Package: zuexact (via r-universe)

October 22, 2026

Title Exactly Rounded Sums, Means and Dot Products

Version 0.1.0

Description Sums of double-precision numbers computed exactly and rounded once, so the result is the correctly rounded true sum and does not depend on the order of the elements, the platform, or whether R was built with long doubles. Built on the small and large superaccumulators of Neal (2015) <doi:10.48550/arXiv.1505.05571>. Provides exactly rounded sums, means, cumulative sums, dot products and norms, row, column and grouped sums, accumulators that can be built in pieces and merged across processes, and a registered 'C' interface for other packages.

License MIT + file LICENSE

URL https://pedrobtz.github.io/zuexact/, https://github.com/pedrobtz/zuexact

BugReports https://github.com/pedrobtz/zuexact/issues

Depends R (>= 4.1)

Suggests gmp, knitr, parallel, PreciseSums, rmarkdown, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

Config/roxygen2/version 8.1.0

SystemRequirements C99

NeedsCompilation yes

Author Pedro Baltazar [aut, cre, cph], Radford M. Neal [cph] (Author of the vendored 'xsum' library)

Maintainer Pedro Baltazar <pedrobtz@gmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-10-22 15:25:17 UTC

RemoteUrl https://github.com/cran/zuexact

RemoteRef HEAD

RemoteSha 548e935164aa456685fc4ac24778e6acf994379f

Contents

xsum 2
xsum_acc 3
xsum_cumsum 5
xsum_dot 6
xsum_mean 7
xsum_rowsums 8
xsum_weighted_mean 9
zuexact_info 10

Index **11**

`xsum` *Exactly rounded sum*

Description

`xsum()` adds its arguments exactly and rounds once, to the nearest double (ties to even). The result is the correctly rounded value of the true sum, so it does not depend on the order of the elements, on the platform, or on whether R was built with long doubles. Base `sum()` accumulates in `long double`, which is 80-bit on x86-64 Linux and Windows and 64-bit on Apple Silicon, so its result varies between machines.

Usage

```
xsum(..., na.rm = FALSE)
```

Arguments

`...`	Numeric, integer, logical or complex vectors, or `NULL`.
`na.rm`	Remove `NA` and `NaN` values before summing?

Details

Integers and logicals are converted to double exactly; the result is always a double, or a complex if any argument is complex. Complex values are summed as two exact sums, of the real and of the imaginary parts.

Value

A double of length one, or a complex if any argument is complex.

Special values

With `na.rm = FALSE`, the rules are the same on every platform:

Input contains	Result
nothing	`0`
any `NA`	`NA`
no `NA`, any other `NaN`	`NaN`
both `Inf` and `-Inf`	`NaN`
`Inf` (and finite values)	`Inf`
`-Inf` (and finite values)	`-Inf`
finite values whose exact sum is out of range	`Inf` or `-Inf`
only zeros, of any sign	`0` (never `-0`)

`na.rm = TRUE` removes `NA` and `NaN`, as `sum()` does; infinities stay. For complex input, each part follows the table, and an element is removed when either of its parts is `NA` or `NaN`.

See Also

`zuexact_info()` for the build's accumulator threshold.

Examples

```
# Base sum() gives 1 on x86-64 and 0 on Apple Silicon; xsum() gives 1.
xsum(c(1e16, 1, -1e16))

# Neither platform's sum() gets this one right.
sum(c(1e20, 1, -1e20))
xsum(c(1e20, 1, -1e20))

# The order of the terms never matters.
x <- c(1, 1e100, 1, -1e100)
xsum(x) == xsum(rev(x))

# Several arguments are summed together.
xsum(1:10, c(0.5, 0.25), TRUE)
```

---

xsum_acc *Accumulators: exact sums built in pieces*

Description

An accumulator holds an exact sum in progress. Add vectors to it in any order and in any number of pieces, merge accumulators built elsewhere, and read the correctly rounded value at any time. The result is the same as `xsum()` of all the pieces together, bit for bit.

Usage

```
xsum_acc(x = NULL, na.rm = FALSE)

xsum_add(acc, x, na.rm = FALSE)

xsum_merge(acc, other)

xsum_negate(acc)

xsum_value(acc)

xsum_mean_value(acc, n)
```

Arguments

x	A numeric or logical vector (NULL for none).
na.rm	Remove NA and NaN values before summing?
acc, other	Accumulators, from xsum_acc().
n	The number of terms the accumulator holds, for the mean.

