Very large numbers in R: introducing package Brobdingnag

Robin K. S. Hankin  
Auckland University of Technology

Abstract

This vignette shows how to use the Brobdingnag package to manipulate very large numbers; it is based on Hankin (2007c).

The other vignette shows how to use S4 methods in the context of a simple package.

Keywords: S4 methods, Brobdingnag, R.

1. Introduction

The largest floating point number representable in standard double precision arithmetic is a little under $2^{1024}$, or about $1.79 \times 10^{308}$. This is too small for some applications. The R package Brobdingnag (Swift 1726) overcomes this limit by representing a real number $x$ using a double precision variable with value $\log |x|$, and a logical corresponding to $x \geq 0$; the S4 class of such objects is brob. Complex numbers with large absolute values (class glub) may be represented using a pair of brobs to represent the real and imaginary components.

The package allows user-transparent access to the large numbers allowed by Brobdingnagian arithmetic. The package also includes a vignette—S4_brob—which documents the S4 methods used and includes a step-by-step tutorial. The vignette also functions as a “Hello, World!” example of S4 methods as used in a simple package. It also includes a full description of the glub class.

2. Package Brobdingnag in use

Most readers will be aware of a googol which is equal to $10^{100}$:

```r
> require(Brobdingnag)

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

[1] +exp(230.26)
```

Note the coercion of double value 10 to an object of class brob using function as.brob(): raising this to the power 100 (also double) results in another brob. The result is printed using exponential notation, which is convenient for very large numbers.
A googol is well within the capabilities of standard double precision arithmetic. Now, however, suppose we wish to compute its factorial. Taking the first term of Stirling’s series gives

```r
> stirling <- function(n){n^n*exp(-n)*sqrt(2*pi*n)}

which then yields

```r
> stirling(googol)

[1] 1.000000e+100
```  

Note the transparent coercion to `brob` form within function `stirling()`. It is also possible to represent numbers very close to 1. Thus

```r
> 2^(1/googol)

```  

It is worth noting that if \( x \) has an exact representation in double precision, then \( e^x \) is exactly representable using the system described here. Thus \( e \) and \( e^{1000} \) may be represented exactly.

### 2.1. Accuracy

For small numbers (that is, representable using standard double precision floating point arithmetic), `Brobdingnag` suffers a slight loss of precision compared to normal representation. Consider the following function, whose return value for nonzero arguments is algebraically zero:

```r
f <- function(x){
  as.numeric((pi*x -3*x -(pi-3)*x)/x)
}
```

This function combines multiplication and addition; one might expect a logarithmic system such as described here to have difficulty with it.

```r
> f(1/7)

[1] 1.700029e-16
```  

```r
> f(as.brob(1/7))

[1] -3.143988e-16
```  

This typical example shows that Brobdingnagian numbers suffer a slight loss of precision for numbers of moderate magnitude. This degradation increases with the magnitude of the argument:
Here, the brobs’ accuracy is about two orders of magnitude worse than double precision arithmetic: this would be expected, as the number of bits required to specify the exponent goes as $\log \log x$. Compare

```r
> f(as.brob(1e100))
[1] -3.219444e-14
```

showing a further degradation of precision. However, observe that conventional double precision arithmetic cannot deal with numbers this big, and the package returns about 12 correct significant figures.

### 3. A practical example

In the field of population dynamics, and especially the modelling of biodiversity (Hankin 2007a; Hubbell 2001), complicated combinatorial formulae often arise.

Etienne (2005), for example, considers a sample of $N$ individual organisms taken from some natural population; the sample includes $S$ distinct species, and each individual is assigned a label in the range 1 to $S$. The sample comprises $n_i$ members of species $i$, with $1 \leq i \leq S$ and $\sum n_i = N$. For a given sample $D$ Etienne defines, amongst other terms, $K(D, A)$ for $1 \leq A \leq N - S + 1$ as

$$
K(D, A) = \sum_{\{a_1, \ldots, a_S\} \sum_{i=1}^S a_i = A} \prod_{i=1}^S \frac{s(n_i, a_i)}{s(n_i, 1)} \prod_{i=1}^S \frac{s(n_i, a_i)}{s(n_i, 1)}
$$

where $s(n, a)$ is the Stirling number of the second kind (Abramowitz and Stegun 1965). The summation is over $a_i = 1, \ldots, n_i$ with the restriction that the $a_i$ sum to $A$, as carried out by blockparts() of the partitions package (Hankin 2006, 2007b).

Taking an intermediate-sized dataset due to Saunders\(^1\) of only 5903 individuals—a relatively small dataset in this context—the maximal element of $K(D, A)$ is about $1.435 \times 10^{1165}$. The accuracy of package Brobdingnag in this context may be assessed by comparing it with that computed by PARI/GP (Batut, Belabas, Bernardi, Cohen, and Olivier 2000) with a working precision of 100 decimal places; the natural logs of the two values are $2682.8725605988689$ and $2682.87256059887$ respectively: identical to 14 significant figures.

\(^1\)The dataset comprises species counts on kelp holdfasts; here saunders.exposed.tot of package untb (Hankin 2007a), is used.
4. Conclusions

The **Brobdingnag** package allows representation and manipulation of numbers larger than those covered by standard double precision arithmetic, although accuracy is eroded for very large numbers. This facility is useful in several contexts, including combinatorial computations such as encountered in theoretical modelling of biodiversity.

**Acknowledgments**

I would like to acknowledge the many stimulating and helpful comments made by the R-help list over the years.

**References**


**Affiliation:**

Robin K. S. Hankin
Auckland University of Technology
E-mail: hankin.robin@gmail.com