Package: jmatrix (via r-universe)

September 4, 2024

Type Package

Title Read from/Write to Disk Matrices with any Data Type in a Binary Format

Version 1.5.2

Date 2024-07-05

Author Juan Domingo [aut, cre] (<https://orcid.org/0000-0003-4728-6256>), Guillermo Ayala [ctb] (<https://orcid.org/0000-0002-6231-2865>), Spanish Ministry of Science and Innovation, MCIN/AEI <doi:10.13039/501100011033> [fnd]

Maintainer Juan Domingo <Juan.Domingo@uv.es>

Description A mainly instrumental package meant to allow other packages whose core is written in 'C++' to read, write and manipulate matrices in a binary format so that the memory used for them is no more than strictly needed. Its functionality is already inside 'parallelpam' and 'scellpam', so if you have installed any of these, you do not need to install 'jmatrix'. Using just the needed memory is not always true with 'R' matrices or vectors, since by default they are of double type. Trials like the 'float' package have been done, but to use them you have to coerce a matrix already loaded in 'R' memory to a float matrix, and then you can delete it. The problem comes when your computer has not memory enough to hold the matrix in the first place, so you are forced to load it by chunks. This is the problem this package tries to address (with partial success, but this is a difficult problem since 'R' is not a strictly typed language, which is anyway quite hard to get in an interpreted language). This package allows the creation and manipulation of full, sparse and symmetric matrices of any standard data type.

License GPL (>= 2)

Imports Rcpp (>= 1.0.8), memuse (>= 4.2.1)

LinkingTo Rcpp

RoxygenNote 7.2.3

Encoding UTF-8

Suggests knitr

VignetteBuilder knitr

NeedsCompilation yes

Date/Publication 2024-07-05 23:50:01 UTC

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

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

RemoteRef HEAD

RemoteSha 73adb823e41f26563453b1d006be335fb73ba4e0

Contents

`CsvToJMat` *CsvToJMat*

Description

Gets a csv/tsv file and writes to a disk file the binary matrix of counts contained in it in the jmatrix binary format.
First line of the .csv is supposed to have the field names.
First column of each line is supposed to have the row name.
The fields are supposed to be separated by one occurrence of a character-field sepparator (usually, comma or tab) .tsv files can be read with this function, too, setting the csep argument to '\t'

Usage

```
CsvToJMat(
  ifname,
  ofname,
  mtype = "sparse",
  csep = ",",
  ctype = "raw",
  valuetype = "float",
  transpose = FALSE,
  comment = ""
)
```

Arguments

ifname	A string with the name of the .csv/.tsv text file.
ofname	A string with the name of the binary output file.
mtype	A string to indicate the matrix type: 'full', 'sparse' or 'symmetric'. Default: 'sparse'
csep	The character used as separator in the .csv file. Default: ',' (comma) (Set to '\t' for .tsv)
ctype	The string 'raw' or 'log1' to write raw counts or log(counts+1), or the normalized versions, 'rawn' and 'log1n', which normalize ALWAYS BY COLUMNS (before transposition, if requested to transpose). The logarithm is taken base 2. Default: raw
valuetype	The data type to store the matrix. It must be one of the strings 'uint32', 'float' or 'double'. Default: float
transpose	Boolean to indicate if the matrix should be transposed before writing. See Details for a comment about this. Default: FALSE
comment	A comment to be stored with the matrix. Default: "" (no comment)

Details

The parameter transpose has the default value of FALSE. But don't forget to set it to TRUE if you want the cells (which in single cell common practice are by columns) to be written by rows. This will be needed later to calculate the dissimilarity matrix, if this is the next step of your workflow. See help of CalcAndWriteDissimilarityMatrix

Special note for loading symmetric matrices:
If you use this function to load what you expect to be a symmetric matrix from a .csv file, remember that the input table MUST be square, but only the lower-diagonal matrix will be stored, including the main diagonal. The rest of the input table is completely ignored, except to check that there are values in it. It is not checked if the table really represents a symmetric matrix or not.
Furthermore, symmetric matrices can only be loaded in raw mode, i.e.: no normalization is allowed, and they cannot be transposed.

