Package 'discretes'

March 31, 2026

Type Package

Title Discrete Numeric Series

Version 0.1.0

Description Provides a framework for representing discrete numeric series (enumerable sets of numbers) that may be finite or infinite. Series can be traversed, combined using arithmetic operations, tested for membership, and queried for limit points (``sinks''), without explicit enumeration of all elements.

License MIT + file LICENSE

Imports checkmate, ellipsis, graphics, rlang

Encoding UTF-8

RoxygenNote 7.3.3

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

URL https://discretes.netlify.app, https://github.com/vincenzocoia/discretes

BugReports https://github.com/vincenzocoia/discretes/issues

NeedsCompilation no

Author Vincenzo Coia [aut, cre, cph]

Maintainer Vincenzo Coia <vincenzo.coia@gmail.com>

Repository CRAN

Date/Publication 2026-03-31 10:00:08 UTC

Contents

`arithmetic` *Arithmetic series*

Description

Construct an arithmetic progression, with possibly infinite values. The progression is anchored at `representative` and extends `n_left` steps to the left (decreasing values) and `n_right` steps to the right (increasing values) with constant `spacing` between consecutive terms.

Usage

```
arithmetic(representative, spacing, ..., n_left = Inf, n_right = Inf)
```

Arguments

`representative`	Numeric scalar giving a known term in the progression.
`spacing`	Non-negative numeric scalar describing the distance between adjacent terms.
`...`	Reserved for future extensions; must be empty.
`n_left`, `n_right`	Non-negative counts (possibly `Inf`, the default) describing how many steps exist to the left and right of `representative`.

Value

A numeric series (class `dsct_arithmetic`, inheriting from `discretes`).

Note

While `spacing` can be zero, this results in a numeric series containing only the `representative` value as its single discrete value.

The series can only contain `-0` if the `representative` is set as such.

See Also

`integers()`

Examples

```
arithmetic(representative = -0.6, spacing = 0.7)
arithmetic(representative = 0.6, spacing = 0.7, n_right = 0)
arithmetic(representative = 0, spacing = 2, n_left = 2, n_right = 2)

# Negative zero, resulting in `-Inf` upon inversion:
has_negative_zero(arithmetic(-0, 1))
```

`as.double.discretes` *Convert a numeric series to a numeric vector*

Description

Return all discrete values in the numeric series, if finite. Throws an error if infinite.

Usage

```
## S3 method for class 'discretes'
as.double(x, ...)
```

Arguments

`x`	Numeric series (`numeric` vector or object of class `"discretes"`).
`...`	Arguments to pass downstream to `as.numeric()` that's called on the resulting vector of discrete values.

Value

A numeric vector containing all discrete values in `x`, ordered from smallest to largest. Returns `numeric(0)` when the interval contains no discrete values. Numeric outputs are wrapped in `as.numeric()`.

See Also

`get_discretes_in()`

Examples

```
as.numeric(integers(-3.5, 10))
```

as_discretes *Convert to a discretes object*

Description

Convert a foreign object to a "discretes" object.

Usage

```
as_discretes(x)

## S3 method for class 'discretes'
as_discretes(x)

## S3 method for class 'numeric'
as_discretes(x)
```

Arguments

x	Object to convert to object of class "discretes".

Value

A numeric series (object of class `"discretes"`). When `x` is a numeric vector, the series contains all unique values of `x`.

Methods (by class)

- `as_discretes(discretes)`: Convert a numeric vector to discretes object.
- `as_discretes(numeric)`: Keeps the discretes object as-is.

Examples

```
as_discretes(0:10)
```

dsct_drop *Subset a numeric series*

Description

Subset a numeric series

Usage

```
dsct_drop(
  x,
  from = -Inf,
  to = Inf,
  ...,
  include_from = TRUE,
  include_to = TRUE
)

dsct_keep(
  x,
  from = -Inf,
  to = Inf,
  ...,
  include_from = TRUE,
  include_to = TRUE
)
```

Arguments

x	Numeric series (numeric vector or object of class "discretes").
from, to	Numeric values defining the range to keep; single numerics with from <= to.
...	Reserved for future extensions; must be empty.
include_from, include_to	Logical values indicating whether the from and to values should be included in the kept range; single logicals.

