BigMatrix-class  
Class "BigMatrix"

Description

The “BigMatrix” class is designed for massive matrices, stored in C, of type double, integer, short, or char. Through various methods, a “BigMatrix” acts much like a traditional R matrix, but helps protect the user from many inadvertent memory-consuming pitfalls of traditional R matrices and data frames. That said, we caution the user against casual use of the big matrices. In Unix, they may also be used in shared memory.

Objects from the Class

Objects can be created by calls of the form `new("BigMatrix", ...)`. The functions `BigMatrix()` and `BigSharedMatrix` are intended for the user.

Slots

- `address`: Object of class "externalptr" points to the memory location of the C data structure.
- `type`: Object of class "character" can be one of "double", "integer", "short", or "char", using 8, 4, 2, and 1 bytes, respectively, per element.

Methods

```
[- signature(x = "BigMatrix", i = "numeric", j = "numeric", value = "ANY"):
  ...
[- signature(x = "BigMatrix", i = "numeric", j = "character", value = "ANY"):
  ...
[- signature(x = "BigMatrix", i = "numeric", j = "missing", value = "numeric"):
  ...
[- signature(x = "BigMatrix", i = "missing", j = "numeric", value = "numeric"):
  ...
[- signature(x = "BigMatrix", i = "missing", j = "character", value = "numeric"):
  ...
[- signature(x = "BigMatrix", i = "missing", j = "missing", value = "numeric"):
  ...
[ signature(x = "BigMatrix", i = "numeric", j = "numeric", drop = "missing"):
  ...
[ signature(x = "BigMatrix", i = "numeric", j = "character", drop = "missing"):
  ...
[ signature(x = "BigMatrix", i = "numeric", j = "missing", drop = "missing"):
  ...
```
[signature(x = "BigMatrix", i = "missing", j = "numeric", drop = "missing"):
...
signature(x = "BigMatrix", i = "missing", j = "character", drop = "missing"):
...
signature(x = "BigMatrix", i = "missing", j = "missing", drop = "missing"):
...
signature(x = "BigMatrix", i = "logical", j = "missing", drop = "missing"):
...
signature(x = "BigMatrix", i = "logical", j = "logical", drop = "missing"):
...
signature(x = "BigMatrix", i = "logical", j = "missing", drop = "missing"):
...

colmax signature(x = "BigMatrix"): find the maximum of each column (or the specified columns, optionally).
colmean signature(x = "BigMatrix"): find the mean of each column (or the specified columns, optionally).
colmin signature(x = "BigMatrix"): find the min of each column (or the specified columns, optionally).
colrange signature(x = "BigMatrix"): find the range of each column (or the specified columns, optionally).
colstd signature(x = "BigMatrix"): find the standard deviation of each column (or the specified columns, optionally).
colvar signature(x = "BigMatrix"): find the variance of each column (or the specified columns, optionally).
dim signature(x = "BigMatrix"): find the dimension of the BigMatrix.
dimnames<- signature(x = "BigMatrix", value = "list"): set the row and column names.
dimnames signature(x = "BigMatrix"): get the row and column names.
head signature(x = "BigMatrix"): get the first 6 (or n) rows.
max signature(x = "BigMatrix"): find the maximum of all the values.
mean signature(x = "BigMatrix"): find the mean of all the values.
min signature(x = "BigMatrix"): find the minimum of all the values.
ncol signature(x = "BigMatrix"): get the number of columns.
nrow signature(x = "BigMatrix"): get the number of rows.
print signature(x = "BigMatrix"): print is intentionally disabled, and returns head(x) unless options()$bm.print.warning==FALSE; in this case, print(x[,]) is the result, which could be very big!
range signature(x = "BigMatrix"): find the range of all the values.
summary signature(object = "BigMatrix"): produce a summary of each of the columns.
tail signature(x = "BigMatrix"): get the last 6 (of n) rows.
write.bm signature(bigMat = "BigMatrix", fileName = "character"): produce a file from the BigMatrix.
is.shared signature(x = "BigMatrix"): is this object in shared memory?
Author(s)
John W. Emerson and Michael J. Kane

See Also

BigMatrix

Examples

showClass("BigMatrix")

BigMatrix, is.BigMatrix, as.BigMatrix

The basic “BigMatrix” operations.

