

# Package ‘Brobdingnag’

May 6, 2026

**Type** Package

**Title** Very Large Numbers in R

**Version** 1.2-9

**Maintainer** Robin K. S. Hankin <hankin.robin@gmail.com>

**Depends** R (>= 2.13.0), methods, Matrix (>= 1.5-0)

**Description** Very large numbers in R. Real numbers are held using their natural logarithms, plus a logical flag indicating sign. Functionality for complex numbers is also provided. The package includes a vignette that gives a step-by-step introduction to using S4 methods.

**Suggests** cubature, testthat

**License** GPL

**Repository** CRAN

**URL** <https://github.com/RobinHankin/Brobdingnag>

**NeedsCompilation** no

**Author** Robin K. S. Hankin [aut, cre] (ORCID:  
<<https://orcid.org/0000-0001-5982-0415>>)

**Date/Publication** 2022-10-19 10:50:02 UTC

## Contents

Brobdingnag-package . . . . .	2
Arith-methods . . . . .	4
as.numeric . . . . .	5
brob . . . . .	6
brob-class . . . . .	7
brobmat . . . . .	8
brobmat-class . . . . .	9
brobmat.mult . . . . .	11
cbrob . . . . .	12
Compare-methods . . . . .	13

Complex . . . . .	13
Extract.brob . . . . .	14
getP . . . . .	15
glub . . . . .	16
glub-class . . . . .	17
index-class . . . . .	18
infinite-methods . . . . .	19
length-methods . . . . .	20
Logic . . . . .	20
Math . . . . .	21
plot . . . . .	21
Print . . . . .	22
sum . . . . .	23
swift-class . . . . .	24
<b>Index</b>	<b>25</b>

---

Brobdingnag-package    *Very Large Numbers in R*

---

## Description

Very large numbers in R. Real numbers are held using their natural logarithms, plus a logical flag indicating sign. Functionality for complex numbers is also provided. The package includes a vignette that gives a step-by-step introduction to using S4 methods.

## Details

The DESCRIPTION file:

```

Package:      Brobdingnag
Type:        Package
Title:       Very Large Numbers in R
Version:     1.2-9
Authors@R:   person(given=c("Robin", "K. S."), family="Hankin", role = c("aut","cre"), email="hankin.robin@gmail.com",
Maintainer:  Robin K. S. Hankin <hankin.robin@gmail.com>
Depends:     R (>= 2.13.0), methods, Matrix (>= 1.5-0)
Description: Very large numbers in R. Real numbers are held using their natural logarithms, plus a logical flag indicating sign.
Suggests:   cubature, testthat
License:     GPL
Repository:  CRAN
URL:        https://github.com/RobinHankin/Brobdingnag
Author:     Robin K. S. Hankin [aut, cre] (<https://orcid.org/0000-0001-5982-0415>)

```

Index of help topics:

Arith-methods	Methods for Function Arith in package Brobdingnag
Brobdingnag-package	Very Large Numbers in R
Compare-methods	Methods for Function Compare in Package Brobdingnag
Re	Real and imaginary manipulation
[.brob	Extract or Replace Parts of brobs or glubs
abs	Various logarithmic and circular functions for brobs
as.numeric	Coerces to numeric or complex form
brob	Brobdingnagian numbers
brob-class	Class "brob"
brobmat	Brobdingnagian matrices
brobmat-class	Class "'brobmat''
brobmat.mult	Brobdingagian matrix arithmetic
cbrob	Combine Brobdingnagian vectors
getP	Get and set methods for brob objects
glub	Glubbdubdribian numbers: complex numbers with Brobdingnagian real and imaginary parts
glub-class	Class "glub"
index-class	Class "'index''
infinite	Infinite brobs and glubs
length	Get lengths of brobs and glubs
logic.brob	Logical operations on brobs
plot	Basic plotting of Brobs
print.brob	Methods for printing brobs and glubs
sum	Various summary statistics for brobs and glubs
swift-class	Class "swift"

Real numbers are represented by two objects: a real, holding the logarithm of their absolute values; and a logical, indicating the sign. Multiplication and exponentiation are easy: the challenge is addition. This is achieved using the (trivial) identity  $\log(e^x + e^y) = x + \log(1 + e^{y-x})$  where, WLOG,  $y < x$ .

Complex numbers are stored as a pair of brobs: objects of class glub.

The package is a simple example of S4 methods.

However, it *could* be viewed as a cautionary tale: the underlying R concepts are easy yet the S4 implementation is long and difficult. I would not recommend using S4 methods for a package as simple as this; S3 methods would have been perfectly adequate. I would suggest that S4 methods should only be used when S3 methods are *demonstrably* inadequate.