Value

A numeric series representing the subset of discrete values within the specified range.

Examples

```
x <- integers(from = -3)
dsct_keep(x, from = -1.5, to = 2.5)
dsct_keep(x, to = 2)
```

---

dsct_transform *Monotonically transform a numeric series*

Description

Apply a function that is strictly increasing or strictly decreasing to a numeric series.

Usage

```
dsct_transform(x, fun, ...)

## S3 method for class 'discretes'
dsct_transform(
  x,
  fun,
  inv,
  ...,
  domain = c(-Inf, Inf),
  range = c(-Inf, Inf),
  dir = c("increasing", "decreasing")
)
```

Arguments

x	Numeric series (numeric vector or object of class "discretes").
fun, inv	A vectorized, strictly monotonic function to apply to the discrete values, and its inverse, inv.
...	Arguments to pass to specific methods.
domain, range	Numeric vectors of length 2, indicating the domain and range of fun (that is, the interval on which fun is valid, and the interval to which fun maps).
dir	A string, either "increasing" or "decreasing", indicating the monotonicity of the function fun.

Details

Strictly increasing means that for any x1 < x2, it holds that fun(x1) < fun(x2), for all values on the real line. The function -1/x, for example, is not strictly increasing: its derivative is increasing, but switches to smaller values after x = 0, therefore is not strictly increasing. Strictly decreasing is the opposite, in that we have fun(x1) > fun(x2).

If a decreasing function is provided, the transformation is negated internally first, and then transformed with fun(-x).

Value

A numeric series with the transformation applied.

Examples

```
dsct_transform(integers(), fun = pnorm, inv = qnorm, range = c(0, 1))
dsct_transform(
  as_discretes(0:3),
  fun = cos,
  inv = acos,
  domain = c(0, pi),
  range = c(-1, 1),
  dir = "decreasing"
```

```
)

# For numeric inputs, function is applied directly.
# Other arguments beyond `fun` get absorbed in `...` and are not used.
dsct_transform(0:5, exp)
dsct_transform(0:5, exp, log)
```

`dsct_union` *Combine numeric series*

Description

Combine one or more numeric series (or numeric vectors interpreted as numeric series) into a single union, where each unique discrete value is counted once.

Usage

```
dsct_union(...)
```

Arguments

`...`	Objects to combine. Each must be either a numeric series or a numeric vector.

Details

Values are flattened

Value

A numeric series (inheriting from `dsct_union`).

Note

While both `-0` and `+0` can both exist,

Examples

```
x1 <- natural1()
x2 <- -natural1()
dsct_union(x1, x2)
```

empty_series *Create an empty numeric series*

Description

Create a numeric series with no discrete values.

Usage

```
empty_series(mode = c("double", "integer"))
```

Arguments

mode	Character vector of length 1 indicating the numeric type of the numeric series; either "double" (the default) or "integer".

Value

An empty numeric series of the specified mode.

Examples

```
empty_series()
```

get_discretes_at *Extract discrete values from a numeric series*

Description

Extract a finite subset of discrete values from a numeric series by asking for specific values (`get_discretes_at()`) or by setting a range (`get_discretes_in()`). For `get_discretes_at()`, each value in `values` is checked against the series using `tol` (passed down to the underlying series); if it is in the series, the corresponding discrete value is returned, otherwise it is dropped. `NA` values are kept in place.

Usage

```
get_discretes_at(x, values, ..., tol = sqrt(.Machine$double.eps))

get_discretes_in(
  x,
  from = -Inf,
  to = Inf,
  ...,
  include_from = TRUE,
  include_to = TRUE,
  tol = sqrt(.Machine$double.eps)
)
```

Arguments

x	Numeric series (numeric vector or object of class "discretes").
values	Numeric vector of values to pull from the numeric series x.
...	Reserved for future extensions; must be empty.
tol	Numerical tolerance when checking if a value is in the series. Single non-negative numeric. See next_discrete() for details.
from, to	Reference values, possibly infinite. from must be less than or equal to to; both must be length-1 numeric vectors.
include_from, include_to	Should the from value be included in the query? Should the to value? Both must be length-1 logical vectors.