Description

Create a BigMatrix (or check to see if an object is a BigMatrix, or create a BigMatrix from a matrix).

Usage

BigMatrix(nrow, ncol, type = "integer", init = 0,
          colnames = NULL, rownames = NULL)
as.BigMatrix(x, type = "integer")
is.BigMatrix(x)

Arguments

x an object; if a vector, a one-column BigMatrix is created by as.BigMatrix.
nrow number of rows.
ncol number of columns.
type the type of the atomic element ("integer" by default).
init a scalar value for initializing the matrix, 0 by default.
colnames a vector of column names.
rownames a vector or row names (see details below for a word of caution).
Details

A BigMatrix consists of an object in R that does little more than point to the data structure implemented in C. Through various methods, the object acts much like a traditional R matrix, but helps protect the user from many inadvertent memory-consuming pitfalls of traditional R matrices and data frames. That said, we caution the user against casual use of these big matrices.

Four atomic types are implemented (see argument type, above) to help provide memory efficiency in different applications: double (equivalent to numeric in R), integer (using 4 bytes), short (using 2 bytes), and char (using a single byte). The value NA only exists for double and integer matrices, and char behaves like a signed integer.

If \( x \) is a BigMatrix, then \( x[1:5,] \) is evaluated as an R matrix containing the first five rows of \( x \); if \( x \) is of type double, then the result will be numeric; otherwise, the result will be an integer R matrix. For “safety,” the expression \( x \) alone will display information about the R object (e.g. the type) rather than printing the matrix itself (the user should try \( x[,] \) with extreme caution to print the entire matrix, recognizing that a huge temporary R version of the matrix will exist in the process).

If \( x \) has a huge number of rows (very likely with large data sets), then the use of rownames will be extremely memory-intensive and should be avoided.

Finally, we note that when a BigMatrix, \( x \), is passed as an argument to a function, it is essentially providing call-by-reference rather than call-by-value behavior. If the function modified any of the values of \( x \) within the function, the changes are really made (and not limited in scope to a local copy within the function). This will be both an advantage and disadvantage.

Value

A BigMatrix is returned (for BigMatrix and as.BigMatrix), and TRUE or FALSE for is.BigMatrix.

Author(s)

John W. Emerson and Michael J. Kane

References

See http://www.stat.yale.edu/~jay/bigmemoRy.

See Also

bigmemoRy, and perhaps the class documentation of BigMatrix.

Examples

\[
x <- \text{BigMatrix}(10, 2, \text{type='integer'}, \text{init=-5, colnames=c("alpha", "beta")})
\]
\[
is.\text{BigMatrix}(x)
\]
\[
dim(x)
\]
\[
x[1:8,1] <- 11:18
\]
\[
x[,]\]

4
y <- as.BigMatrix(matrix(1:20, 10, 2))
y[,]
cbind(x[,2], y[,1])
y[,2] <- x[,1]
head(y)

# which rows contain 15 in either columns 1 or 2?
which.bm(y, 1:2, 15, 'eq')

# which rows contain the value 4 or 6 in column 1, or 15 in column 2?
which.bm(y, c(1,1,2), list(4, 6, 15), 'eq', 'OR')

# which rows contain values between 4 and 6 inclusive in column 1, 
# and the value in column 2 is not equal to 15?
which.bm(y, 1:2, list(c(4,6), 15), list(c('ge','le'), 'neq'), 'AND')

# Testing 'neq' mostly because this was an issue in the latest
# redesign.
which.bm(y, 1, 2, 'neq')
which.bm(y, 2, 12, 'neq')

which.bm(y, 1:2, list(3, 12), list('neq', 'neq'), 'OR')
which.bm(y, 1:2, list(3, 12), list('neq', 'neq'), 'AND')

which.bm(y, 1:2, list(3, 12), list('eq', 'neq'), 'OR')
which.bm(y, 1:2, list(3, 12), list('neq', 'eq'), 'AND')

