

# Package ‘Brobdingnag’

March 19, 2018

**Type** Package

**Title** Very Large Numbers in R

**Version** 1.2-5

**Date** 2018-03-19

**Author** Robin K. S. Hankin

**Depends** R (>= 2.13.0), methods

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

**Description** Handles very large numbers in R. Real numbers are held using their natural logarithms, plus a logical flag indicating sign. The package includes a vignette that gives a step-by-step introduction to using S4 methods.

**LazyLoad** yes

**License** GPL

**Repository** CRAN

**NeedsCompilation** no

**Date/Publication** 2018-03-18 23:15:09 UTC

## R topics documented:

Brobdingnag-package . . . . .	2
Arith-methods . . . . .	3
as.numeric . . . . .	4
brob . . . . .	5
brob-class . . . . .	6
cbrob . . . . .	7
Compare-methods . . . . .	8
Complex . . . . .	8
Extract.brob . . . . .	9
getP . . . . .	10
glub . . . . .	11
glub-class . . . . .	12
length-methods . . . . .	13

Logic . . . . .	14
Math . . . . .	14
plot . . . . .	15
Print . . . . .	15
sum . . . . .	16
swift-class . . . . .	17

<b>Index</b>	<b>18</b>
--------------	-----------

---

Brobdingnag-package	<i>Brobdingnagian numbers: storing large numbers by their natural logarithms</i>
---------------------	--

---

## Description

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.

## Details

Package:	Brobdingnag
Type:	Package
Version:	1.0-1
Date:	2006-09-21
License:	GPL

The user should coerce numeric vectors to brobs using `as.brob()`. The 4 arithmetic operations, concatenation, trig functions, comparisons, and so forth, should operate on brobs transparently. Concatenation of Brobdingnagian numbers should be done with `cbrob()` instead of `c()`.

The basic low-level function is `brob()`, which takes two vectors: a double for the value and a logical for the sign (defaulting to positive). Given `x`, function `brob(x)` returns  $e^x$ .

Functions `as.glub()` and `glub()` perform analogous operations for the complex plane.

## Author(s)

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

**Examples**

```
googol <- as.brob(1e100)
googolplex <- 10^googol

f <- function(n){exp(n)*n^n*sqrt(2*pi*n)}

f(googol) # close to factorial(googol)
```

---

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	<i>Coerces to numeric or complex form</i>
------------	---

---

**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 *Brobdingnagian numbers*

---

## 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

The package has poor handling of NA and NaN:

```
as.brob(1) + as.brob(c(1,NA))
```

currently 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
```

---

cbrob

*Combine Brobdingnagian vectors*


---

**Description**

Combine Brobdingnagian or Glubdubbribian vectors through concatenation

**Usage**

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

**Arguments**

x	Brobdingnagian vector
...	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)
```

---

 Compare-methods

*Methods for Function Compare in Package Brobdingnag*


---

**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
```

---

 Complex

*Real and imaginary manipulation*


---

**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 whether 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

Function `glub()` takes two arguments that are coerced to Brobdingnagian numbers and returns a complex 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 colloquial R idiom
```

---

length-methods	<i>Get lengths of brobs and glubs</i>
----------------	---------------------------------------

---

**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**

<code>x</code>	Object of class brob (or sometimes glub)
<code>base</code>	In function <code>log()</code> , 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**

`x, ...`                Objects of class brob or, in the case of `sum()` and `prob()`, class glub  
`na.rm`                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

## \*Topic **classes**

- brob-class, 6
- glub-class, 12
- swift-class, 17

## \*Topic **math**

- Arith-methods, 3
- as.numeric, 4
- brob, 5
- cbrob, 7
- Compare-methods, 8
- Complex, 8
- Extract.brob, 9
- getP, 10
- glub, 11
- length-methods, 13
- Logic, 14
- Math, 14
- plot, 15
- Print, 15
- sum, 16

## \*Topic **methods**

- Arith-methods, 3
- Compare-methods, 8
- length-methods, 13

## \*Topic **package**

- Brobdingnag-package, 2
- .cPair, ANY, ANY-method (brob-class), 6
- .cPair, ANY, brob-method (brob-class), 6
- .cPair, ANY, glub-method (glub-class), 12
- .cPair, brob, ANY-method (brob-class), 6
- .cPair, brob, brob-method (brob-class), 6
- .cPair, brob, complex-method (brob-class), 6
- .cPair, brob, glub-method (glub-class), 12
- .cPair, complex, brob-method (brob-class), 6
- .cPair, glub, ANY-method (glub-class), 12
- .cPair, glub, brob-method (glub-class), 12
- .cPair, glub, glub-method (glub-class), 12

- [, brob-method (Extract.brob), 9
- [, glub-method (Extract.brob), 9
- [, brob (Extract.brob), 9
- [, glub (Extract.brob), 9
- [<- , brob-method (Extract.brob), 9
- [<- , glub-method (Extract.brob), 9
- [<- .brob (Extract.brob), 9
- [<- .glub (Extract.brob), 9