#### Author(s)

NA

Maintainer: Robin K. S. Hankin <hankin.robin@gmail.com>

**References**

R. K. S. Hankin 2007. “Very Large Numbers in R: Introducing Package Brobdingnag”. R News, volume 7, number 3, pages 15-16

**Examples**

```
googol <- as.brob(10)^100

googol
googol + googol/2

1/(googol + 1e99)

(1:10)^googol

googolplex <- 10^googol

googolplex

googolplex * googol # practically the same as googolplex (!)
```

---

Arith-methods

*Methods for Function Arith in package Brobdingnag*

---

**Description**

Methods for Arithmetic functions in package Brobdingnag: +, -, \*, /, ^

**Note**

The unary arithmetic functions (viz “+” and “-”) do no coercion.

The binary arithmetic functions coerce numeric <op> brob to brob; and numeric <op> glub, complex <op> brob, and brob <op> glub, to glub.

**Author(s)**

Robin K. S. Hankin

**Examples**

```
x <- as.brob(1:10)
y <- 1e10

x+y

as.numeric((x+y)-1e10)
```

$x^{(1/y)}$ 

---

`as.numeric`*Coerces to numeric or complex form*

---

**Description**

Coerces an object of class brob to numeric, or an object of class glub to complex

**Arguments**

`x`                    Object of class brob or glub  
`...`                  Further arguments (currently ignored)

**Details**

Function `as.numeric()` coerces a brob to numeric; if given a glub, the imaginary component is ignored (and a warning given).

Function `as.complex()` coerces to complex.

**Note**

If  $|x|$  is greater than `.Machine$double.xmax`, then `as.numeric(x)` returns `Inf` or `-Inf` but no warning is given.

**Author(s)**

Robin K. S. Hankin

**Examples**

```
a <- as.brob(1:10)
a <- cbrob(a, as.brob(10)^1e26)
a
as.numeric(a)

as.complex(10i + a)
```

---

brob	<i>Brobdingnagian numbers</i>
------	-------------------------------

---

### Description

Create, coerce to or test for a Brobdingnagian object

### Usage

```
brob(x = double(), positive)
as.brob(x)
is.brob(x)
```

### Arguments

x	Quantity to be tested, coerced in to Brobdingnagian form
positive	In function brob(), logical indicating whether the number is positive (actually, positive or zero)

### Details

Function `as.brob()` is the user's workhorse: use this to coerce numeric vectors to brobs.

Function `is.brob()` tests for its arguments being of class brob.

Function `brob()` takes argument `x` and returns a brob formally equal to  $e^x$ ; set argument `positive` to `FALSE` to return  $-e^x$ . Thus calling function `exp(x)` simply returns `brob(x)`. This function is not really intended for the end user: it is confusing and includes no argument checking. In general numerical work, use function `as.brob()` instead, although be aware that if you really really want  $e^{10^7}$ , you should use `brob(1e7)`; this would be an **exact** representation.

### Note

Real numbers are represented by two objects: a real, holding the logarithm of their absolute values; and a logical, indicating the sign. Multiplication and exponentiation are easy: the challenge is addition. This is achieved using the (trivial) identity  $\log(e^x + e^y) = x + \log(1 + e^{y-x})$  where, WLOG,  $y < x$ .

Complex numbers are stored as a pair of brobs: objects of class `glub`.

The package is a simple example of S4 methods. However, it *could* be viewed as a cautionary tale: the underlying R concepts are easy yet the S4 implementation is long and difficult. I would not recommend using S4 methods for a package as simple as this; S3 methods would have been perfectly adequate. I would suggest that S4 methods should only be used when S3 methods are *demonstrably* inadequate.

The package has poor handling of NA and NaN. Currently, `as.brob(1) + as.brob(c(1, NA))` returns an error.

**Author(s)**

Robin K. S. Hankin

**See Also**

[glub](#)

**Examples**

```
googol <- as.brob(10)^100
googolplex <- 10^googol

(googolplex/googol) / googolplex
# Thus googolplex/googol == googolplex (!)

# use cbrob() instead of c() when Brobdingnagian numbers are involved:
cbrob(4,exp(as.brob(1e55)))
```

---

brob-class

Class "brob"

---

**Description**

The formal S4 class for Brobdingnagian numbers

**Objects from the Class**

Objects *can* be created by calls of the form `new("brob", ...)` but this is not encouraged. Use functions `brob()` and, especially, `as.brob()` instead.

**Slots**

**x:** Object of class "numeric" holding the log of the absolute value of the number to be represented  
**positive:** Object of class "logical" indicating whether the number is positive (see Note, below)

**Extends**

Class "swift", directly.

**Note**

Slot `positive` indicates non-negativity, as zero is conventionally considered to be "positive".

**Author(s)**

Robin K. S. Hankin

**See Also**

[glub-class](#), [swift-class](#)

**Examples**

```
new("brob",x=5,positive=TRUE) # not intended for the user
as.brob(5) # Standard user-oriented idiom
```

---

brobmat

*Brobdingnagian matrices*

---

**Description**

Basic matrix arithmetic for Brobdingnagian numbers. Matrix addition, multiplication extraction and replacement implemented but not the determinant or matrix inverse.