Value

No return value, called for side effects (creates a file)

Examples

```
# Since we have no a .csv file to test, we will generate one with another funcion of this package
Rf <- matrix(runif(48),nrow=6)
rownames(Rf) <- c("A","B","C","D","E","F")
colnames(Rf) <- c("a","b","c","d","e","f","g","h")
tmpfile1=paste0(tempdir(),"/Rfullfloat.bin")
tmpfile2=paste0(tempdir(),"/Rfullfloat2.bin")
tmpcsvfile1=paste0(tempdir(),"/Rfullfloat.csv")
JWriteBin(Rf,tmpfile1,dtype="float",dmtype="full",comment="Full matrix of floats")
JMatToCsv(tmpfile1,tmpcsvfile1)
CsvToJMat(tmpcsvfile1,tmpfile2)
# It can be checked that files Rfullfloat.bin and Rfullfloat2.bin contain the same data
# (even they differ in the comment, which has been eliminated when converting to csv)
```

`FilterJMatByName` *FilterJMatByName*

Description

Takes a jmatrix binary file containing a table with rows and columns and filters it by name, eliminating the rows or columns whose whose names are not in certain list

Usage

```
FilterJMatByName(fname, Gn, filname, namesat = "rows")
```

Arguments

`fname`	A string with the file name of the original table
`Gn`	A list of R strings with the names of the rows or columns that must remain. All others will be filtered out
`filname`	A string with the file name of the filtered table
`namesat`	The string "rows" or "cols" indicating if the searched names are in the rows or in the columns of the original table. Default: "rows"

Details

If the table has no list of names in the requested dimension (rows or colums), an error is rised.
The row or column names whose names are not found obviosuly cannot remain, and the program rises a warning indicating for which row/column names this happens.
The matrix contained in the filtered file will have the same nature (full or sparse) and the same data type as the original.
This function can be used to filter either by row or by column name, with appropriate usage of parameter namesat

Value

No return value, called for side effects (creates a file)

Examples

```
Rf <- matrix(runif(48),nrow=6)
rownames(Rf) <- c("A","B","C","D","E","F")
colnames(Rf) <- c("a","b","c","d","e","f","g","h")
tmpfile1=paste0(tempdir(),"/Rfullfloat.bin")
tmpfile2=paste0(tempdir(),"/Rfullfloatrowfilt.bin")
tmpfile3=paste0(tempdir(),"/Rfullfloatrowcolfilt.bin")
tmpcsvfile1=paste0(tempdir(),"/Rfullfloat.csv")
tmpcsvfile3=paste0(tempdir(),"/Rfullfloatrowcolfilt.csv")
JWriteBin(Rf,tmpfile1,dtype="float",dmtype="full",comment="Full matrix of floats")
# Let's keep only rows A, C and E
FilterJMatByName(tmpfile1,c("A","C","E"),tmpfile2,namesat="rows")
# and from the result, let's keep only columns b, d and g
FilterJMatByName(tmpfile2,c("b","d","g"),tmpfile3,namesat="cols")
JMatToCsv(tmpfile1,tmpcsvfile1)
JMatToCsv(tmpfile3,tmpcsvfile3)
# You can now compare both ASCII/csv files
```

`GetJCol` *GetJCol*

Description

Returns (as a R numeric vector) the requested column number from the matrix contained in a jmatrix binary file

Usage

```
GetJCol(fname, ncol)
```

Arguments

`fname`	String with the file name that contains the binary data.
`ncol`	The number of the column to be returned, in R-numbering (from 1)

Value

A numeric vector with the values of elements in the requested column

Examples

```
Rf <- matrix(runif(48),nrow=6)
rownames(Rf) <- c("A","B","C","D","E","F")
colnames(Rf) <- c("a","b","c","d","e","f","g","h")
tmpfile1=paste0(tempdir(),"/Rfullfloat.bin")
JWriteBin(Rf,tmpfile1,dtype="float",dmtype="full",comment="Full matrix of floats")
Rf[,3]
vf<-GetJCol(tmpfile1,3)
vf
```

GetJColByName *GetJColByName*

Description

Returns (as a R numeric vector) the requested named column from the matrix contained in a jmatrix binary file

Usage

```
GetJColByName(fname, colname)
```

Arguments

fname	String with the file name that contains the binary data.
colname	The name of the column to be returned. If the matrix has no column names, or the name is not found, an empty vector is returned

Value

A numeric vector with the values of elements in the requested column

Examples

```
Rf <- matrix(runif(48),nrow=6)
rownames(Rf) <- c("A","B","C","D","E","F")
colnames(Rf) <- c("a","b","c","d","e","f","g","h")
tmpfile1=paste0(tempdir(),"/Rfullfloat.bin")
JWriteBin(Rf,tmpfile1,dtype="float",dmtype="full",comment="Full matrix of floats")
Rf[,"c"]
vf<-GetJColByName(tmpfile1,"c")
vf
```

GetJColNames *GetJColNames*

Description

Returns a R StringVector with the column names of a matrix stored in the binary format of package jmatrix, if it has them stored.

