Package ‘bignum’

May 4, 2023

Title  Arbitrary-Precision Integer and Floating-Point Mathematics

Version  0.3.2

Description  Classes for storing and manipulating arbitrary-precision integer vectors and high-precision floating-point vectors. These extend the range and precision of the 'integer' and 'double' data types found in R. This package utilizes the 'Boost.Multiprecision' C++ library. It is specifically designed to work well with the 'tidyverse' collection of R packages.

License  MIT + file LICENSE


BugReports  https://github.com/davidchall/bignum/issues

Depends  R (>= 3.3.0)

Imports  rlang, vctrs (>= 0.3.0)

Suggests  knitr, pillar (>= 1.6.3), rmarkdown, testthat

LinkingTo  BH, cpp11

VignetteBuilder  knitr

Config/testthat/edition  3

Encoding  UTF-8

RoxygenNote  7.2.3

NeedsCompilation  yes

Author  David Hall [aut, cre, cph] (<https://orcid.org/0000-0002-2193-0480>)

Maintainer  David Hall <david.hall.physics@gmail.com>

Repository  CRAN

Date/Publication  2023-05-04 06:40:02 UTC
**Description**

`bigfloat()` and `as_bigfloat()` construct a vector designed to store numbers with 50 decimal digits of precision.

`is_bigfloat()` checks if an object is of class `bignum_bigfloat`.

**Usage**

```r
bigfloat(x = character())
as_bigfloat(x)
is_bigfloat(x)
```

**Arguments**

- `x`  
  Object to be coerced or tested.

**Value**

An S3 vector of class `bignum_bigfloat`.

**See Also**

- `NA_bigfloat_` to represent missing values.
- `format()` for pretty printing.
- `vignette("operations")` for supported operations.
Examples

# default options limit displayed precision
bigfloat(1) / 3

# display full precision
format(bigfloat(1) / 3, sigfig = 50, notation = "dec")

---

biginteger

_Arbitrary-Precision Integer Vectors_

Description

biginteger() and as_biginteger() construct a vector designed to store _any_ integer.

is_biginteger() checks if an object is of class bignum_biginteger.

Usage

biginteger(x = character())

as_biginteger(x)

is_biginteger(x)

Arguments

x Object to be coerced or tested.

Value

An S3 vector of class bignum_biginteger.

See Also

NA_biginteger_ to represent missing values.

format() for pretty printing.

vignette("operations") for supported operations.

Examples

# default options limit displayed precision
biginteger(2)^50L

# display full precision
format(biginteger(2)^50L, notation = "dec")

# lossy casts raise a warning
biginteger(c(2, 2.5, 3))
suppressWarnings(biginteger(c(2, 2.5, 3)))

biginteger("0xffffffff")

---

**Description**

`biginteger` and `bigfloat` vectors support the standard arithmetic operations. The base R documentation can be found at `Arithmetic`.

**Value**

These arithmetic operations are **type-stable**, which means the output type depends only on the input types (not the input values). A biginteger vector is returned when the result must be an integer (e.g., addition of two integers). Otherwise a bigfloat vector is returned.

The following table summarizes the return type for each combination, where "integer-like" refers to integer and biginteger vectors and "float-like" refers to double and bigfloat vectors.

<table>
<thead>
<tr>
<th>Input 1</th>
<th>Operator</th>
<th>Input 2</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer-like</td>
<td>+, -, *, ^, %%</td>
<td>Integer-like</td>
<td>-&gt; biginteger</td>
</tr>
<tr>
<td>Integer-like</td>
<td>+, -, *, ^, %%</td>
<td>Float-like</td>
<td>-&gt; bigfloat</td>
</tr>
<tr>
<td>Float-like</td>
<td>+, -, *, ^, %%</td>
<td>Integer-like</td>
<td>-&gt; bigfloat</td>
</tr>
<tr>
<td>Float-like</td>
<td>+, -, *, ^, %%</td>
<td>Float-like</td>
<td>-&gt; bigfloat</td>
</tr>
<tr>
<td>Any</td>
<td>/</td>
<td>Any</td>
<td>-&gt; bigfloat</td>
</tr>
<tr>
<td>Any</td>
<td>%/%</td>
<td>Any</td>
<td>-&gt; biginteger</td>
</tr>
</tbody>
</table>

**See Also**

Other bignum operations: `bignum-compare, bignum-math, bignum-special`

**Examples**

```r
x <- biginteger(5)
y <- bigfloat(2)
+x
-x
x + y
x - y
x * y
x / y
x^y
```
bignum-compare

\[
x \mod y \\
x \equiv y
\]

bignum-compare  
Comparison operations

Description

biginteger and bigfloat vectors support the standard comparison operations. The base R documentation can be found at Comparison.