## Not run:
# This example won't run on the Windows version of the package.
# It's also a bit silly, as you wouldn't likely do this on a single
# processor. But if zdescription were passed to another R process
# via Rmpi, Networkspaces, or even by a simple file read/write,
# then the attach on the second R process would give access to the
# same object in memory.
z <- BigSharedMatrix(3, 3, type='integer', init=3)
z[,]
dim(z)
z[1,1] <- 2
z[,]
zdescription <- DescribeBigSharedMatrix(z)
zdescription
y <- AttachBigSharedMatrix(zdescription)
y[,]
y z
y[1,1] <- -100
y[,]
z[,]
## End(Not run)
BigSharedMatrix, DescribeBigSharedMatrix, AttachBigSharedMatrix

The basic “BigSharedMatrix” operations.

Description
Create a BigMatrix in shared memory (or check to see if an object is in shared memory).

Usage

BigSharedMatrix(nrow, ncol, type = "integer",
    init = 0, colnames = NULL, rownames = NULL)
DescribeBigSharedMatrix(x, file = NULL, path)
AttachBigSharedMatrix(obj, path)

Arguments

x a shared BigMatrix.
nrow number of rows.
col number of columns.
type the type of the atomic element ("integer" by default).
init a scalar value for initializing the matrix; this cannot be a matrix or a vector of values.
colnames a vector of column names.
rownames a vector of row names, which we recommend avoiding when nrow(x) is large.
obj either a file name or an object as returned by DescribeBigSharedMatrix.
file a file name if used to store the description of the BigMatrix.
path a path to be used if obj is a file name, and must terminate in ‘/’.

Details
A shared BigMatrix consists of an object in R that does little more than point to the data structure implemented in shared memory in C. Through various method functions, the object acts much like a traditional R matrix, but helps protect the user from many inadvertent memory-consuming pitfalls of traditional R matrices and data frames. That said, we caution the user against casual use of these big matrices.

Value
A shared memory BigMatrix is returned by each of these functions. BigSharedMatrix creates a new matrix in shared memory, while AttachSharedBigMatrix creates the R BigMatrix object referencing an existing matrix in shared memory.
Author(s)
John W. Emerson and Michael J. Kane

References
http://www.stat.yale.edu/~jay/bigmemRy.

See Also
bigmemRy, BigMatrix, or the class documentation BigMatrix.

Examples
# This example won't run on the Windows version of the package,
# and if you are reading this message you are most certainly using
# a non-Windows version.

# The example is also a bit silly, as you wouldn't likely do this on a
# single R session. But if zdescription were passed to another R session
# via Rmpi, Networkspaces, or even by a simple file read/write,
# then the attach on the second R process would give access to the
# same object in memory.
z <- BigSharedMatrix(3, 3, type='integer', init=3)
z[,] dim(z) z[1,1] <- 2 z[,]
zdescription <- DescribeBigSharedMatrix(z)
zdescription
y <- AttachBigSharedMatrix(zdescription)
y[,] y z y[1,1] <- -100 y[,] z[,]
Slots

address: Object of class "externalptr"

Methods

UserRW MutexInfo signature(x = "UserRW Mutex"): ...

Warning

We probably do have warnings.

Note

Additional notes.

Author(s)

Michael Kane and John W. Emerson

References

pthread.h

See Also

BigMatrix

Examples

showClass("UserRW Mutex")

UserRW Mutex, ConnectUserRW Mutex, UserRW MutexInfo

Mutual exclusion functionality for shared memory matrices in package "bignemoRy"

Description

These functions provide mutexes (mutual exclusions) for use with matrices in shared memory (class BigMatrix of bignemoRy).

Usage

UserRW Mutex()
ConnectUserRW Mutex(mutexId, countId, countMutexId)
UserRW MutexInfo(x)
Arguments

\[ x \quad \text{an UserRWMutex} \]
\[ \text{mutexId} \quad \text{the shared memory key for the mutex.} \]
\[ \text{countId} \quad \text{the key for the mutex counter.} \]
\[ \text{countMutexId} \quad \text{the key for the mutex on the counter itself.} \]

Details

Mutexes are provided with every shared matrix and managed by the various functions provided for \texttt{BigMatrix} objects (\texttt{which.bm}, for example). However, the user may create an additional layer of mutexes using these functions. This may be important in certain shared memory applications.

