

# Package ‘rmumps’

October 20, 2022

**Type** Package

**Title** Wrapper for MUMPS Library

**Version** 5.2.1-20

**Date** 2022-10-19

**Maintainer** Serguei Sokol <sokol@insa-toulouse.fr>

**Description** Some basic features of 'MUMPS' (Multifrontal Massively Parallel sparse direct Solver) are wrapped in a class whose methods can be used for sequentially solving a sparse linear system (symmetric or not) with one or many right hand sides (dense or sparse). There is a possibility to do separately symbolic analysis, LU (or LDL<sup>t</sup>) factorization and system solving. Third part ordering libraries are included and can be used: 'PORD', 'METIS', 'SCOTCH'. 'MUMPS' method was first described in Amestoy et al. (2001) <[doi:10.1137/S0895479899358194](https://doi.org/10.1137/S0895479899358194)> and Amestoy et al. (2006) <[doi:10.1016/j.parco.2005.07.004](https://doi.org/10.1016/j.parco.2005.07.004)>.

**License** GPL (>= 2)

**Depends** methods

**Imports** Rcpp (>= 0.12.0)

**LinkingTo** Rcpp

**SystemRequirements** C++11, GNU Make

**NeedsCompilation** yes

**Biarch** yes

**Suggests** testthat, Matrix, slam

**BugReports** <https://github.com/sgsokol/rmumps/issues>

**URL** <http://www.mumps-solver.org/>, <https://github.com/sgsokol/rmumps/>

**RoxygenNote** 7.0.2

**Encoding** UTF-8

**Author** Serguei Sokol [aut, cre],  
Emmanuel Agullo [ctb],  
Patrick Amestoy [ctb, cph],

Maurice Bremond [ctb],  
Alfredo Buttari [ctb],  
Philippe Combes [ctb],  
Marie Durand [ctb],  
Aurelia Fevre [ctb],  
Abdou Guermouche [ctb],  
Guillaume Joslin [ctb],  
Jacko Koster [ctb],  
Jean-Yves L'Excellent [ctb],  
Stephane Pralet [ctb],  
Chiara Puglisi [ctb],  
Francois-Henry Rouet [ctb],  
Wissam Sid-Lakhdar [ctb],  
Tzvetomila Slavova [ctb],  
Bora Ucar [ctb],  
Clement Weisbecker [ctb],  
Juergen Schulze [ctb],  
George Karypis [ctb],  
Douglas C. Schmidt [ctb],  
Isamu Hasegawa [ctb],  
Alexander Chemeris [ctb],  
Makoto Matsumoto [ctb],  
Takuji Nishimura [ctb],  
Francois Pellegrini [ctb],  
David Goudin [ctb],  
Pascal Henon [ctb],  
Pierre Ramet [ctb],  
Sebastien Fourestier [ctb],  
Jun-Ho Her [ctb],  
Cedric Chevalier [ctb],  
Timothy A. Davis [ctb, cph],  
Iain S. Duff [ctb, cph],  
John K. Reid [ctb, cph],  
Richard Stallman [ctb],  
Samuel Thibault [ctb, cph],  
CERFACS [cph],  
CNRS [cph],  
ENS Lyon [cph],  
INP Toulouse [cph],  
INRIA [cph],  
University of Bordeaux [cph],  
Regents of the University of Minnesota [cph],  
Free Software Foundation, Inc [cph],  
Alexander Chemeris [cph],  
Makoto Matsumoto [cph],  
Takuji Nishimura [cph],  
Universite de Bordeaux [cph],  
CNRS [cph],

INRAE [cph]

**Repository** CRAN**Date/Publication** 2022-10-20 09:32:37 UTC**R topics documented:**

rmumps-package . . . . .	3
Rcpp_Rmumps-class . . . . .	4
RMUMPS_PERM . . . . .	6
Rmumps__del_ptr . . . . .	7
Rmumps__get_permutation . . . . .	8
Rmumps__ptr_ijv . . . . .	8
Rmumps__set_mat_ptr . . . . .	9
Rmumps__set_permutation . . . . .	9
Rmumps__solveptr . . . . .	10
Rmumps__triplet . . . . .	10
<b>Index</b>	<b>11</b>

---

rmumps-package	<i>Rcpp port of MUMPS library for LU or LDL<sup>t</sup> factorization of sparse matrices</i>
----------------	--

---

**Description**

Creates a MUMPS compatible object storing a sparse matrix. Gives a possibility to do separately symbolic analysis, factorization and system solving.