**Usage**

```
brobmat(..., positive)
newbrobmat(x,positive)
as.brobmat(x)
is.brobmat(x)
brobmat_to_brob(x)
diag(x,...)
## S3 method for class 'brobmat'
print(x,...)
t(x,...)
```

**Arguments**

x	Argument
...	Further arguments
positive	Logical, indicating whether an element is positive

**Details**

Basic arithmetic for Brobdingnagian matrices.

Function `brobmat()` is like `brob()` in that it interprets its first argument as the exponent (but creates a matrix). Function `as.brobmat()` coerces a numeric matrix to a `brobmat`.

**Value**

Generally return a `brobmat` or `brob`.

**Author(s)**

Robin K. S. Hankin

**Examples**

```
brobmat(-10:19,5,6)
as.brobmat(matrix(-10:19,5,6))
```

---

brobmat-class	<i>Class "brobmat"</i>
---------------	------------------------

---

**Description**

The brobmat class provides basic Brobdingnagian arithmetic for matrices.

**Objects from the Class**

Objects can be created by calls of the form `new("brobmat", ...)`, although functions `brobmat()`, `as.brobmat()` are more user-friendly.

**Slots**

**x:** Object of class "matrix" that specifies the exponent  
**positive:** Object of class "logical" that specifies the sign

**Methods**

```
[ signature(x = "brobmat", i = "ANY", j = "ANY"): ...
[ signature(x = "brobmat", i = "index", j = "index"): ...
[ signature(x = "brobmat", i = "index", j = "missing"): ...
[ signature(x = "brobmat", i = "missing", j = "index"): ...
[ signature(x = "brobmat", i = "missing", j = "missing"): ...
[ signature(x = "brobmat", i = "matrix", j = "missing"): ...
[<- signature(x = "brobmat", i = "index", j = "index"): ...
[<- signature(x = "brobmat", i = "index", j = "missing"): ...
[<- signature(x = "brobmat", i = "missing", j = "index"): ...
[<- signature(x = "brobmat", i = "matrix", j = "missing"): ...
[<- signature(x = "brobmat", i = "missing", j = "missing"): ...
%% signature(x = "ANY", y = "brobmat"): ...
%% signature(x = "brobmat", y = "ANY"): ...
%% signature(x = "brobmat", y = "brobmat"): ...
Arith signature(e1 = "ANY", e2 = "brobmat"): ...
```

**Arith** signature(e1 = "brob", e2 = "brobmat"): ...  
**Arith** signature(e1 = "brobmat", e2 = "ANY"): ...  
**Arith** signature(e1 = "brobmat", e2 = "brob"): ...  
**Arith** signature(e1 = "brobmat", e2 = "brobmat"): ...  
**Arith** signature(e1 = "brobmat", e2 = "missing"): ...  
**as.matrix** signature(x = "brobmat"): ...  
**as.vector** signature(x = "brobmat"): ...  
**coerce** signature(from = "brobmat", to = "matrix"): ...  
**colnames** signature(x = "brobmat"): ...  
**colnames<-** signature(x = "brobmat"): ...  
**Compare** signature(e1 = "ANY", e2 = "brobmat"): ...  
**Compare** signature(e1 = "brobmat", e2 = "ANY"): ...  
**Compare** signature(e1 = "brobmat", e2 = "brobmat"): ...  
**diag** signature(x = "brobmat"): ...  
**dimnames** signature(x = "brobmat"): ...  
**dimnames<-** signature(x = "brobmat"): ...  
**getP** signature(x = "brobmat"): ...  
**getX** signature(x = "brobmat"): ...  
**length** signature(x = "brobmat"): ...  
**Math** signature(x = "brobmat"): ...  
**ncol** signature(x = "brobmat"): ...  
**nrow** signature(x = "brobmat"): ...  
**rownames** signature(x = "brobmat"): ...  
**rownames<-** signature(x = "brobmat"): ...  
**show** signature(object = "brobmat"): ...  
**t** signature(x = "brobmat"): ...

### Author(s)

Robin K. S. Hankin

### References

Brobdingnag R News paper

### See Also

[as.brob](#), [brob](#)

### Examples

```
showClass("brobmat")
```

---

brobmat.mult	<i>Brobdingagian matrix arithmetic</i>
--------------	--

---

## Description

Basic arithmetic for Brobdingnagian matrices

## Usage

```
brobmat.mult(e1, e2)
brobmat.add(e1, e2)
brobmat.mult(e1, e2)
brobmat.power(e1, e2)
brobmat.inverse(e1)
brobmat.greater(e1, e2)
brobmat.equal(e1, e2)
getat(e1,e2)
```

## Arguments

e1, e2                    Arguments coerced to brobmat

## Details

These functions are helper functions used by the brobmat Arith group and are not designed to be user-friendly. Function `getat()` is a helper function that sets attributes such as `dimnames` of returned values.

## Value

Return a brobmat, or logical for the comparison operators.

