Package 'PreciseSums'

May 7, 2026

Title Accurate Floating Point Sums and Products

Version 0.7

Description Most of the time floating point arithmetic does approximately the right thing. When adding sums or having products of numbers that greatly differ in magnitude, the floating point arithmetic may be incorrect. This package implements the Kahan (1965) sum <doi:10.1145/363707.363723>, Neumaier (1974) sum <doi:10.1002/zamm.19740540106>, pairwise-sum (adapted from 'NumPy', See Castaldo (2008) <doi:10.1137/070679946> for a discussion of accuracy), and arbitrary precision sum (adapted from the fsum in 'Python' ; Shewchuk (1997) <https://people.eecs.berkeley.edu/~jrs/papers/robustr.pdf>). In addition, products are changed to long double precision for accuracy, or changed into a log-sum for accuracy.

Depends R (>= 3.2)

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.3.2

Suggests testthat

BugReports https://github.com/nlmixr2/PreciseSums/issues/

URL https://github.com/nlmixr2/PreciseSums

NeedsCompilation yes

Author Matthew Fidler [aut, cre, cph],
Raymond Hettinger [cph, aut],
Jonathan Shewchuk [cph, aut],
Julian Taylor [cph, aut],
Nathaniel Smith [cph, aut],
NumPy Team [cph],
Python Team [cph]

Maintainer Matthew Fidler <matthew.fidler@gmail.com>

Repository CRAN

Date/Publication 2024-09-17 22:20:12 UTC

Contents

`fsum` *Return an accurate floating point sum of values*

Description

This method avoids loss of precision by tracking multiple intermediate partial sums. Based on python's math.fsum

Usage

```
fsum(numbers)
```

Arguments

`numbers`	A vector of numbers to sum.

Value

Sum of numbers without loss of precision

The algorithm's accuracy depends on IEEE-754 arithmetic guarantees and the typical case where the rounding mode is half-even. On some non-Windows builds, the underlying C library uses extended precision addition and may occasionally double-round an intermediate sum causing it to be off in its least significant bit.

Author(s)

Matthew Fidler (R implementation), Raymond Hettinger, Jonathan Shewchuk, Python Team

References

https://docs.python.org/2/library/math.html https://github.com/python/cpython/blob/a0ce375e10b50f7606cb86b072fed7d8cd574fe7/Modules/mathmodule.c

Shewchuk, JR. (1996) *Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates.* http://www-2.cs.cmu.edu/afs/cs/project/quake/public/papers/robust-arithmetic.ps

Examples

```
sum(c(1,1e100,1,-1e100)) ## Should be 2, gives 0
fsum(c(1,1e100,1,-1e100)) ## Gives 2.
```

---

kahanSum *Using the Kahan method, take a more accurate sum*

Description

Using the Kahan method, take a more accurate sum

Usage

```
kahanSum(numbers)
```

Arguments

numbers	A vector of numbers to sum.

Value

Sum of numbers

References

https://en.wikipedia.org/wiki/Kahan_summation_algorithm

Examples

```
sum(c(1,1e100,1,-1e100)) ## Should be 2, gives 0
kahanSum(c(1,1e100,1,-1e100)) ## Not accurate enough for the correct result. (still = 0)
```

---

neumaierSum *Using the Neumaier method, take a more accurate sum*

Description

Using the Neumaier method, take a more accurate sum

Usage

```
neumaierSum(numbers)
```

Arguments

numbers	A vector of numbers to sum.

Value

Sum of numbers, a bit more accurate than kahanSum

References

https://en.wikipedia.org/wiki/Kahan_summation_algorithm

Examples

```
sum(c(1,1e100,1,-1e100)) ## Should be 2, gives 0
neumaierSum(c(1,1e100,1,-1e100)) ## Gives 2
```

`pairwiseSum` *Return an accurate floating point sum of values*

Description

This was taken by NumPy and adapted for use here. It is more accurate than a standard sum, but still has numerical issues. Its main benefit is that it is about the same amount of time as a standard time with the added accuracy.

Usage

```
pairwiseSum(numbers)
```

Arguments

`numbers`	A vector of numbers to sum.

Value

A sum of numbers with a rounding error of O(lg n) instead of O(n).

Author(s)

Matthew Fidler (R implementation), Julian Taylor, Nathaniel J Smith, and NumPy team. #' @examples sum(c(1,1e100,1,-1e100)) ## Should be 2, gives 0 pairwiseSum(c(1,1e100,1,-1e100)) ## Should be 2, still 0

References

https://github.com/juliantaylor/numpy/blob/b0bc01275cac04483e6df021211c1fa2ba65eaa3/numpy/core/src/umath/loops.c.src

https://github.com/numpy/numpy/pull/3685

psProd — *Using PreciceSums's default method, take a product*

Description

Using PreciceSums's default method, take a product

Usage

```
psProd(numbers)
```

Arguments

numbers	A vector of numbers to sum.

Value

Product of numbers

psSetProd — *Choose the type of product to use in PreciceSums. These are used in the PreciceSums* prod *blocks*

Description

Choose the type of product to use in PreciceSums. These are used in the PreciceSums prod blocks

Usage

```
psSetProd(type = c("long double", "double", "logify"))
```

Arguments

type	Product to use for prod() in PreciceSums blocks long double converts to long double, performs the multiplication and then converts back. double uses the standard double scale for multiplication.

Value

nothing

Author(s)

Matthew L. Fidler

psSetSum *Choose the type of sums to use for PreciceSums.*

Description

Choose the types of sums to use in PreciceSums. These are used in the PreciceSums `sum` blocks and the `psSum` function

Usage

```
psSetSum(type = c("pairwise", "fsum", "kahan", "neumaier", "klein", "c"))
```

Arguments

`type`	Sum type to use for `psSum` and `sum()` in PreciceSums code blocks. `pairwise` uses the pairwise sum (fast, default) `fsum` uses Python's fsum function (most accurate) `kahan` uses kahan correction `neumaier` uses Neumaier correction `klein` uses Klien correction `c` uses no correction, bud default/native summing

Value

nothing

Author(s)

Matthew L. Fidler

psSum *Using PreciceSums's default method, take a sum*

Description

Using PreciceSums's default method, take a sum

Usage

```
psSum(numbers)
```

Arguments

`numbers`	A vector of numbers to sum.

Value

Sum of numbers

Index