**Details**

Create a new Rmumps object with `A <- Rmumps$new(asparsed)` then solve a linear system with one or many right hand sides `x <- solve(A, b)`

**Author(s)**

Serguei Sokol, INRA

Maintainer: Serguei Sokol (sokol at insa-toulouse.fr)

**References**MUMPS official site <http://mumps.enseiht.fr>Sokol S (2022). `_Rmumps: Rcpp port of MUMPS_`. rmumps package version 5.2.1-19, <URL: <http://CRAN.R-project.org/package=rmumps>>.

**Examples**

```
## Not run:
A <- Rmumps$new(asparse)
x <- solve(A, b)

## End(Not run)
```

---

Rcpp\_Rmumps-class      *Rcpp Exported Class Wrapping MUMPS library*

---

**Description**

This class can be used for storing sparse matrix and solving corresponding linear system with one or many right hand sides. There is a possibility to do separately symbolic analysis, LU factorization and system solving.

**Fields**

**sym:** integer (read only), 0=non symmetric matrix, 1=symmetric with pivots on diagonal or 2=general symmetric

**copy:** logical, copy or not rhs and matrix values

**mrhs:** numeric matrix, multiple rhs (always overwritten with solution)

**rhs:** numeric vector, single rhs (always overwritten with solution)

**Methods**

**new(asp, sym=0, copy=TRUE):** constructor from Matrix::dgTMatrix class (or from convertible to it) and slam::simple\_triplet\_matrix class

**new(i, j, x, n, copy=TRUE):** constructor from triade rows, cols, vals

**symbolic():** do symbolic analysis (stored internally)

**numeric():** do LU or LDL<sup>t</sup> factorization (stored internally)

**solve(b):** solve single rhs (if b is a vector) or multiple rhs if b is a matrix (can be dense or sparse).  
Return the solution(s).

**solvet(b):** same as solve() but solves with transposed matrix

**det():** Return determinant of the matrix

**inv():** Return inverse of the matrix)

**set\_mat\_data(x):** updates matrix entries (x must be in the same order as in previous calls)

**set\_icntl(iv, ii):** set ICNTL parameter vector

**get\_icntl():** get ICNTL parameter vector

**set\_cntl(v, iv):** set CNTL parameter vector

**get\_cntl():** get CNTL parameter vector

**get\_infos():** get a named list of information vectors: info, rinfo, infog and rinfog

`dim()`: Return a dimension vector of the matrix  
`nrow()`: Return a row number of the matrix  
`ncol()`: Return a column number of the matrix  
`print()`: Print summary information on the matrix  
`show()`: Print summary information on the matrix  
`set_keep()`: Set KEEP array elements (undocumented feature of MUMPS)  
`get_keep()`: Get a copy of KEEP array elements (length=500)  
`set_permutation(perm)`: Set permutation type which can impact storage and factorization performances. Parameter perm can take one of the following predefined integer values RMUMPS\_PERM\_AMD, RMUMPS\_PERM\_AMF, RMUMPS\_PERM\_SCOTCH, RMUMPS\_PERM\_PORD, RMUMPS\_PERM\_METIS, RMUMPS\_PERM\_QAMD. This method should be called once and before symbolic analysis of the matrix. If it is called afterward, a new symbolic and numeric factorization will be performed when one of other methods (e.g. `solve()`) will request them. In other words, previous symbolic and numeric factorizations are canceled by this method.  
`get_permutation()`: get permutation type currently set in the object  
`mumps_version()`: Return a string with MUMPS version used in rmumps

### Note

When creating a symmetric matrix (`sym=1` or `sym=2`), the upper (or lower) part of the input matrix must be zeroed.

For meaning of entries in MUMPS vectors `cntl`, `icntl`, `info`, `rinfo`, `infof` and `rinfof` cf. original documentation of MUMPS project.

No need to call `symbolic()` and `numeric()` methods before a `solve()` call.

If in constructor, a parameter `copy` is set to `FALSE`, no rhs neither matrix copying is done. The solution is written "in place" thus overwriting rhs (watch out side effects)

For a detailed error diagnostic (e.g. when factorizing a singular matrix), use method `get_infos()` and cf. MUMPS documentation on the official MUMPS site).

### Author(s)

Serguei Sokol, INRA

### References

MUMPS official site <http://mumps.enseeiht.fr>

Sokol S (2020). `_Rmumps`: Rcpp port of MUMPS\_. `rmumps` package version 5.2.1-X, <URL: <http://CRAN.R-project.org/package=rmumps>>.