## Author(s)

Robin K. S. Hankin

## Examples

```
a <- brobmat(1:54,6,9)
rownames(a) <- letters[1:6]
a + 1e30
a-a

b <- as.brobmat(matrix(rnorm(27),9,3))
colnames(b) <- month.abb[1:3]

a %**% b
```

---

`cbrob`*Combine Brobdingnagian vectors*

---

**Description**

Combine Brobdingnagian or Glubdubbdruban vectors through concatenation

**Usage**

```
cbrob(x, ...)
```

**Arguments**

<code>x</code>	Brobdingnagian vector
<code>...</code>	Other arguments coerced to brob form

**Details**

If any argument has class `glub`, all arguments are coerced to `glubs`. Otherwise, if any argument has class `brob`, all arguments are coerced to `brobs`.

Function `cbrob()` operates recursively, calling `.cPair()` repeatedly. Function `.cPair()` uses S4 method dispatch to call either `.Brob.cpair()` or `.Glub.cpair()` according to the classes of the arguments.

**Note**

As of R-2.4.0, it is apparently not possible to use S4 methods to redefine `c()` to coerce to class `brob` form and concatenate as expected. This would seem to be a reasonable interpretation of `c()` from the user's perspective.

Conceptually, the operation is simple: concatenate the value slot and the positive slot separately, then call `brob()` on the two resulting vectors. When concatenating `glub` objects, the real and imaginary components (being `brobs`) are concatenated using `.Brob.cpair()`

The choice of name—`cbrob()`—is not entirely logical. Because it operates consistently on `brob` and `glub` objects, it might be argued that `cSwift()` would be a more appropriate name.

**Author(s)**

Robin K. S. Hankin; original idea due to John Chambers

**Examples**

```
a <- as.brob(2)^1e-40
cbrob(1:4, 4:1, a)
cbrob(1:4, a, 1i)
```

**Description**

Methods for comparison (greater than, etc) in package Brobdingnag

**Note**

As for `min()` and `max()`, comparison is not entirely straightforward in the presence of NAs.

The low-level workhorses are `.Brob.equal()` for equality and `.Brob.greater()` for 'strictly greater than'. All other comparisons are calculated by combining these two.

Comparison [function `.Brob.compare()`] explicitly tests for a zero length argument and if given one returns `logical(0)` to match base behaviour.

**Examples**

```
a <- as.brob(10)^(0.5 + 97:103)
a < 1e100
```

**Description**

Get or set real and imaginary components of brobs or glubs.

**Usage**

```
## S4 method for signature 'glub'
Re(z)
## S4 method for signature 'glub'
Im(z)
## S4 method for signature 'glub'
Mod(z)
## S4 method for signature 'glub'
Conj(z)
## S4 method for signature 'glub'
Arg(z)
Re(z) <- value
Im(z) <- value
```

**Arguments**

z                    object of class `glub` (or, in the case of `Im<-( )` or `Im(z) <- value`, class `brob`)  
 value                object of class `numeric` or `brob`

**Value**

Functions `Re( )` and `Im( )` return an object of class `brob`; functions `Re<-( )` and `Im<-( )` return an object of class `glub`

**Author(s)**

Robin K. S. Hankin

**Examples**

```
a <- cbrob(1:10, brob(1e100))
Im(a) <- 11:1
a
```

---

Extract.brob

*Extract or Replace Parts of brobs or glubs*

---

**Description**

Methods for "`[`" and "`[<-`", i.e., extraction or subsetting of brobs and glubs.

**Arguments**

x                    Object of class `brob` or `glub`  
 i                    elements to extract or replace  
 value                replacement value

**Value**

Always returns an object of the same class as `x`.

**Note**

If `x` is a numeric vector and `y` a brob, one might expect typing `x[1] <- y` to result in `x` being a brob. This is impossible, according to John Chambers.

**Author(s)**

Robin K. S. Hankin

**Examples**

```
a <- as.brob(10)^c(-100,0,100,1000,1e32)

a[4]

a[4] <- 1e100

a
```

---

getP

*Get and set methods for brob objects*

---

**Description**

Get and set methods for brobs: sign and value

**Usage**

```
getP(x)
getX(x)
sign(x) <- value
```

**Arguments**

x	Brobdingnagian object
value	In function sign<-(), Boolean specifying whehter the brob object is positive

**Author(s)**

Robin K. S. Hankin

**See Also**

[brob](#)

**Examples**

```
x <- as.brob(-10:10)
sign(x) <- TRUE
```

---

glub	<i>Glubbudribian numbers: complex numbers with Brobdingnagian real and imaginary parts</i>
------	--

---

## Description

Create, coerce to or test for a Glubbudribian object

## Usage

```
glub(real = double(), imag = double())  
as.glub(x)  
is.glub(x)
```

## Arguments

real, imag	Real and imaginary components of complex number: must be Brobdingnagian numbers
x	object to be coerced to or tested for Glubbudribian form

## Details

A *Glubbudribian* number is the Brobdingnagian equivalent of a complex number.

Function `glub()` takes two arguments that are coerced to Brobdingnagian numbers and returns a Glubbudribian number. This function is not really intended for the end user: it is confusing and includes no argument checking. Use function `as.glub()` instead.

Function `as.glub()` is the user's workhorse: use this to coerce numeric or complex vectors to Glubbudribian form.