Details

Accumulators are ordinary R values (raw vectors of class "xsum_acc", 568 bytes): xsum_add(), xsum_merge() and xsum_negate() return a new accumulator and leave their argument unchanged. Because they are values, they serialise: an accumulator built on a worker by parallel, mirai or future comes back to the caller and merges there. Reduce(xsum_merge, accs) combines a list of them in any order.

xsum_negate() changes the sign of the sum, so xsum_merge(a, xsum_negate(b)) is the exact difference of two sums, rounded once.

An accumulator does not count its terms; xsum_mean_value() takes the count. Special values follow the table in xsum(); an accumulator remembers whether it has seen NA, NaN, Inf or -Inf.

Accumulator bytes are those of the machine that made them. Every CRAN platform is little-endian; an accumulator from a big-endian machine signals a zuexact_accumulator_error.

Value

xsum_acc(), xsum_add(), xsum_merge() and xsum_negate() return an accumulator. xsum_value() and xsum_mean_value() return a double of length one.

Examples

```
acc <- xsum_acc()
acc <- xsum_add(acc, c(1e20, 1))
acc <- xsum_add(acc, -1e20)
xsum_value(acc)

# Pieces in any order, merged in any order.
chunks <- split(c(1e100, 1:10, -1e100), rep(1:3, length.out = 12))
```

```
accs <- lapply(chunks, xsum_acc)
xsum_value(Reduce(xsum_merge, rev(accs)))

# The exact difference of two sums.
a <- xsum_acc(c(0.1, 0.2))
b <- xsum_acc(0.3)
xsum_value(xsum_merge(a, xsum_negate(b)))
```

`xsum_cumsum` *Exactly rounded cumulative sums*

Description

Each element of `xsum_cumsum(x)` is the exactly rounded sum of `x[1:i]`. Unlike base `cumsum()`, no rounding error carries from one element to the next. Special values follow the table in `xsum()`, applied to each prefix: once an `NA` appears, every later element is `NA`.

Usage

```
xsum_cumsum(x)
```

Arguments

`x`	A numeric or logical vector.

Details

This is the slowest function in the package: every prefix is rounded, which costs a scan of the accumulator per element.

Value

A double vector of the same length as `x`.

Examples

```
x <- c(1e20, 1, -1e20, 1)
cumsum(x)
xsum_cumsum(x)
```

`xsum_dot` *Exactly rounded dot products and norms*

Description

`xsum_dot()` returns the correctly rounded value of the exact dot product `sum(x * y)`, products included: each product is split into its rounded value and its exact rounding error with `fma()`, and both are added exactly. One rounding.

Usage

```
xsum_dot(x, y)

xsum_sqnorm(x)

xsum_norm(x)
```

Arguments

`x, y`	Numeric or logical vectors of the same length.

Details

`xsum_sqnorm()` is `xsum_dot(x, x)`, the exactly rounded squared Euclidean norm. `xsum_norm()` is its square root, so it has **two** roundings: the squared norm and the square root.

Value

A double of length one.

Limits

The product error is exact only while each product is above about 2^-970 in magnitude (about 1e-292). Below that, the low bits of the error are lost to underflow, so a dot product made only of such tiny products can be off in its last bits. A product whose magnitude exceeds the largest double is `Inf` even if the exact dot product would be finite.

Special values

Any `NA` in `x` or `y` gives `NA`. Otherwise a `NaN` product (a `NaN` input, or `0 * Inf`) gives `NaN`; infinite products of both signs give `NaN`; infinite products of one sign give that infinity.

Examples

```
# (1e8 + 1) * (1e8 - 1) is 1e16 - 1, which rounds to 1e16 as a double.
x <- c(1e8 + 1, 1)
y <- c(1e8 - 1, -1e16)
sum(x * y)
```

```
xsum_dot(x, y)

xsum_norm(c(3, 4))
```

---

xsum_mean *Exactly rounded mean*

Description

xsum_mean() divides the exact sum by the number of terms and rounds once: the result is the correctly rounded value of the true mean, not xsum(x) / length(x), which would round twice. Base mean() makes a second pass to correct its first estimate, but its accumulator is still platform-dependent.

Usage

```
xsum_mean(x, na.rm = FALSE)
```

Arguments

x	A numeric, integer, logical or complex vector.
na.rm	Remove NA and NaN values before summing?

Details

Special values follow the table in xsum(). The mean of no values is NaN, as in base R. xsum divides by a 32-bit count, so a mean over more than 2^32 - 1 terms signals a zuexact_count_error.

Value

A double of length one (complex for complex x).