Value

A numeric vector containing all discrete values in the provided series x:

- For get_discretes_in(), all discrete values between from and to, ordered from smallest to largest.
- For get_discretes_at(), the discrete values in the series that match the given values. Whether a value is in the series is decided with tol at the underlying series, returning the discrete values in the series rather than the supplied values. Discrete values not within tol of the supplied values are dropped.

An error will be thrown in get_discretes_in() if there are infinitely many points in the range.

See Also

as.double.discretes()

Examples

```
get_discretes_in(integers(), from = 6.6, to = 10.1)
get_discretes_in(1 / arithmetic(1, 4, n_left = 3, n_right = 5))
get_discretes_at(integers(), values = c(-10, 4, 3.5, 10, NA))
get_discretes_at(integers(), values = 5.5)
```

---

has_discretes *Check which values are in a numeric series*

Description

Check which values are in a numeric series

Usage

```
has_discretes(x, values, ..., tol = sqrt(.Machine$double.eps))
```

Arguments

x	Numeric series (numeric vector or object of class "discretes").
values	A vector of values to check.
...	Reserved for future extensions; must be empty.
tol	Numerical tolerance when checking if a value is in the series. Single non-negative numeric. See next_discrete() for details.

Value

A logical vector indicating whether each value is in the numeric series x. NA values are preserved such that NA in values results in NA in the output.

Note

This function does not distinguish between +0 and -0. For that, use has_negative_zero() or has_positive_zero().

Examples

```
has_discretes(natural0(), c(-1, 0, 1, 12.5, NA))
has_discretes(1 / natural1(), 0)
```

---

has_negative_zero *Check if a numeric series has a signed zero*

Description

Check if a numeric series contains zero with a negative sign (-0) or a positive sign (+0). See details.

Usage

```
has_negative_zero(x)

has_positive_zero(x)
```

Arguments

x	Numeric series (numeric vector or object of class "discretes").

Details

While +0 and -0 are identical in R, they have a latent sign that appears in reciprocals: 1 / +0 is Inf, while 1 / -0 is -Inf. The has_negative_zero() and has_positive_zero() functions report whether -0 and +0 are discrete values in the numeric series. Behaviour is consistent with signed zero in numeric vectors.

A numeric series can contain both -0 and +0, like c(0, -0). Only one zero is returned by next_discrete(), prev_discrete(), or get_discretes_in(), as with unique(c(0, -0)). Their presence remains latent and appears when the series is inverted, giving both Inf and -Inf. See the examples.

Value

A single logical: whether `-0` is a discrete value in the series for `has_negative_zero()`, and whether `+0` is a discrete value for `has_positive_zero()`. Both can be `TRUE`; see details.

Examples

```
has_negative_zero(integers())
has_positive_zero(integers())

# Integer 0 can never be negative, but double can:
has_negative_zero(-integers())
has_negative_zero(-1.5 * integers())

# -0 and +0 can co-exist, but are never double counted. However, they
# get expressed differently when the series is inverted.
a <- c(0, -0)
num_discretes(a)
num_discretes(1 / a)

b <- dsct_union(integers(from = -1, to = 1), -0)
num_discretes(b)
num_discretes(1 / b)
```

---

has_sink_in *Test for sinks in a numeric series*

Description

`has_sink_in()` tests whether a numeric series has a sink in the interval `[from, to]`. `has_sink_at()` tests whether there is a sink at a given value, optionally with a specific direction.

Usage

```
has_sink_in(x, from = -Inf, to = Inf)

has_sink_at(x, value, dir = c("either", "left", "right", "both"))
```

Arguments

`x`	Numeric series (`numeric` vector or object of class `"discretes"`).
`from`, `to`	Reference values, possibly infinite. `from` must be less than or equal to `to`; both must be length-1 numeric vectors.
`value`	Single numeric to check for the existence of a sink.
`dir`	Single character: direction of the sink. `"either"` (default) ignores direction; `"left"` or `"right"` require the sink to be approached from that side; `"both"` requires it to be approached from both sides.