Function `is.glub()` tests for its arguments being Glubbudribian.

## Note

Function `glub()` uses recycling inherited from `cbind()`.

## Author(s)

Robin K. S. Hankin

## See Also

[brob](#)

**Examples**

```

a <- as.glub(1:10 + 5i)
a^2 - a*a

f <- function(x){sin(x) +x^4 - 1/x}
as.complex(f(a)) - f(as.complex(a)) # should be zero (in the first
# term, f() works with glubs and coerces to
# complex; in the second, f()
# works with complex numbers directly)

```

glub-class

*Class "glub"***Description**

Complex Brobdingnagian numbers

**Objects from the Class**

A glub object holds two slots, both brobs, representing the real and imaginary components of a complex vector.

**Slots**

**real:** Object of class "brob" representing the real component  
**imag:** Object of class "brob" representing the imaginary component

**Extends**

Class "swift", directly.

**Methods**

**.cPair** signature(x = "brob", y = "glub"): ...  
**.cPair** signature(x = "ANY", y = "glub"): ...  
**.cPair** signature(x = "glub", y = "glub"): ...  
**.cPair** signature(x = "glub", y = "ANY"): ...  
**.cPair** signature(x = "glub", y = "brob"): ...  
**Im<-** signature(x = "glub"): ...  
**Re<-** signature(x = "glub"): ...

**Author(s)**

Robin K. S. Hankin

**See Also**

[brob-class](#), [swift-class](#)

**Examples**

```
a <- as.brob(45)
new("glub", real=a, imag=a)

as.brob(5+5i) # standard R idiom; imaginary component discarded
as.glub(5+5i) # returns a Glubbubdribian object
```

---

index-class

*Class "index"*

---

**Description**

A virtual class for matrix extraction, copied from the `Matrix` package.

**Objects from the Class**

A virtual Class: No objects may be created from it.

**Methods**

```
[ signature(x = "brobmat", i = "index", j = "index"): ...
[ signature(x = "brobmat", i = "index", j = "missing"): ...
[ signature(x = "brobmat", i = "missing", j = "index"): ...
[<- signature(x = "brobmat", i = "index", j = "index"): ...
[<- signature(x = "brobmat", i = "index", j = "missing"): ...
[<- signature(x = "brobmat", i = "missing", j = "index"): ...
```

**Author(s)**

Bates and Maechler, I guess

**References**

Douglas Bates and Martin Maechler (2019). `Matrix: Sparse and Dense Matrix Classes and Methods`. R package version 1.2-18. <https://CRAN.R-project.org/package=Matrix>

**See Also**

[brobmat](#)

**Examples**

```
showClass("index")
```

---

infinite-methods      *Infinite brobs and glubs*

---

## Description

Brobdingnagian and Glubbudbribian infinity

## Usage

```
## S4 method for signature 'brob'  
is.infinite(x)  
## S4 method for signature 'glub'  
is.infinite(x)  
## S4 method for signature 'brob'  
is.finite(x)  
## S4 method for signature 'glub'  
is.finite(x)
```

## Arguments

x                      vector of class brob or glub

## Details

For a Brobdingnagian number, `is.infinite()` returns TRUE if the exponent is infinite.  
A Glubbudbribian number is infinite if either the real or imaginary component is infinite.  
Function `is.finite()` is simply the logical negation of `is.infinite()`.

## Author(s)

Robin K. S. Hankin

## Examples

```
is.infinite(brob(c(1,4,Inf)))  
  
is.infinite(glub(3,Inf))  
is.infinite(glub(Inf,3))  
  
is.infinite(exp(1e300))  
is.infinite(brob(1e300))  
# (Brobdingnagian infinity is bigger than regular infinity ;-)
```

length-methods

*Get lengths of brobs and glubs*

---

**Description**

Get lengths of brob and glub vectors

**Usage**

```
## S4 method for signature 'brob'  
length(x)  
## S4 method for signature 'glub'  
length(x)
```

**Arguments**

x                    vector of class brob or glub

**Author(s)**

Robin K. S. Hankin

**Examples**

```
x <- as.brob(-10:10)  
length(x)
```

---

Logic

*Logical operations on brobs*

---

**Description**

Logical operations on brobs are not supported

**Note**

The S4 group generic “Logic” appeared in R-2.4.0-patched.

Carrying out logical operations in this group will call `.Brob.logic()`, which reports an error.

Negation, “!”, is not part of this group: attempting to negate a brob will not activate `.Brob.logic()`; an “invalid argument type” error is given instead.

**Author(s)**

Robin K. S. Hankin

**Examples**

```
## Not run:
!brob(10)

## End(Not run)
```

---

 Math

*Various logarithmic and circular functions for brobs*


---

**Description**

Various elementary functions for brobs

**Arguments**

x	Object of class brob (or sometimes glub)
base	In function log(), the base of the logarithm

**Details**

For brobs: apart from abs(), log(), exp(), sinh() and cosh(), these functions return f(as.numeric(x)) so are numeric; the exceptional functions return brobs.