### Examples

```
## Not run:
# prepare random sparse matrix
library(Matrix)
library(rmumps)
```

```

n=2000
a=Matrix(0, n, n)
set.seed(7)
ij=sample(1:(n*n), 15*n)
a[ij]=runif(ij)
diag(a)=0
diag(a)=-rowSums(a)
a[1,1]=a[1,1]-1
am=Rmumps$new(a)
b=as.double(a%*(1:n)) # rhs for an exact solution vector 1:n
# following time includes symbolic analysis, LU factorization and system solving
system.time(x<-solve(am, b))
bb=2*b
# this second time should be much shorter
# as symbolic analysis and LU factorization are already done
system.time(xx<-solve(am, bb))
# compare to Matrix corresponding times
system.time(xm<-solve(a, b))
system.time(xxm<-solve(a, bb))
# compare to Matrix precision
range(x-1:n) # mumps
range(xm-1:n) # Matrix

# matrix inversion
system.time(aminv <- solve(am))
system.time(ainv <- solve(a)) # the same in Matrix

# symmetric matrix
asy=as(a+t(a), "symmetricMatrix")
bs=as.double(asy%*(1:n)) # rhs for 1:n solution
au=asy
# Here, we keep only diagonal and upper values of asy matrix.
# It could be also diagonal and lower values.
au[row(au)>col(au)]=0
ams=Rmumps$new(au, sym=1)
system.time(xs<-solve(ams, bs)) # rmumps
system.time(xsm<-solve(asy, bs)) # Matrix
# compare to Matrix precision
range(xs-1:n) # mumps
range(xsm-1:n) # Matrix

# clean up by hand to avoid possible interference between gc() and
# Rcpp object destructor after unloading this namespace
rm(am, ams)
gc()

## End(Not run)

```

**Description**

Integer constants defining permutation types and exported from rmumps are following:

- RMUMPS\_PERM\_AMD
- RMUMPS\_PERM\_AMF
- RMUMPS\_PERM\_SCOTCH
- RMUMPS\_PERM\_PORD
- RMUMPS\_PERM\_METIS
- RMUMPS\_PERM\_QAMD
- RMUMPS\_PERM\_AUTO

They are all regrouped in a named vector RMUMPS\_PERM where names are items above and values are corresponding constants.

**Examples**

```
am=rmumps::Rmumps$new(slam::as.simple_triplet_matrix(diag(1:3)))
am$set_permutation(RMUMPS_PERM_SCOTCH)
am$solve(1:3)
```

---

Rmumps__del_ptr	<i>Delete via Pointer</i>
-----------------	---------------------------

---

**Description**

This is a C wrapper to Rmumps::~~Rmumps() destructor. Available in R too. In C++ code can be used as rmumps::Rmumps\_\_del\_ptr(pm)

**Usage**

```
Rmumps__del_ptr(pm)
```

**Arguments**

pm                    pointer of type XPtr<Rmumps>, object to be deleted

---

Rmumps\_\_get\_permutation

*Get Permutation Parameter*


---

### Description

This is a C wrapper to `Rmumps::get_permutation()` method. Available in R too. In C++ code can be used as `rmumps::Rmumps__get_permutation(pm)`

### Usage

```
Rmumps__get_permutation(pm)
```

### Arguments

pm	pointer of type <code>XPtr&lt;Rmumps&gt;</code> , object having sparse matrix permuted according to some method.
----	--

### Value

integer defining permutation method used before matrix decomposition.

---

Rmumps\_\_ptr\_ijv

*Construct via Triplet Pointers*


---

### Description

This is a C wrapper to `Rmumps::Rmumps(i, j, v, n, nz, sym)` constructor. Available in R too. In C++ code can be used as `rmumps::Rmumps__ptr_ijv(pi, pj, pa, n, nz, sym)`

### Usage

```
Rmumps__ptr_ijv(pi, pj, pa, n, nz, sym)
```

### Arguments

pi	pointer of type <code>XPtr&lt;int&gt;</code> , vector of i-indeces for sparse triplet
pj	pointer of type <code>XPtr&lt;int&gt;</code> , vector of j-indeces for sparse triplet
pa	pointer of type <code>XPtr&lt;double&gt;</code> , vector or values for sparse triplet
n	integer, size of the matrix (n x n)
nz	integer, number of non zeros in the matrix
sym	integer, 0 means general (non symmetric) matrix, 1 - symmetric with pivotes on the main diagonal, 2 - general symmetric (pivotes may be anywhere)



**Value**

pointer of type XPtr<Rmumps> pointing to newly created object. To avoid memory leakage, it is user's responsibility to call Rmumps\_\_del\_ptr(pm) in a due moment (where pm is the returned pointer).