Value

A length-one logical: TRUE if a sink exists in the range (has_sink_in) or at value with the given dir (has_sink_at); FALSE otherwise.

See Also

See sinks() to get all sinks, and a description of sinks.

Examples

```
# The set of integers have sinks at +Inf and -Inf
has_sink_in(integers())
has_sink_at(integers(), Inf)
has_sink_at(integers(), -Inf, dir = "right")

# The set 1, 1/2, 1/4, 1/8, ... has a sink at 0 approached from the right.
halves <- 0.5^natural0()
has_sink_in(halves, to = 0)
has_sink_at(halves, 0, dir = "right")

# Reciprocal of integers: sink at 0 from both sides
reciprocals <- 1 / integers()
has_sink_at(reciprocals, 0, dir = "both")
```

---

integers *Integer numeric series*

Description

Use integers() to create a numeric series whose discrete values are integers within a specified range, possibly unbounded on either end. Use natural0() and natural1() for the natural numbers starting at 0 or 1.

Usage

```
integers(from = -Inf, to = Inf)

natural1()

natural0()
```

Arguments

from, to	Numeric values defining the range of integers. Defaults to -Inf and Inf, representing all integers; the series is not closed, so -Inf and Inf are never discrete values.

Value

A numeric series (arithmetic, class "dsct_arithmetic") whose discrete values are the integers in the specified range.

See Also

arithmetic()

Examples

```
integers()                  # All integers
integers(from = 0)          # Non-negative integers
integers(to = 1.5)          # Ends at 1.
integers(-5, 5) # Integers from -5 to 5.
natural1()
natural0()

# Infinity is never contained in the series.
has_discretes(integers(), Inf)
```

---

is_series *Check if an object is treated as a numeric series*

Description

Returns TRUE if an object inherits the class "discretes" or is a numeric vector.

Usage

```
is_series(x)
```

Arguments

x	Object to check.

Value

TRUE if x is treated as a numeric series, FALSE otherwise.

Examples

```
is_series(natural0())
is_series(c(1, 2, 3))
is_series("not a numeric series")
```

Math.discretes *Math group generic for numeric series*

Description

Support for exp, log, log10, log2, and sqrt on numeric series. The new series behaves as though the function is applied to each discrete value in the series.

Usage

```
## S3 method for class 'discretes'
Math(x, ...)
```

Arguments

x	Numeric series (numeric vector or object of class "discretes").
...	Passed to log() for the base argument when op == "log".

Value

A numeric series with the math function applied.

Examples

```
exp(integers(from = 0, to = 3))
log(natural1())
sqrt(integers(from = 0, to = 10))
```

next_discrete *Traversing a numeric series*

Description

next_discrete() and prev_discrete() find the n discrete values in a numeric series next to a reference point. num_discretes() finds the number of discrete values within a range.

Usage

```
next_discrete(
  x,
  from,
  ...,
  n = 1L,
  include_from = FALSE,
  tol = sqrt(.Machine$double.eps)
)
```

```
prev_discrete(
  x,
  from,
  ...,
  n = 1L,
  include_from = FALSE,
  tol = sqrt(.Machine$double.eps)
)
```

Arguments

x	Numeric series (numeric vector or object of class "discretes").
from	Reference value to start searching from; single numeric.
...	Reserved for future extensions; must be empty.
n	Number of discrete values to find; single positive integer.
include_from	Should the from value be included in the query? Single logical; defaults to TRUE.
tol	Numerical tolerance when checking if a value is in the series. Single non-negative numeric. See next_discrete() for details.

Details

For a transformed or combined series (e.g. 1 / integers()), traversal is delegated to the base series, and tol is passed through to those calls. This means that tol is only realized on the underlying series:

- **Numeric vector** (or as_discretes(...)): a value is a member if it is within tol of a stored value.
- **Arithmetic series**: the implied step index (distance from the representative in steps) is treated as an integer if it is within tol of an integer. See the Tolerance vignette for examples.