For glubs: mostly direct transliteration of the appropriate formula; one might note that log(z) is defined as glub(log(Mod(x)), Arg(x)).

**Author(s)**

Robin K. S. Hankin

**Examples**

```
exp(as.brob(3000)) #exp(3000) is represented with zero error
```

---

 plot

*Basic plotting of Brobs*


---

**Description**

Plotting methods. Essentially, any brob is coerced to a numeric and any glub is coerced to a complex, and the argument or arguments are passed to plot().

**Usage**

```
plot(x, y, ...)
```

**Arguments**

x, y	Brob or glub
...	Further arguments passed to plot()

**Author(s)**

Robin K. S. Hankin

**Examples**

```
plot(as.brob(1:10))
```

---

Print

*Methods for printing brobs and glubs*

---

**Description**

Methods for printing brobs and glubs nicely using exponential notation

**Usage**

```
## S3 method for class 'brob'  
print(x, ...)  
## S3 method for class 'glub'  
print(x, ...)
```

**Arguments**

x	An object of class brob or glub
...	Further arguments (currently ignored)

**Author(s)**

Robin K. S. Hankin

**Examples**

```
a <- as.brob(1:5)  
dput(a)  
a
```

---

sum

*Various summary statistics for brobs and glubs*

---

## Description

Various summary statistics for brobs and glubs

## Arguments

<code>x, ...</code>	Objects of class brob or, in the case of <code>sum()</code> and <code>prob()</code> , class glub
<code>na.rm</code>	Boolean, with default FALSE meaning to interpret NAs literally and TRUE meaning to ignore any such elements

## Details

For a brob object, being NA is not entirely straightforward. The S4 method for `is.na` is too “strict” for some of the functions considered here. Consider `max(a)` where `a` includes only positive, fully specified, elements, and elements with known negative sign and exponents that include NA values. Here, `max(a)` is unambiguously determined.

Similar logic applies to `min()` and, by extension, `range()`.

## Note

Function `prod()` is *very* slow for long glub vectors. It has to compute four Brobdingnagian products and two Brobdingnagian sums per element of its argument, and this takes a long time.

## Author(s)

Robin K. S. Hankin

## See Also

[is.na](#)

## Examples

```
a <- as.brob(1:10)
max(cbrob(1:10, brob(NA, FALSE)))
```

---

`swift-class`*Class "swift"*

---

**Description**

A (virtual) class that extends brob and glub objects

**Objects from the Class**

A virtual Class: No objects may be created from it.

**Methods**

No methods defined with class "swift" in the signature.

**Author(s)**

Robin K. S. Hankin

**See Also**

[brob-class](#), [glub-class](#)

# Index

## \* classes

- brob-class, 7
- brobmat-class, 9
- glub-class, 17
- index-class, 18
- swift-class, 24

## \* math

- Arith-methods, 4
- as.numeric, 5
- brob, 6
- cbrob, 12
- Compare-methods, 13
- Complex, 13
- Extract.brob, 14
- getP, 15
- glub, 16
- infinite-methods, 19
- length-methods, 20
- Logic, 20
- Math, 21
- plot, 21
- Print, 22
- sum, 23

## \* methods

- Arith-methods, 4
- Compare-methods, 13
- infinite-methods, 19
- length-methods, 20

## \* package

- Brobdingnag-package, 2
- .cPair, ANY, ANY-method (brob-class), 7
- .cPair, ANY, brob-method (brob-class), 7
- .cPair, ANY, glub-method (glub-class), 17
- .cPair, brob, ANY-method (brob-class), 7
- .cPair, brob, brob-method (brob-class), 7
- .cPair, brob, complex-method (brob-class), 7
- .cPair, brob, glub-method (glub-class), 17
- .cPair, complex, brob-method

- (brob-class), 7
- .cPair, glub, ANY-method (glub-class), 17
- .cPair, glub, brob-method (glub-class), 17
- .cPair, glub, glub-method (glub-class), 17
- [, brob, ANY, ANY, ANY-method (Extract.brob), 14
- [, brob, ANY, ANY-method (Extract.brob), 14
- [, brob-method (Extract.brob), 14
- [, brobmat, ANY, ANY, ANY-method (Extract.brob), 14
- [, brobmat, ANY, ANY-method (brobmat-class), 9
- [, brobmat, index, index, ANY-method (Extract.brob), 14
- [, brobmat, index, index-method (brobmat-class), 9
- [, brobmat, index, missing, ANY-method (Extract.brob), 14
- [, brobmat, index, missing-method (brobmat-class), 9
- [, brobmat, matrix, missing, ANY-method (Extract.brob), 14
- [, brobmat, matrix, missing-method (brobmat-class), 9
- [, brobmat, missing, index, ANY-method (Extract.brob), 14
- [, brobmat, missing, index-method (brobmat-class), 9
- [, brobmat, missing, missing, ANY-method (Extract.brob), 14
- [, brobmat, missing, missing-method (brobmat-class), 9
- [, glub, ANY, ANY, ANY-method (Extract.brob), 14
- [, glub, ANY, ANY-method (Extract.brob), 14
- [, glub-method (Extract.brob), 14
- [.brob (Extract.brob), 14
- [.glub (Extract.brob), 14
- [<- , brob, ANY, ANY, ANY-method

