Tests for floating point arithmetic in **R**

Richard M. Heiberger and John C. Nash

2017-04-24

Abstract

Floating point calculations in any computing language may give results that are very different from results humans expect from arithmetic on real numbers. While in most instances these differences are inconsequential, there are several cases where they could be important:

- results are changed in ways that will affect outcomes or decisions;
- differences or changes in output from new versions of software may point to errors in that software;
- the differences cause upset and wasted effort in either explaining or addressing them.

This article and the accompanying **R** package are intended to provide a structure for building programs that test and report such issues. It is anticipated that this will be open-ended, that is, the development of such test codes will be ongoing and react to innovations in software and hardware.

Motivations

- many queries on R-help and elsewhere (e.g., https://stat.ethz.ch/pipermail/r-help/2017-April/446375.html, which initiated this particular work)
- R Core Team (2017) is the essential subject of the most frequent R-help postings. ??Can we get numbers?? I suspect that is an interesting R exercise. JN
- detect changes that could affect results
  - versions of R
  - versions of compilers (Fortran, C, C++, Java, others?)
  - packages
- provide guidance
  - what is the answer, correct to the working precision
  - what is the best answer available via a prescribed calculation (assuming all steps are carried out to the best limits)
  - how different is computed answer from "correct"?; from "best available"?
- suggest "best practice" if we can

While this article focuses on **R**, we anticipate that some tests will be ported (if they are not already available) to computing languages that **R** may call and which will therefore affect users' results.

Background

- Heiberger and Holland (2015), particularly Appendix G.
- FAQ 7.31 R Core Team (2017) of the **R** documentation
- Gnumeric test xls
- PARANOIA Karpinski (1985)
- Goldberg (1991)

Scope

Because arithmetic affects a large range of computations, it is very easy to allow the scope of tests to become too wide. Here we will categorize the possibilities, but defer the decision of where to draw the line around our tests.

Core arithmetic

Tests of core arithmetic concern the basic arithmetic capabilities of

- fundamental operations of add, subtract, multiply, and divide on the numeric data types in the language.
- input of numeric quantities
- output of numeric quantities
  - for display or printing
  - for reuse in other computations
- those special functions that are "built-in" to some hardware and assumed part of the floating-point arithmetic standards, namely, ?? need to check exactly which
  - sqrt
  - log – NOT part of standard,
  - power, that is a^b, where we may wish to distinguish b integer or not. NOT part of standard

Note that we consider logical tests on numerical quantities separately below, as these raise some issues of programming that need to be addressed.

The numeric data types in **R** that we shall consider are: – numeric (same as double in R) – integer (??R seems to have just 32 bit integers??) – ??any others

Specialized arithmetic

There are a number of specialized tools to make use of features of computing hardware that may be available. These complicate our testing, and may require a large amount of extra effort, so for the moment, this project currently has no tests relating to the computational tools mentioned in this section. However, to the extent that we can do so easily, our tests should be capable of at least recognizing the existence of such features, rendering them amenable to extension. The following are three categories of specialized arithmetic capabilities:

- parallel processing where the order of computation is potentially uncertain
- graphical processing unit (GPUs), which are increasingly used to perform computations e.g., http://www.cs.unc.edu/~ibr/projects/paranoia/
- specialized processing capabilities (database machines, quantum computing, random number devices, others?)

Many **R** users are interested in parallel operations, so the first of these categories is likely the most important to address first. However, the use of GPUs (so-called Graphical Processing Units) is becoming more common. These often have limited precision for storage and arithmetic. Very particular tools in the third category will only be considered in the more distant future, if at all, though others may wish to attempt to port our tests.

Special functions within the language

We have already noted log and power functions above. Trigonometric functions are also part of the **R** language, and are worth testing as errors can have important implications in real life, as they are critical to

computing navigational information. (??do we want refs)

- tests of principal range of inverse trigonometric functions. These should be testable fairly easily within **R**.

xx <- (-40:40)/40 xx asin(xx) # check if in -pi/2, pi/2 acos(xx) # check if in 0, pi atan(xx) # check if in -pi/2, pi/2

- ??similar for hyperbolic functions
- ?? Do we want to consider implications of arithmetic (storage modes) for random number generation. JN suggests no, but keeping a list of potential issues for later consideration.

Special functions in packages

?? What packages generate special functions? Which are worth testing?

Sums and products

- variance algorithms
  - 2 pass
  - 1 pass
  - Kahan
  - other fixes
  - pairwise
- sums and products
  - ordering
  - other strategies
  - good test cases??

Logical tests – `if` statements

- program flow – convergence and termination tests
- sorting
- ?? others

Concerns: - optimizing compilers - setting of tolerances - "Fuzz" or equivalent (get ref to Tektronix e.g. Nash 1978) - other?

R structures for floating-point operations

- "+" "-" "*" "/" basic ops
- "<-"
- "as." for integer and (double / numeric), noting the equivalence of latter two
- dput(), dget()
- print, sprintf, ??

How tests and reports should be presented

Tests against stored `standard` results

Numerical results

Error condition results

Some computations should raise error conditions. Therefore it is sensible for us to include some tests of this nature. The Gnome Office project has a spreadsheet processor **Gnumeric** and one of us (JCN) prepared a test spreadsheet `trig.xls` (https://projects-old.gnome.org/gnumeric/func-tests/trig.xls) that presents many "bad" inputs.

Verifying internal nature of R objects

- str()
- need to show bit pattern / hex pattern ??
- all.equal()
- identical()

Other useful features of R for floating-point testing

Known issues ??

Tests using identities

There are a number of tests we can create that use mathematical identities that may or may not be satisfied in floating point arithmetic.

Square of square root

The test script `isqrt.R` uses the identity

```
$(sqrt(x)) ^ 2 == x$
```

to test this identity for the sequence of integers 0:49. These are established using integer input, and verified using the **R** function `str()` to display the object structure.

How to extend the tests

Discussion and open issues

Index of tests by name

isqrt: squares of square roots of integers

References

Goldberg, David. 1991. "What Every Computer Scientist Should Know About Floating-Point Arithmetic." *ACM Computing Surveys*, 5–48. http://www.validlab.com/goldberg/paper.pdf.

Heiberger, Richard M., and Burt Holland. 2015. *Statistical Analysis and Data Display: An Intermediate Course with Examples in R*. Second. Springer-Verlag, New York. http://www.springer.com/978-1-4939-2121-8.

Karpinski, Richard. 1985. "Paranoia: A Floating-Point Benchmark." *Byte Magazine* 10 (2): 223–35.

R Core Team. 2017. Vienna, Austria: R Foundation for Statistical Computing. https://cran.r-project.org/doc/FAQ/R-FAQ.html#Why-doesn_0027t-R-think-these-numbers-are-equal_003f.