Value

A logical vector.

See Also

Other bignum operations: bignum-arith, bignum-math, bignum-special

Examples

```r
x <- biginteger(5)
y <- bigfloat(2)

x < y
x > y
x <= y
x >= y
x == y
x != y
```

bignum-constants  
Constants

Description

NA_biginteger_ and NA_bigfloat_ support missing values.

bigpi is a higher precision version of pi.

Usage

NA_biginteger_

NA_bigfloat_

bigpi
Value

A `biginteger` or `bigfloat` vector of length 1.

See Also

`NA` and `pi` are the base constants.

Examples

```r
NA_biginteger_

NA_bigfloat_

# default options limit displayed precision
bigpi

# display full precision
format(bigpi, sigfig = 50, notation = "dec")
```

Description

Customize how a `biginteger` or `bigfloat` vector is displayed. The precision can be controlled with a number of significant figures, or with a maximum or fixed number of digits after the decimal point. You can also choose between decimal, scientific and hexadecimal notations.

The default formatting applied when printing depends on the type of object:

- **standalone vector**: consults "bignum.sigfig" and "bignum.max_dec_width"
- **tibble column**: consults "pillar.sigfig" and "pillar.max_dec_width"

Usage

```r
## S3 method for class 'bignum_biginteger'
format(
x, 
...
, sigfig = NULL,
  digits = NULL,
  notation = c("fit", "dec", "sci", "hex")
)

## S3 method for class 'bignum_bigfloat'
format(x, ..., sigfig = NULL, digits = NULL, notation = c("fit", "dec", "sci"))
```
Arguments

x  A biginteger or bigfloat vector.

...  These dots are for future extensions and must be empty.

sigfig  Number of significant figures to show. Must be positive. Cannot be combined with digits.

If both sigfig and digits are unspecified, then consults the "bignum.sigfig" option (default: 7).

digits  Number of digits to show after the decimal point. Positive values indicate the exact number of digits to show. Negative values indicate the maximum number of digits to show (terminal zeros are hidden if there are no subsequent non-zero digits). Cannot be combined with sigfig.

notation  How should the vector be displayed? Choices:

  • "fit": Use decimal notation if it fits, otherwise use scientific notation. Consults the "bignum.max_dec_width" option (default: 13).
  • "dec": Use decimal notation, regardless of width.
  • "sci": Use scientific notation.
  • "hex": Use hexadecimal notation (positive biginteger only).

Value

Character vector

Examples

# default uses decimal notation
format(bigfloat(1e12))

# until it becomes too wide, then it uses scientific notation
format(bigfloat(1e12))

# hexadecimal notation is supported for positive integers
format(biginteger(255), notation = "hex")

# significant figures
format(bigfloat(12.5), sigfig = 2)

# fixed digits after decimal point
format(bigfloat(12.5), digits = 2)

# maximum digits after decimal point
format(bigfloat(12.5), digits = -2)
bignum-math

Mathematical operations

Description

`biginteger` and `bigfloat` vectors support many of the standard mathematical operations. The base R documentation can be found by searching for the individual functions (e.g. `mean()`).

Value

The returned value depends on the individual function. We recommend reading the base R documentation for a specific function to understand the expected result.

See Also

Other bignum operations: `bignum-arith`, `bignum-compare`, `bignum-special`

Examples

```r
# summary
x <- bigfloat(1:5)
sum(x)
prod(x)
max(x)
min(x)
range(x)
mean(x)

# cumulative
x <- bigfloat(1:5)
cumsum(x)
cumprod(x)
cummax(x)
cummin(x)

# rounding
x <- bigfloat(1.5)
floor(x)
ceiling(x)
trunc(x)

# miscellaneous
x <- bigfloat(2)
abs(x)
sign(x)
sqrt(x)

# logarithms and exponentials
x <- bigfloat(2)
log(x)
```
\[
\begin{align*}
\log_{10}(x) \\
\log_2(x) \\
\log_1p(x) \\
\exp(x) \\
\expm1(x)
\end{align*}
\]

