Package 'ScaledMatrix'

April 25, 2025

Version 1.17.0

Date 2024-02-29

Title Creating a DelayedMatrix of Scaled and Centered Values

Imports methods, Matrix, S4Vectors, DelayedArray

Suggests testthat, BiocStyle, knitr, rmarkdown, BiocSingular, DelayedMatrixStats

biocViews Software, DataRepresentation

Description Provides delayed computation of a matrix of scaled and centered values. The result is equivalent to using the scale() function but avoids explicit realization of a dense matrix during block processing. This permits greater efficiency in common operations, most notably matrix multiplication.

License GPL-3

VignetteBuilder knitr

RoxygenNote 7.3.1

BugReports https://github.com/LTLA/ScaledMatrix/issues

URL https://github.com/LTLA/ScaledMatrix

git_url https://git.bioconductor.org/packages/ScaledMatrix

git_branch devel

git_last_commit f2e2676

git_last_commit_date 2025-04-15

Repository Bioconductor 3.22

Date/Publication 2025-04-24

Author Aaron Lun [aut, cre, cph]

Maintainer Aaron Lun <infinite.monkeys.with.keyboards@gmail.com>

Contents

ScaledMatrix *The ScaledMatrix class*

Description

Defines the ScaledMatrixSeed and ScaledMatrix classes and their associated methods. These classes support delayed centering and scaling of the columns in the same manner as `scale`, but preserving the original data structure for more efficient operations like matrix multiplication.

Usage

```
ScaledMatrix(x, center = NULL, scale = NULL)
```

Arguments

`x`	A matrix or any matrix-like object (e.g., from the **Matrix** package). This can alternatively be a ScaledMatrixSeed, in which case any values of `center` and `scale` are ignored.
`center`	A numeric vector of length equal to `ncol(x)`, where each element is to be subtracted from the corresponding column of `x`. A `NULL` value indicates that no subtraction is to be performed. Alternatively `TRUE`, in which case it is set to the column means of `x`.
`scale`	A numeric vector of length equal to `ncol(x)`, where each element is to divided from the corresponding column of `x` (after subtraction). A `NULL` value indicates that no division is to be performed. Alternatively `TRUE`, in which case it is set to the column-wise root-mean-squared differences from `center` (interpretable as standard deviations if `center` is set to the column means, see `scale` for commentary).

Value

The `ScaledMatrixSeed` constructor will return a ScaledMatrixSeed object.

The `ScaledMatrix` constructor will return a ScaledMatrix object equivalent to `t((t(x) - center)/scale)`.

Methods for ScaledMatrixSeed objects

ScaledMatrixSeed objects are implemented as DelayedMatrix backends. They support standard operations like `dim`, `dimnames` and `extract_array`.

Passing a ScaledMatrixSeed object to the `DelayedArray` constructor will create a ScaledMatrix object.

It is possible for `x` to contain a ScaledMatrix, thus nesting one ScaledMatrix inside another. This can occasionally be useful in combination with transposition to achieve centering/scaling in both dimensions.

Methods for ScaledMatrix objects

ScaledMatrix objects are derived from DelayedMatrix objects and support all of valid operations on the latter. Several functions are specialized for greater efficiency when operating on ScaledMatrix instances, including:

- Subsetting, transposition and replacement of row/column names. These will return a new ScaledMatrix rather than a DelayedMatrix.
- Matrix multiplication via `%*%`, `crossprod` and `tcrossprod`. These functions will return a DelayedMatrix.
- Calculation of row and column sums and means by `colSums`, `rowSums`, etc.

All other operations applied to a ScaledMatrix will use the underlying **DelayedArray** machinery. Unary or binary operations will generally create a new DelayedMatrix instance containing a ScaledMatrixSeed.

Tranposition can effectively be used to allow centering/scaling on the rows if the input `x` is transposed.

Efficiency vs precision

The raison d'etre of the ScaledMatrix is that it can offer faster matrix multiplication by avoiding the **DelayedArray** block processing. This is done by refactoring the scaling/centering operations to use the (hopefully more efficient) multiplication operator of the original matrix `x`. Unfortunately, the speed-up comes at the cost of increasing the risk of catastrophic cancellation. The procedure requires subtraction of one large intermediate number from another to obtain the values of the final matrix product. This could result in a loss of numerical precision that compromises the accuracy of downstream algorithms. In practice, this does not seem to be a major concern though one should be careful if the input `x` contains very large positive/negative values.

Author(s)

Aaron Lun

Examples

```
library(Matrix)
y <- ScaledMatrix(rsparsematrix(10, 20, 0.1),
    center=rnorm(20), scale=1+runif(20))
y

crossprod(y)
tcrossprod(y)
y %*% rnorm(20)
```

Index