Value

A vector of sequential points starting from from, which is included in the vector if it is a discrete value in the numeric series and include_from = TRUE. The length of the vector is at most n, and will return an empty numeric vector if there is no such discrete value. next_discrete() is increasing, while prev_discrete() is decreasing, so that earlier values are encountered sooner.

Examples

```
x <- integers(from = 2)
next_discrete(x, from = 1.3)
prev_discrete(x, from = 4, n = 10)
next_discrete(x - 0.7, from = 1.3, n = 4)
```

num_discretes *Number of discrete values in a series*

Description

Return the number of discrete values in `x` that lie between `from` and `to`, or test whether the number of discrete values is infinite.

Usage

```
num_discretes(
  x,
  from = -Inf,
  to = Inf,
  ...,
  include_from = TRUE,
  include_to = TRUE,
  tol = sqrt(.Machine$double.eps)
)
```

Arguments

`x`	Numeric series (`numeric` vector or object of class `"discretes"`).
`from`, `to`	Reference values, possibly infinite. `from` must be less than or equal to `to`; both must be length-1 numeric vectors.
`...`	Reserved for future extensions; must be empty.
`include_from`, `include_to`	Should the `from` value be included in the query? Should the `to` value? Both must be length-1 logical vectors.
`tol`	Numerical tolerance when checking if a value is in the series. Single non-negative numeric. See `next_discrete()` for details.

Value

For `num_discretes()`, a single non-negative integer, or possibly `Inf` for infinitely many discrete values.

Examples

```
num_discretes(-3:3)
num_discretes(c(0.4, 0.4, 0.4, 0))

x <- arithmetic(-3.2, spacing = 0.5)
num_discretes(x)
num_discretes(x, from = -2, to = 2)
num_discretes(1 / x, from = -2, to = 2)
```

Ops.discretes *Arithmetic and power operators for numeric series*

Description

Support for +, -, *, /, and ^ between a numeric series and a single number. One operand must be a numeric series and the other a number.

Usage

```
## S3 method for class 'discretes'
Ops(e1, e2)
```

Arguments

e1, e2	Operands; one must be a numeric series (class discretes), the other a single numeric.

Value

A numeric series resulting from the operation (e.g. series + number, number * series, series^number).

Examples

```
integers() + 1
2 * natural1()
1 / integers(from = 1, to = 5)
natural0()^2
```

plot.discretes *Plot a numeric series*

Description

Plot the discrete values in a numeric series within a specified interval.

Usage

```
## S3 method for class 'discretes'
plot(
  x,
  from = -Inf,
  to = Inf,
  ...,
  closeness = 0.01,
  tol = sqrt(.Machine$double.eps)
)
```

Arguments

x	Numeric series (numeric vector or object of class "discretes").
from, to	Numeric values defining the range to plot; single numerics.
...	Additional arguments passed to the underlying plot() function.
closeness	Numeric value indicating how close to the (non-infinite) sinks the points should no longer be plotted. This is because there are an infinite number of points around each sink.
tol	Passed to dsct_keep() when subsetting the series between from and to.

Details

Sinks at finite values are indicated by vertical dotted gray lines. A red tick mark is used to indicate that a finite sink value is part of the series.

When the series extends to infinity in either direction, three arrows (< or >) are drawn to indicate this. When infinity is part of the series, the last arrow is red.

This is a simple plotting scheme with naive handling of infinite discrete values:

- The closeness parameter does not adjust with the scale of the data, so may require tuning more often by the user.
- When the series extends to infinity (in either direction), an arbitrary cutoff of 10 units beyond the last finite sink or representative() value (whichever is closer to the infinite sink) is used. This can be manually adjusted by changing the from and to parameters.

Value

Invisibly returns the input x object after printing a plot in Base R.

Note

If this function takes a long time to plot, it's likely because your series has slowly varying behaviour next to a sink, like 1 / natural1(), where discrete values pile up rapidly while approaching the sink very slowly. To avoid so many points from being plotted, increase the closeness argument.

Examples

```
plot(integers())
plot(integers(), from = -50, to = 50)
plot(0.5^natural1(), closeness = 1e-3)
plot(dsct_union(0.5^natural1(), 0), closeness = 1e-3)
```

print.discretes *Print a numeric series*

Description

Print a numeric series to the console.