# trigonometric
\[x \leftarrow \text{bigfloat}(0.25)\]
\[
\cos(x) \\
\sin(x) \\
\tan(x) \\
\acos(x) \\
\asin(x) \\
\atan(x) \\
\cospi(x) \\
\sinpi(x) \\
\tanpi(x)
\]

# hyperbolic
\[x \leftarrow \text{bigfloat}(0.25)\]
\[
\cosh(x) \\
\sinh(x) \\
\tanh(x) \\
\acosh(\text{bigfloat}(2)) \\
\asinh(x) \\
\atanh(x)
\]

# special functions
\[x \leftarrow \text{bigfloat}(2.5)\]
\[
\gamma(x) \\
\lgamma(x) \\
\digamma(x) \\
\trigamma(x) \\
\factorial(x) \\
\lfactorial(x)
\]

---

**bignum-special**  
*Check for special values*

**Description**

`biginteger` and `bigfloat` support missing values (via `NA_biginteger_` and `NA_bigfloat_` respectively).

`bigfloat` additionally supports positive and negative infinity and 'Not a Number' values. Usually these are the result of a calculation, but they can also be created manually by casting from `numeric` to `bigfloat`.

These functions check for the presence of these special values. The base R documentation can be found at `is.na()` and `is.finite()`.
Value

A logical vector.

See Also

Other bignum operations: `bignum-arith`, `bignum-compare`, `bignum-math`

Examples

```r
x <- bigfloat(c(0, NA, Inf, -Inf, NaN))

is.na(x)
is.finite(x)
is.infinite(x)
is.nan(x)
```

Description

Generate a regular sequence of `biginteger` or `bigfloat` values.

When calling `seq()`, exactly two of the following must be specified:

• to
• by
• Either `length.out` or `along.with`

Usage

```r
## S3 method for class 'bignum_vctr'
seq(from, to = NULL, by = NULL, length.out = NULL, along.with = NULL, ...)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>from</td>
<td>Start value of the sequence. Always included in the result. A <code>biginteger</code> or <code>bigfloat</code> scalar.</td>
</tr>
<tr>
<td>to</td>
<td>Stop value of the sequence. Only included in the result if <code>by</code> divides the interval between <code>from</code> and <code>to</code> exactly. <code>to</code> is cast to the type of <code>from</code>.</td>
</tr>
<tr>
<td>by</td>
<td>Amount to increment the sequence by. <code>by</code> is cast to the type of <code>from</code>.</td>
</tr>
<tr>
<td>length.out</td>
<td>Length of the resulting sequence.</td>
</tr>
<tr>
<td>along.with</td>
<td>Vector who’s length determines the length of the resulting sequence.</td>
</tr>
<tr>
<td>...</td>
<td>These dots are for future extensions and must be empty.</td>
</tr>
</tbody>
</table>
**seq.bignum_vctr**

**Value**

A sequence with the type of from.

**Examples**

```
seq(biginteger(0), 10, by = 2)
seq(biginteger(0), 10, length.out = 3)
seq(biginteger(0), by = 3, length.out = 3)
seq(bigfloat(0), by = -0.05, length.out = 6)
```
Index

* **bignum operations**
  - bignum-arith, 4
  - bignum-compare, 5
  - bignum-math, 8
  - bignum-special, 9

* **datasets**
  - bignum-constants, 5

Arithmetic, 4
as_bigfloat (bigfloat), 2
as_biginteger (biginteger), 3

bigfloat, 2, 4–10
biginteger, 3, 4–10
bignum-arith, 4
bignum-compare, 5
bignum-constants, 5
bignum-format, 6
bignum-math, 8
bignum-special, 9
bigpi (bignum-constants), 5

Comparison, 5

format(), 2, 3
format.bignum_bigfloat (bignum-format), 6
format.bignum_biginteger (bignum-format), 6

is.finite(), 9
is.na(), 9
is_bigfloat (bigfloat), 2
is_biginteger (biginteger), 3

mean(), 8

NA, 6
NA_bigfloat_, 2
NA_bigfloat_ (bignum-constants), 5
NA_biginteger_, 3

NA_biginteger_ (bignum-constants), 5
numeric, 9
pi, 5, 6
seq.bignum_vctr, 10