---

Rmumps\_\_set\_mat\_ptr     *Set Matrix via Pointer*

---

**Description**

This is a C wrapper to Rmumps::set\_mat\_ptr(a) method. Available in R too. In C++ code can be used as rmumps::Rmumps\_\_set\_mat\_ptr(pm). Using this method invalidates previous numeric decomposition (but not symbolic one).

**Usage**

```
Rmumps__set_mat_ptr(pm, pa)
```

**Arguments**

pm	pointer of type XPtr<Rmumps>, object having sparse matrix to be replaced with second parameter
pa	pointer of type XPtr<double>, value vector from sparse triplet providing a new matrix. Structure of the new matrix must be identical to the old one. That's why there is no need to provide i and j for the new triplet.

---

Rmumps\_\_set\_permutation  
*Set Permutation Parameter*

---

**Description**

This is a C wrapper to Rmumps::set\_permutation(permutation) method. Available in R too. In C++ code can be used as rmumps::Rmumps\_\_set\_permutation(pm, permutation)

**Usage**

```
Rmumps__set_permutation(pm, permutation)
```

**Arguments**

pm	pointer of type XPtr<Rmumps>, object having sparse matrix permuted according to a chosen method.
permutation	integer one of predefined constants (cf. <a href="#">RMUMPS_PERM</a> ). Setting a new permutation invalidates current symbolic and numeric matrix decompositions.

---

Rmumps\_\_solveptr      *Solve via Pointer*

---

### Description

This is a C wrapper to Rmumps::solveptr() method. Available in R too. In C++ code can be used as rmumps::Rmumps\_\_solveptr(pobj, pb, lrhs, nrhs)

### Usage

```
Rmumps__solveptr(pobj, pb, lrhs, nrhs)
```

### Arguments

pobj	pointer of type XPtr<Rmumps>, object having sparse matrix
pb	pointer of type XPtr<double>, vector or dense matrix of rhs
lrhs	integer, leading dimension in pb
nrhs	integer, number of rhs to solve.

---

Rmumps\_\_triplet      *Explore via Triplet*

---

### Description

This is a C wrapper to Rmumps::triplet() method. Available in R too. In C++ code can be used as rmumps::Rmumps\_\_triplet(pm)

### Usage

```
Rmumps__triplet(pm)
```

### Arguments

pm	pointer of type XPtr<Rmumps>, object having sparse matrix to be explored
----	--

### Value

a list with sparse triplet described with fields i, j, v

# Index

## \* classes

- Rcpp\_Rmumps-class, 4
- determinant.Rcpp\_Rmumps  
(Rcpp\_Rmumps-class), 4
- dim.Rcpp\_Rmumps (Rcpp\_Rmumps-class), 4
- ncol.Rcpp\_Rmumps (Rcpp\_Rmumps-class), 4
- nrow.Rcpp\_Rmumps (Rcpp\_Rmumps-class), 4
- print.Rcpp\_Rmumps (Rcpp\_Rmumps-class), 4
- Rcpp\_Rmumps-class, 4
- Rmumps (Rcpp\_Rmumps-class), 4
- rmumps (rmumps-package), 3
- rmumps-package, 3
- Rmumps\_\_del\_ptr, 7
- Rmumps\_\_get\_permutation, 8
- Rmumps\_\_ptr\_ijv, 8
- Rmumps\_\_set\_mat\_ptr, 9
- Rmumps\_\_set\_permutation, 9
- Rmumps\_\_solveptr, 10
- Rmumps\_\_triplet, 10
- RMUMPS\_PERM, 6, 9
- RMUMPS\_PERM\_AMD (RMUMPS\_PERM), 6
- RMUMPS\_PERM\_AMF (RMUMPS\_PERM), 6
- RMUMPS\_PERM\_AUTO (RMUMPS\_PERM), 6
- RMUMPS\_PERM\_METIS (RMUMPS\_PERM), 6
- RMUMPS\_PERM\_PORD (RMUMPS\_PERM), 6
- RMUMPS\_PERM\_QAMD (RMUMPS\_PERM), 6
- RMUMPS\_PERM\_SCOTCH (RMUMPS\_PERM), 6
- show.Rcpp\_Rmumps (Rcpp\_Rmumps-class), 4
- solve.Rcpp\_Rmumps (Rcpp\_Rmumps-class), 4
- solvet (Rcpp\_Rmumps-class), 4