Usage

```
## S3 method for class 'discretes'
print(x, len = 6, ...)
```

Arguments

x	Numeric series (numeric vector or object of class "discretes").
len	Number of discrete values to display.
...	Further arguments to pass to downstream methods; currently not used.

Value

Invisibly returns the input object x.

Examples

```
print(integers())
print(1 / natural1())
print(-1 / natural1())
print(1 / integers())
print(1 / integers(), len = 1)
print(1 / integers(), len = 0)
```

representative *Get a representative discrete value in a numeric series*

Description

Get a representative discrete value in a numeric series

Usage

```
representative(x)
```

Arguments

x	Numeric series (numeric vector or object of class "discretes").

Value

A single numeric: a representative discrete value from the numeric series, with the proper mode.

Examples

```
representative(integers())
representative(natural1() + 7)
```

`sinks` *Sinks*

Description

Sinks are limit points in a numeric series. That means that discrete values get arbitrarily close to the sink (from the left or right), and there are infinitely many discrete values around the sink. This function returns a matrix of all sinks in the numeric series.

Usage

```
sinks(x)
```

Arguments

`x`	Numeric series (`numeric` vector or object of class `"discretes"`).

Value

A matrix with two columns: `location` and `direction`. Each row corresponds to a sink, where `location` is the location of the sink (possibly infinite), and `direction` indicates whether the sink is approached from the left (-1) or right (1).

See Also

`has_sink_in()`, `has_sink_at()`

Examples

```
# The set of integers have sinks at +Inf and -Inf
sinks(integers())

# The set 1, 1/2, 1/4, 1/8, ... has a sink at 0 approached from the right.
halves <- 0.5^natural0()
sinks(halves)

# The reciprocal of the integers has a sink at 0 approached from both the
# left and right; while the integer 0 gets mapped to Inf, infinity is not a
# sink because discrete values don't get arbitrarily close to it.
reciprocals <- 1 / integers()
sinks(reciprocals)
has_discretes(reciprocals, Inf) # Yet Inf is a discrete value.
```

`[.discretes` *Subset a numeric series by position*

Description

When a series has a well-defined "first" element (e.g. `natural1()` starts at 1), subsetting with `[]` materializes a specified part of the series, and mirrors the behaviour of numeric vector subsetting. Positive `i` returns the discrete values at those positions, and negative `i` tries to return the full series with the specified positions dropped.

Usage

```
## S3 method for class 'discretes'
x[i]

## S3 replacement method for class 'discretes'
x[i] <- value
```

Arguments

`x`	A numeric series (object of class `"discretes"`).
`i`	Numeric vector of indices. Omit for the full series (finite only).
`value`	Replacement value; ignored, because replacement via `[<-` is not supported.

Details

Subsetting via `[]` tries to delegate to native behaviour on numeric vectors as quickly as possible by first materializing the series as a vector, and then conducting the subsetting.

- If `i` is missing or has negative values, subsetting is delegated to the full series materialized via `get_discretes_in()` (if possible).
- If `i` is `NULL` or length-0, subsetting is delegated to a representative value of the series.
- If `i` doesn't have negative values, subsetting is delegated to the series materialized as far out as needed to cover all `i`s, via `next_discrete()` from `-Inf`.

Value

A vector of discrete values. When the series has no first element or too few values for positive `i`, R returns NA as for ordinary vectors. For negative `i` or missing `i`, the full series is obtained first; infinite series behaviour defaults to that of `get_discretes_in()`.

Note

Unlike subsetting numeric vectors, the following actions are not supported:

- Replacement via `[<-` (throws an error).
- Subsetting by a character vector, as though subsetting by entry names.
- Subsetting by a logical vector.

Examples

```
natural1()[2]
natural1()[c(1, 3, 5)]
integers(1, 5)[-1]   # full series with first value dropped

# Subsetting from the other side of a sink
x <- 1 / natural1()
x[1:3]   # No such thing as a "first" value; returns NA.
y <- dsct_union(x, -1)
y[1:3]   # "-1" is the 1st value, but no such thing as 2nd or 3rd value.
```

Index