- abs (Math), 14
- acos (Math), 14
- acosh (Math), 14
- Arg (Complex), 8
- Arg, brob-method (Complex), 8
- Arg, glub-method (Complex), 8
- Arith, ANY, brob-method (Arith-methods), 3
- Arith, ANY, glub-method (Arith-methods), 3
- Arith, brob, ANY-method (Arith-methods), 3
- Arith, brob, brob-method (Arith-methods), 3
- Arith, brob, complex-method (Arith-methods), 3
- Arith, brob, glub-method (Arith-methods), 3
- Arith, brob, missing-method (Arith-methods), 3
- Arith, complex, brob-method (Arith-methods), 3
- Arith, complex, glub-method (Arith-methods), 3
- Arith, glub, ANY-method (Arith-methods), 3
- Arith, glub, brob-method (Arith-methods), 3
- Arith, glub, complex-method (Arith-methods), 3
- Arith, glub, glub-method (Arith-methods), 3
- Arith, glub, missing-method (Arith-methods), 3
- Arith-methods, 3

- as.brob (brob), 5
- as.complex (as.numeric), 4
- as.complex, brob-method (as.numeric), 4
- as.complex, glub-method (as.numeric), 4
- as.glub (glub), 11
- as.numeric, 4
- as.numeric, brob-method (as.numeric), 4
- as.numeric, glub-method (as.numeric), 4
- asin (Math), 14
- asinh (Math), 14
- atan (Math), 14
- atanh (Math), 14
  
- brob, 5, 10, 11
- brob-class, 6
- Brobdingnag (Brobdingnag-package), 2
- Brobdingnag-package, 2
  
- cBrob (cbrob), 7
- cbrob, 7
- ceiling (Math), 14
- coerce, brob, complex-method (as.numeric), 4
- coerce, brob, numeric-method (as.numeric), 4
- coerce, glub, complex-method (as.numeric), 4
- coerce, glub, numeric-method (as.numeric), 4
- Compare, ANY, brob-method (Compare-methods), 8
- Compare, ANY, glub-method (Compare-methods), 8
- Compare, brob, ANY-method (Compare-methods), 8
- Compare, brob, brob-method (Compare-methods), 8
- Compare, brob, glub-method (Compare-methods), 8
- Compare, glub, ANY-method (Compare-methods), 8
- Compare, glub, brob-method (Compare-methods), 8
- Compare, glub, glub-method (Compare-methods), 8
- Compare-methods, 8
- Complex, 8
- Complex, brob-method (Complex), 8
- Complex, glub-method (Complex), 8
- Complex-methods (Complex), 8
- Conj (Complex), 8
- Conj, brob-method (Complex), 8
- Conj, glub-method (Complex), 8
- cos (Math), 14
- cosh (Math), 14
- cumsum (Math), 14
  
- exp (Math), 14
- Extract.brob, 9
  
- floor (Math), 14
  
- gamma (Math), 14
- getP, 10
- getP, brob-method (brob-class), 6
- getX (getP), 10
- getX, brob-method (brob-class), 6
- glub, 5, 11
- glub-class, 12
  
- Im (Complex), 8
- Im, brob-method (Complex), 8
- Im, glub-method (Complex), 8
- Im<- (Complex), 8
- Im<-, brob-method (Complex), 8
- Im<-, glub-method (Complex), 8
- is.brob (brob), 5
- is.glub (glub), 11
- is.na, 16
  
- length (length-methods), 13
- length, brob-method (length-methods), 13
- length, glub-method (length-methods), 13
- length-methods, 13
- lgamma (Math), 14
- log (Math), 14
- Logic, 14
- Logic, ANY, swift-method (Logic), 14
- Logic, swift, ANY-method (Logic), 14
- Logic, swift, swift-method (Logic), 14
- logic.brob (Logic), 14
  
- Math, 14
- Math, brob-method (Math), 14
- Math, glub-method (Math), 14
- max (sum), 16
- min (sum), 16
- Mod (Complex), 8
- Mod, brob-method (Complex), 8

Mod,glub-method (Complex), 8

plot, 15  
plot,ANY, brob-method (plot), 15  
plot,ANY,glub-method (plot), 15  
plot,brob,ANY-method (plot), 15  
plot,brob,missing-method (plot), 15  
plot,brob-method (plot), 15  
plot,glub,ANY-method (plot), 15  
plot,glub,missing-method (plot), 15  
plot,glub-method (plot), 15  
Print, 15  
print.brob (Print), 15  
print.glub (Print), 15  
prod (sum), 16

range (sum), 16  
Re (Complex), 8  
Re, brob-method (Complex), 8  
Re,glub-method (Complex), 8  
Re<- (Complex), 8  
Re<- ,glub-method (Complex), 8

show, brob-method (Print), 15  
show,glub-method (Print), 15  
sign<- (getP), 10  
sign<- , brob-method (brob-class), 6  
sin (Math), 14  
sinh (Math), 14  
sqrt (Math), 14  
sqrt, brob-method (Math), 14  
sqrt,glub-method (Math), 14  
sum, 16  
Summary, brob-method (sum), 16  
Summary,glub-method (sum), 16  
swift-class, 17

tan (Math), 14  
tanh (Math), 14  
trunc (Math), 14