixct_)*, 26

repeat_test, 27

show_example, 28

tibble_, 28
tibble_of, 29