- (Extract.brob), 14
- [<- , brob, ANY, ANY-method (Extract.brob), 14
- [<- , brob-method (Extract.brob), 14
- [<- , brobmat, index, index, ANY-method (Extract.brob), 14
- [<- , brobmat, index, index-method (brobmat-class), 9
- [<- , brobmat, index, missing, ANY-method (Extract.brob), 14
- [<- , brobmat, index, missing-method (brobmat-class), 9
- [<- , brobmat, matrix, missing, ANY-method (Extract.brob), 14
- [<- , brobmat, matrix, missing-method (brobmat-class), 9
- [<- , brobmat, missing, index, ANY-method (Extract.brob), 14
- [<- , brobmat, missing, index-method (brobmat-class), 9
- [<- , brobmat, missing, missing, ANY-method (Extract.brob), 14
- [<- , brobmat, missing, missing-method (brobmat-class), 9
- [<- , glub, ANY, ANY, ANY-method (Extract.brob), 14
- [<- , glub, ANY, ANY-method (Extract.brob), 14
- [<- , glub-method (Extract.brob), 14
- [<- .brob (Extract.brob), 14
- [<- .glub (Extract.brob), 14
- %%, ANY, brobmat-method (brobmat-class), 9
- %%, brobmat, ANY-method (brobmat-class), 9
- %%, brobmat, brobmat-method (brobmat-class), 9
- abs (Math), 21
- acos (Math), 21
- acosh (Math), 21
- Arg (Complex), 13
- Arg, brob-method (Complex), 13
- Arg, glub-method (Complex), 13
- Arith, ANY, brob-method (Arith-methods), 4
- Arith, ANY, brobmat-method (brobmat-class), 9
- Arith, ANY, glub-method (Arith-methods), 4
- Arith, brob, ANY-method (Arith-methods), 4
- Arith, brob, brob-method (Arith-methods), 4
- Arith, brob, brobmat-method (brobmat-class), 9
- Arith, brob, complex-method (Arith-methods), 4
- Arith, brob, glub-method (Arith-methods), 4
- Arith, brob, missing-method (Arith-methods), 4
- Arith, brobmat, ANY-method (brobmat-class), 9
- Arith, brobmat, brob-method (brobmat-class), 9
- Arith, brobmat, brobmat-method (brobmat-class), 9
- Arith, brobmat, missing-method (brobmat-class), 9
- Arith, complex, brob-method (Arith-methods), 4
- Arith, complex, glub-method (Arith-methods), 4
- Arith, glub, ANY-method (Arith-methods), 4
- Arith, glub, brob-method (Arith-methods), 4
- Arith, glub, complex-method (Arith-methods), 4
- Arith, glub, glub-method (Arith-methods), 4
- Arith, glub, missing-method (Arith-methods), 4
- Arith-methods, 4
- as.brob, 10
- as.brob (brob), 6
- as.brobmat (brobmat), 8
- as.complex (as.numeric), 5
- as.complex, brob-method (as.numeric), 5
- as.complex, glub-method (as.numeric), 5
- as.glub (glub), 16
- as.matrix, brobmat-method (brobmat-class), 9
- as.numeric, 5
- as.numeric, brob-method (as.numeric), 5
- as.numeric, glub-method (as.numeric), 5
- as.vector, brobmat-method (brobmat-class), 9
- asin (Math), 21
- asinh (Math), 21