See Also

Other means: xsum_weighted_mean()

Examples

```
# Base mean() is accurate to three digits here (Neal, 2015).
x <- c(1e15, -1e15, 0.1)
mean(x)
xsum_mean(x)
```

xsum_rowsums *Exactly rounded row, column and grouped sums*

Description

xsum_rowsums() and xsum_colsums() are exact versions of rowSums() and colSums(): every row or column sum is the correctly rounded exact sum. xsum_by() is an exact version of rowsum(): the rows of x are summed within each group, and the result has one row per group, sorted.

Usage

```
xsum_rowsums(x, na.rm = FALSE)

xsum_colsums(x, na.rm = FALSE)

xsum_by(x, group, na.rm = FALSE)
```

Arguments

x	A numeric or logical matrix, or a data frame of numeric or logical columns. For xsum_by(), also a vector.
na.rm	Remove NA and NaN values before summing?
group	A vector with one value per row of x (per element, for a vector). NA is a group of its own, sorted last, as in rowsum().

Details

Special values follow the table in xsum(), cell by cell. Row sums are computed a block of rows at a time, so the extra memory they need does not grow with the number of rows.

Value

xsum_rowsums() and xsum_colsums() return a double vector named by the row or column names. xsum_by() returns a double matrix with one row per group (named by the group) and one column per column of x.

Examples

```
m <- cbind(a = c(1e20, 1), b = c(1, 1e20), c = c(-1e20, -1e20))
rowSums(m)
xsum_rowsums(m)
xsum_colsums(m)

xsum_by(c(1e20, 1, 2, -1e20), c("x", "y", "y", "x"))
```

xsum_weighted_mean *Weighted mean from exact sums*

Description

`xsum_weighted_mean()` computes `sum(w * x)` and `sum(w)` exactly, each rounded once, and divides. The result therefore has **three** roundings: the two sums and the division. Each sum is exact, which removes the cancellation and accumulation error that dominates `sum(w * x) / sum(w)`; an exact ratio of two superaccumulators is not available.

Usage

```
xsum_weighted_mean(x, w, na.rm = FALSE)
```

Arguments

`x`	A numeric or logical vector.
`w`	Weights, of the same length as `x`.
`na.rm`	Remove `NA` and `NaN` values before summing?

Details

With `na.rm = TRUE`, pairs where `x` is `NA` or `NaN` are removed, as in `stats::weighted.mean()`; `NA` weights are not removed.

Value

A double of length one.

See Also

Other means: `xsum_mean()`

Examples

```
xsum_weighted_mean(c(1e16, 1, -1e16), c(1, 1, 1))
```

`zuexact_info` *Information about this build of zuexact*

Description

Reports the vendored 'xsum' library and how it was built.

Usage

```
zuexact_info()
```

Value

A list with elements:

- `xsum`: the upstream stable version vendored.
- `xsum_commit`: its commit in https://github.com/radfordneal/xsum.
- `large_threshold`: vectors with at least this many elements are summed with xsum's large accumulator, shorter ones with the small one. The choice affects speed only, never the result.
- `simd`: the SIMD instructions xsum was compiled to use: `"avx"`, `"sse2"` or `"none"`.
- `fma_hardware`: whether the compiler reported a hardware fused multiply-add. Dot products are exact either way: with it, each product's error comes from `fma()`; without it, from Dekker's splitting algorithm, because some C libraries' software `fma()` is not exact.
- `long_double`: whether this R has long doubles, which is what makes base `sum()` differ between platforms.

Examples

```
zuexact_info()
```

Index

∗ **means**
    xsum_mean, 7
    xsum_weighted_mean, 9

colSums(), *8*
cumsum(), *5*

mean(), *7*

rowsum(), *8*
rowSums(), *8*

stats::weighted.mean(), *9*
sum(), *2, 3*, *10*

xsum, 2
xsum(), *3–5*, *7, 8*
xsum_acc, 3
xsum_add *(*xsum_acc*)*, 3
xsum_by *(*xsum_rowsums*)*, 8
xsum_colsums *(*xsum_rowsums*)*, 8
xsum_cumsum, 5
xsum_dot, 6
xsum_mean, 7
xsum_mean(), *9*
xsum_mean_value *(*xsum_acc*)*, 3
xsum_merge *(*xsum_acc*)*, 3
xsum_negate *(*xsum_acc*)*, 3
xsum_norm *(*xsum_dot*)*, 6
xsum_rowsums, 8
xsum_sqnorm *(*xsum_dot*)*, 6
xsum_value *(*xsum_acc*)*, 3
xsum_weighted_mean, 9
xsum_weighted_mean(), *7*

zuexact_info, 10
zuexact_info(), *3*