Usage

```
GetJColNames(fname)
```

Arguments

fname	String with the file name that contains the binary data.

Value

A R StringVector with the column names, or the empty vector if the binaryfile has no column names as metadata.

Examples

```
Rf <- matrix(runif(48),nrow=6)
rownames(Rf) <- c("A","B","C","D","E","F")
colnames(Rf) <- c("a","b","c","d","e","f","g","h")
tmpfile1=paste0(tempdir(),"/Rfullfloat.bin")
JWriteBin(Rf,tmpfile1,dtype="float",dmtype="full",comment="Full matrix of floats")
cn<-GetJColNames(tmpfile1)
cn
```

GetJManyCols *GetJManyCols*

Description

Returns (as a R numeric matrix) the columns with the requested column numbers from the matrix contained in a jmatrix binary file

Usage

```
GetJManyCols(fname, extcols)
```

Arguments

fname	String with the file name that contains the binary data.
extcols	A numeric vector with the indexes of the columns to be extracted, in R-numbering (from 1)

Value

A numeric matrix with the values of elements in the requested columns

Examples

```
Rf <- matrix(runif(48),nrow=6)
rownames(Rf) <- c("A","B","C","D","E","F")
colnames(Rf) <- c("a","b","c","d","e","f","g","h")
tmpfile1=paste0(tempdir(),"/Rfullfloat.bin")
JWriteBin(Rf,tmpfile1,dtype="float",dmtype="full",comment="Full matrix of floats")
vc<-GetJManyCols(tmpfile1,c(1,4))
vc
```

GetJManyColsByNames *GetJManyColsByNames*

Description

Returns (as a R numeric matrix) the columns with the requested column names from the matrix contained in a jmatrix binary file

Usage

```
GetJManyColsByNames(fname, extcolnames)
```

Arguments

fname	String with the file name that contains the binary data.
extcolnames	A vector of RStrings with the names of the columns to be extracted. If the binary file has no column names, or _any_ of the column names is not present, an empty matrix is returned.

Value

A numeric matrix with the values of elements in the requested columns

Examples

```
Rf <- matrix(runif(48),nrow=6)
rownames(Rf) <- c("A","B","C","D","E","F")
colnames(Rf) <- c("a","b","c","d","e","f","g","h")
tmpfile1=paste0(tempdir(),"/Rfullfloat.bin")
JWriteBin(Rf,tmpfile1,dtype="float",dmtype="full",comment="Full matrix of floats")
Rf[,c(1,4)]
vf<-GetJManyColsByNames(tmpfile1,c("a","d"))
vf
```

GetJManyRows *GetJManyRows*

Description

Returns (as a R numeric matrix) the rows with the requested row numbers from the matrix contained in a jmatrix binary file

Usage

```
GetJManyRows(fname, extrows)
```

Arguments

fname	String with the file name that contains the binary data.
extrows	A numeric vector with the indexes of the rows to be extracted, in R-numbering (from 1)

Value

A numeric matrix with the values of elements in the requested rows

Examples

```
Rf <- matrix(runif(48),nrow=6)
rownames(Rf) <- c("A","B","C","D","E","F")
colnames(Rf) <- c("a","b","c","d","e","f","g","h")
tmpfile1=paste0(tempdir(),"/Rfullfloat.bin")
JWriteBin(Rf,tmpfile1,dtype="float",dmtype="full",comment="Full matrix of floats")
Rf[c(1,4),]
vc<-GetJManyRows(tmpfile1,c(1,4))
vc
```

GetJManyRowsByNames *GetJManyRowsByNames*

Description

Returns (as a R numeric matrix) the rows with the requested row names from the matrix contained in a jmatrix binary file

Usage

```
GetJManyRowsByNames(fname, extrownames)
```

Arguments

fname	String with the file name that contains the binary data.
extrownames	A vector of RStrings with the names of the rows to be extracted. If the binary file has no row names, or _any_ of the row names is not present, an empty matrix is returned.