- atan (Math), 21
- atanh (Math), 21
- brob, 6, 10, 15, 16
- brob-class, 7
- Brobdingnag (Brobdingnag-package), 2
- Brobdingnag-package, 2
- brobmat, 8, 18
- brobmat-class, 9
- brobmat.add (brobmat.mult), 11
- brobmat.arith (brobmat.mult), 11
- brobmat.compare (brobmat.mult), 11
- brobmat.equal (brobmat.mult), 11
- brobmat.greater (brobmat.mult), 11
- brobmat.inverse (brobmat.mult), 11
- brobmat.mult, 11
- brobmat.power (brobmat.mult), 11
- brobmat\_matrixprod (brobmat.mult), 11
- brobmat\_to\_brob (brobmat), 8
- cBrob (cbrob), 12
- cbrob, 12
- ceiling (Math), 21
- coerce, brob, complex-method (as.numeric), 5
- coerce, brob, numeric-method (as.numeric), 5
- coerce, brobmat, matrix-method (brobmat-class), 9
- coerce, glub, complex-method (as.numeric), 5
- coerce, glub, numeric-method (as.numeric), 5
- colnames, brobmat-method (brobmat-class), 9
- colnames<-, brobmat-method (brobmat-class), 9
- Compare, ANY, brob-method (Compare-methods), 13
- Compare, ANY, brobmat-method (brobmat-class), 9
- Compare, ANY, glub-method (Compare-methods), 13
- Compare, brob, ANY-method (Compare-methods), 13
- Compare, brob, brob-method (Compare-methods), 13
- Compare, brob, glub-method (Compare-methods), 13
- Compare, brobmat, ANY-method (brobmat-class), 9
- Compare, brobmat, brobmat-method (brobmat-class), 9
- Compare, glub, ANY-method (Compare-methods), 13
- Compare, glub, brob-method (Compare-methods), 13
- Compare, glub, glub-method (Compare-methods), 13
- Compare-methods, 13
- Complex, 13
- Complex, brob-method (Complex), 13
- Complex, glub-method (Complex), 13
- Complex-methods (Complex), 13
- Conj (Complex), 13
- Conj, brob-method (Complex), 13
- Conj, glub-method (Complex), 13
- cos (Math), 21
- cosh (Math), 21
- cumsum (Math), 21
- diag (brobmat), 8
- diag, ANY-method (brobmat-class), 9
- diag, brobmat-method (brobmat-class), 9
- dimnames, brobmat-method (brobmat-class), 9
- dimnames<-, brobmat, ANY-method (brobmat-class), 9
- dimnames<-, brobmat-method (brobmat-class), 9
- exp (Math), 21
- Extract.brob, 14
- finite (infinite-methods), 19
- floor (Math), 21
- gamma (Math), 21
- getat (brobmat.mult), 11
- getP, 15
- getP, ANY-method (brobmat-class), 9
- getP, brob-method (brob-class), 7
- getP, brobmat-method (brobmat-class), 9
- getP, numeric-method (brobmat-class), 9
- getX (getP), 15
- getX, ANY-method (brobmat-class), 9
- getX, brob-method (brob-class), 7
- getX, brobmat-method (brobmat-class), 9

- getX, numeric-method (brobmat-class), 9
- glub, 7, 16
- glub-class, 17
- Im (Complex), 13
- Im, brob-method (Complex), 13
- Im, glub-method (Complex), 13
- Im<- (Complex), 13
- Im<- , brob-method (Complex), 13
- Im<- , glub-method (Complex), 13
- index-class, 18
- infinite (infinite-methods), 19
- infinite-methods, 19
- is.brob (brob), 6
- is.brobmat (brobmat), 8
- is.finite (infinite-methods), 19
- is.finite, brob-method (infinite-methods), 19
- is.finite, glub-method (infinite-methods), 19
- is.glub (glub), 16
- is.infinite (infinite-methods), 19
- is.infinite, brob-method (infinite-methods), 19
- is.infinite, glub-method (infinite-methods), 19
- is.na, 23
- length (length-methods), 20
- length, brob-method (length-methods), 20
- length, brobmat-method (brobmat-class), 9
- length, glub-method (length-methods), 20
- length-methods, 20
- lgamma (Math), 21
- log (Math), 21
- Logic, 20
- Logic, ANY, swift-method (Logic), 20
- Logic, swift, ANY-method (Logic), 20
- Logic, swift, swift-method (Logic), 20
- logic.brob (Logic), 20
- Math, 21
- Math, brob-method (Math), 21
- Math, brobmat-method (brobmat-class), 9
- Math, glub-method (Math), 21
- max (sum), 23
- min (sum), 23
- Mod (Complex), 13
- Mod, brob-method (Complex), 13
- ncol, brobmat-method (brobmat-class), 9
- newbrobmat (brobmat), 8
- nrow, brobmat-method (brobmat-class), 9
- plot, 21
- plot, ANY, brob-method (plot), 21
- plot, ANY, glub-method (plot), 21
- plot, brob, ANY-method (plot), 21
- plot, brob, missing-method (plot), 21
- plot, brob-method (plot), 21
- plot, glub, ANY-method (plot), 21
- plot, glub, missing-method (plot), 21
- plot, glub-method (plot), 21
- Print, 22
- print.brob (Print), 22
- print.brobmat (brobmat), 8
- print.glub (Print), 22
- prod (sum), 23
- range (sum), 23
- Re (Complex), 13
- Re, brob-method (Complex), 13
- Re, glub-method (Complex), 13
- Re<- (Complex), 13
- Re<- , glub-method (Complex), 13
- rownames, brobmat-method (brobmat-class), 9
- rownames<- , brobmat-method (brobmat-class), 9
- show, brob-method (Print), 22
- show, brobmat-method (brobmat-class), 9
- show, glub-method (Print), 22
- sign<- (getP), 15
- sign<- , brob-method (brob-class), 7
- sin (Math), 21
- sinh (Math), 21
- sqrt (Math), 21
- sqrt, brob-method (Math), 21
- sqrt, glub-method (Math), 21
- sum, 23
- Summary, brob-method (sum), 23
- Summary, glub-method (sum), 23
- swift-class, 24
- t (brobmat), 8
- t, ANY-method (brobmat-class), 9

t, brobmat-method (brobmat-class), [9](#)  
tan (Math), [21](#)  
tanh (Math), [21](#)  
trunc (Math), [21](#)