

Package ‘bigsparser’

August 24, 2020

Title Sparse Matrix Format with Data on Disk

Version 0.4.0

Description Provides a sparse matrix format with data stored on disk, to be used in both R and C++. This is intended for more efficient use of sparse data in C++ and also when parallelizing, since data on disk does not need copying. Only a limited number of features will be implemented. For now, conversion can be performed from a 'dgCMatrix' or a 'dsCMatrix' from R package 'Matrix'.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

URL <https://github.com/privefl/bigsparser>

BugReports <https://github.com/privefl/bigsparser/issues>

Depends R (>= 3.1)

LinkingTo Rcpp, RcppEigen, rmio (>= 0.2)

Imports Rcpp, bigassertr, methods

Suggests Matrix, testthat (>= 2.1.0)

NeedsCompilation yes

Author Florian Privé [aut, cre]

Maintainer Florian Privé <florian.prive.21@gmail.com>

Repository CRAN

Date/Publication 2020-08-24 08:00:07 UTC

R topics documented:

dim,SFBM-method	2
SFBM-class	2
sp_prodVec	3
sp_solve_sym	4

Index**6**

dim,SFBM-method	<i>Dimension and type methods for class SFBM.</i>
-----------------	---

Description

Dimension and type methods for class SFBM.

Usage

```
## S4 method for signature 'SFBM'
dim(x)
```

```
## S4 method for signature 'SFBM'
length(x)
```

Arguments

x An object of class [SFBM](#).

SFBM-class	<i>Class SFBM</i>
------------	-------------------

Description

A reference class for storing and accessing sparse matrix-like data stored in files on disk.

Convert a dgCMatrix or dsCMatrix to an SFBM.

Usage

```
as_SFBM(spmat, backingfile = tempfile())
```

Arguments

spmat A dgCMatrix (non-symmetric sparse matrix of type 'double') or dsCMatrix (symmetric sparse matrix of type 'double').

backingfile Path to file where to store data. Extension .sbk is automatically added.

Details

An object of class SFBM has many fields:

- \$address: address of the external pointer containing the underlying C++ object to be used as a XPtr<SFBM> in C++ code
- \$extptr: (internal) use \$address instead
- \$nrow: number of rows
- \$ncol: number of columns
- \$nval: number of non-zero values
- \$p: vector of column positions
- \$backingfile or \$sbk: File with extension 'sbk' that stores the data of the SFBM
- \$rds: 'rds' file (that may not exist) corresponding to the 'sbk' file
- \$is_saved: whether this object is stored in \$rds?

And some methods:

- \$save(): Save the SFBM object in \$rds. Returns the SFBM.
- \$add_columns(): Add new columns from another sparse dgCMatrix.

Value

The new [SFBM](#).

Examples

```
spmat <- Matrix::rsparsematrix(1000, 1000, 0.01)
class(spmat)
(X <- as_SFBM(spmat))
```

sp_prodVec

Products with a vector

Description

Products between an [SFBM](#) and a vector.

Usage

```
sp_prodVec(X, y)
```

```
sp_cprodVec(X, y)
```

Arguments

X	An SFBM .
y	A vector of same size of the number of columns of X for sp_prodVec() and as the number of rows of X for sp_cprodVec().

Value

- `sp_prodVec()`: the vector which is equivalent to $X \%*\% y$ if X was a `dgCMatrix`.
- `sp_cprodVec()`: the vector which is equivalent to `Matrix::crossprod(X,y)` if X was a `dgCMatrix`.

Examples

```

spmat <- Matrix::rsparsematrix(1000, 1000, 0.01)
X <- as_SFBM(spmat)
sp_prodVec(X, rep(1, 1000))
sp_cprodVec(X, rep(1, 1000))

```

 sp_solve_sym

Solver for symmetric SFBM

Description

Solve $Ax=b$ where A is a symmetric SFBM, and b is a vector.

Usage

```

sp_solve_sym(
  A,
  b,
  add_to_diag = rep(0, ncol(A)),
  tol = 1e-10,
  maxiter = 10 * ncol(A)
)

```

Arguments

<code>A</code>	A symmetric SFBM .
<code>b</code>	A vector.
<code>add_to_diag</code>	Vector (or single value) to <i>virtually</i> add to the diagonal of A . Default is 0s.
<code>tol</code>	Tolerance for convergence. Default is $1e-10$.
<code>maxiter</code>	Maximum number of iterations for convergence.

Value

The vector x , solution of $Ax=b$.

Examples

```
N <- 100
spmat <- Matrix::rsparsematrix(N, N, 0.01, symmetric = TRUE)
X <- bigsparser::as_SFBM(as(spmat, "dgCMatrix"))
b <- runif(N)

test <- tryCatch(as.vector(Matrix::solve(spmat, b)), error = function(e) print(e))
test2 <- tryCatch(sp_solve_sym(X, b), error = function(e) print(e))

test3 <- as.vector(Matrix::solve(spmat + Matrix::Diagonal(N, 1:N), b))
test4 <- sp_solve_sym(X, b, add_to_diag = 1:N)
all.equal(test3, test4)
```

Index

`as_SFBM` (SFBM-class), [2](#)

`dim`, SFBM-method, [2](#)

`length`, SFBM-method (`dim`, SFBM-method), [2](#)

SFBM, [2–4](#)

SFBM-class, [2](#)

SFBM_RC (SFBM-class), [2](#)

`sp_cprodVec` (`sp_prodVec`), [3](#)

`sp_prodVec`, [3](#)

`sp_solve_sym`, [4](#)