Value

A numeric matrix with the values of elements in the requested rows

Examples

```
Rf <- matrix(runif(48),nrow=6)
rownames(Rf) <- c("A","B","C","D","E","F")
colnames(Rf) <- c("a","b","c","d","e","f","g","h")
tmpfile1=paste0(tempdir(),"/Rfullfloat.bin")
JWriteBin(Rf,tmpfile1,dtype="float",dmtype="full",comment="Full matrix of floats")
Rf[c("A","C"),]
vf<-GetJManyRowsByNames(tmpfile1,c("A","C"))
vf
```

GetJNames *GetJNames*

Description

Returns a R list of two elements, rownames and colnames, each of them being a R StringVector with the corresponding names

Usage

```
GetJNames(fname)
```

Arguments

fname	String with the file name that contains the binary data.

Value

N["rownames","colnames"]: A list with two elements named rownames and colnames which are R StringVectors. If the binary file has no row or column names as metadata BOTH will be returned as empty vectors, even if one of them exists. If you want to extract only one, use either GetJRowNames or GetJColNames, as appropriate.

Examples

```
Rf <- matrix(runif(48),nrow=6)
rownames(Rf) <- c("A","B","C","D","E","F")
colnames(Rf) <- c("a","b","c","d","e","f","g","h")
tmpfile1=paste0(tempdir(),"/Rfullfloat.bin")
JWriteBin(Rf,tmpfile1,dtype="float",dmtype="full",comment="Full matrix of floats")
N<-GetJNames(tmpfile1)
N["rownames"]
N["colnames"]
```

GetJRow *GetJRow*

Description

Returns (as a R numeric vector) the requested row number from the matrix contained in a jmatrix binary file

Usage

```
GetJRow(fname, nrow)
```

Arguments

fname	String with the file name that contains the binary data.
nrow	The number of the row to be returned, in R-numbering (from 1)

Value

A numeric vector with the values of elements in the requested row

Examples

```
Rf <- matrix(runif(48),nrow=6)
rownames(Rf) <- c("A","B","C","D","E","F")
colnames(Rf) <- c("a","b","c","d","e","f","g","h")
tmpfile1=paste0(tempdir(),"/Rfullfloat.bin")
JWriteBin(Rf,tmpfile1,dtype="float",dmtype="full",comment="Full matrix of floats")
Rf[3,]
vf<-GetJRow(tmpfile1,3)
vf
```

GetJRowByName *GetJRowByName*

Description

Returns (as a R numeric vector) the requested named row from the matrix contained in a jmatrix binary file

Usage

```
GetJRowByName(fname, rowname)
```

Arguments

fname	String with the file name that contains the binary data.
rowname	The name of the row to be returned. If the matrix has no row names, or the name is not found, an empty vector is returned

Value

A numeric vector with the values of elements in the requested row

Examples

```
Rf <- matrix(runif(48),nrow=6)
rownames(Rf) <- c("A","B","C","D","E","F")
colnames(Rf) <- c("a","b","c","d","e","f","g","h")
tmpfile1=paste0(tempdir(),"/Rfullfloat.bin")
JWriteBin(Rf,tmpfile1,dtype="float",dmtype="full",comment="Full matrix of floats")
Rf["C",]
vf<-GetJRowByName(tmpfile1,"C")
vf
```

GetJRowNames *GetJRowNames*

Description

Returns a R StringVector with the row names of a matrix stored in the binary format of package jmatrix, if it has them stored.

Usage

```
GetJRowNames(fname)
```

Arguments

fname	String with the file name that contains the binary data.

Value

A R StringVector with the row names, or the empty vector if the binary file has no row names as metadata.

Examples

```
Rf <- matrix(runif(48),nrow=6)
rownames(Rf) <- c("A","B","C","D","E","F")
colnames(Rf) <- c("a","b","c","d","e","f","g","h")
tmpfile1=paste0(tempdir(),"/Rfullfloat.bin")
JWriteBin(Rf,tmpfile1,dtype="float",dmtype="full",comment="Full matrix of floats")
rn<-GetJRowNames(tmpfile1)
rn
```

JMatInfo *JMatInfo*

Description

Shows in the screen or writes to a file information about a matrix stored in the binary format of package jmatrix

Usage

```
JMatInfo(fname, fres = "")
```

Arguments

fname	String with the file name that contains the binary data.
fres	String with the name of the file to write the information. Default: "" (information is written to the console)

Value

No return value, called for its side effects (writes on screen or creates a file)

Examples

```
Rf <- matrix(runif(48),nrow=6)
rownames(Rf) <- c("A","B","C","D","E","F")
colnames(Rf) <- c("a","b","c","d","e","f","g","h")
tmpfile1=paste0(tempdir(),"/Rfullfloat.bin")
JWriteBin(Rf,tmpfile1,dtype="float",dmtype="full",comment="Full matrix of floats")
JMatInfo(tmpfile1)
```

JMatrixSetDebug *JMatrixSetDebug*

Description

Sets debugging in jmatrix package to ON (with TRUE) or OFF (with FALSE).
On package load the default status is OFF.
Setting debugging to ON shows a message. Setting to OFF does not show anything (since debugging is OFF...)

Usage

```
JMatrixSetDebug(deb = TRUE)
```

Arguments

deb	boolean, TRUE to generate debug messages and FALSE to turn them off. Default: true

Value

No return value, called for side effects (internal boolean flag changed)

Examples

```
JMatrixSetDebug(TRUE)
JMatrixSetDebug(FALSE)
```

JMatToCsv *JMatToCsv*

Description

Writes a binary matrix in the jmatrix package format as a .csv file. This is mainly for checking/inspection and to load the data from R as read.csv, if the memory of having all data as doubles allows doing such thing.

Usage

```
JMatToCsv(ifile, csvfile, csep = ",", withquotes = FALSE)
```

Arguments

ifile	String with the file name that contains the binary data.
csvfile	String with the file name that will contain the data as csv.
csep	Character used as separator. Default: , (comma)
withquotes	boolean to mark if row and column names in the .csv file must be written surrounded by doble quotes. Default: FALSE

Details

The numbers are written to text with as many decimal places as allowed by its data type (internally obtained with std::numeric_limits<type>::max_digits10)
NOTE ON READING FROM R: to read the .csv files exported by this function you MUST use the R function read.csv (not read.table) AND set its argument row.names to 1, since we always write a first column with the row names, even if the binary matrix does not store them; in this case they are simply "1","2",...

Value

No return value, called for side effects (creates a file)

Examples

```
Rf <- matrix(runif(48),nrow=6)
rownames(Rf) <- c("A","B","C","D","E","F")
colnames(Rf) <- c("a","b","c","d","e","f","g","h")
tmpfile1=paste0(tempdir(),"/Rfullfloat.bin")
tmpcsvfile1=paste0(tempdir(),"/Rfullfloat.csv")
JWriteBin(Rf,tmpfile1,dtype="float",dmtype="full",comment="Full matrix of floats")
JMatToCsv(tmpfile1,tmpcsvfile1)
```

JWriteBin *JWriteBin*

Description

Writes a R matrix to a disk file as a binary matrix in the jmatrix format

Usage

```
JWriteBin(M, fname, dtype = "float", dmtype = "full", comment = "")
```

Arguments

M	The R matrix to be written
fname	The name of the file to write
dtype	The data type of the matrix to be written: one of the strings 'short', 'int', 'long', 'float' or 'double'. Default: 'float'
dmtype	The matrix type: one of the strings 'full', 'sparse' or 'symmetric'. Default: 'full'
comment	A optional string with the comment to be added as metadata. Default: "" (empty string, no added comment)

Details

Use this function cautiously. Differently to the functions to get one or more rows or columns from the binary file, which book only the memory strictly needed for the vector/matrix and do not load all the binary file in memory, this function books the full matrix in the requested data type and writes it later so with very big matrices you might run out of memory.
Type 'int' is really long int (8-bytes in most modern machines) so using 'int' or 'long' is equivalent.
Type is coerced from double (the internal type of R matrices) to the requested type, which may provoke a loose of precision.
If M is a named-R matrix, row and column names are written as metadata, too.
Also, if you write as symmetric a matrix which is not such, only the lower-diagonal part will be written. The rest of the data will be lost. In this case, if the matrix has row and column names, only row names are written.

Value

No return value, called for side effects (creates a file)

Examples

```
Rf <- matrix(runif(48),nrow=6)
rownames(Rf) <- c("A","B","C","D","E","F")
colnames(Rf) <- c("a","b","c","d","e","f","g","h")
tmpfile1=paste0(tempdir(),"/Rfullfloat.bin")
JWriteBin(Rf,tmpfile1,dtype="float",dmtype="full",comment="Full matrix of floats")
```

Index