Value

The only function that returns anything interesting is \texttt{UserRWMutexInfo}. This returns a vector of three values which correspond to the arguments of \texttt{ConnectUserRWMutex}.

Note

Shared memory is not presently supported by the Windows version of \texttt{bigmemoRy} (but you wouldn’t be reading this comment if you had the Windows version).

Author(s)

John W. Emerson and Michael Kane

References

C libraries pthread, ipc, and shm are used for shared memory and mutexes.

See Also

\texttt{BigMatrix}, \texttt{BigSharedMatrix}

Examples

\# None.
add.cols.bm, rm.cols.bm

Add and remove columns of a “BigMatrix”.

Description
Add and remove columns of a BigMatrix.

Usage
add.cols.bm(x, numCols = 1, init = 0, names = NULL)
rm.cols.bm(x, rmCols)

Arguments
x a BigMatrix.
numCols the number of columns to add.
rmCols a vector of integer indices or variable names to remove.
init a scalar value for initializing the column; this may not be a vector or a matrix of values.
names a vector of numCols names (optional).

Details
Adding and removing columns of a BigMatrix is efficient, with no memory overhead because of the data structure. Note that these operations are not permitted if the matrix is in shared memory (if is.shared(x) is TRUE).

Value
None; the BigMatrix x is actually modified.

Author(s)
John W. Emerson and Michael J. Kane

See Also
BigMatrix
Examples

```
x <- BigMatrix(5, 2, init=1, colnames=c("alpha", "beta"))
add.cols.bm(x, 1, names="gamma")
x[,]  
rm.cols.bm(x, 2)
x[,]  
add.cols.bm(x, 2, names=c("jay", "mike"))
rm.cols.bm(x, "alpha")
x[,]  
```

biglm.bm, bigglm.bm  Use Lumley’s “biglm” package with a “BigMatrix”

Description

This is a wrapper to Lumley’s biglm package, allowing its use with data stored in BigMatrix objects.

Usage

```
biglm.bm(formula, data, fc=NULL, chunksize=NULL, weights=NULL, sandwich=FALSE)
bigglm.bm(formula, data, family=gaussian(), fc=NULL, chunksize=NULL, weights=NULL, sandwich=FALSE, maxit=8, tolerance=1e-7, start=NULL)
```

Arguments

- **formula**: a model formula.
- **data**: a BigMatrix.
- **fc**: either column indices or names of variables that are factors.
- **chunksize**: an integer maximum size of chunks of data to process iteratively.
- **weights**: a one-sided, single term formula specifying weights (see biglm for more information).
- **sandwich**: TRUE to compute the Huber/White sandwich covariance matrix (see biglm for more information).
- **family**: a glm family object
- **maxit**: maximum number of Fisher scoring iterations.
- **tolerance**: tolerance for change in coefficient (as multiple of standard error).
- **start**: optional starting values for coefficients. If NULL, maxit should be at least 2 as some quantities will not be computed on the first iteration.

Details

See Thomas Lumley’s biglm package for more information; chunksize defaults to floor(nrow(data)/ncol(data)^2).
Value

an object of class biglm.

Author(s)

Michael J. Kane

References


See Also

biglm, BigMatrix

Examples

# This example is quite silly, creating an integer data set from the iris
# data. But it shows that our wrapper to Lumley's biglm() function produces
# the same answer as the plain old lm() function.

x <- matrix(as.integer(unlist(iris)), ncol=5)
colnames(x) <- names(iris)
x <- as.BigMatrix(x)
head(x)
silly.biglm <- biglm.bm(Sepal.Length ~ Sepal.Width + Species, data=x, fc="Species")
summary(silly.biglm)
y <- data.frame(x[,])
y$Species <- as.factor(y$Species)
head(y)
silly.lm <- lm(Sepal.Length ~ Sepal.Width + Species, data=y)
summary(silly.lm)

bigmemoRy-package

bigmemoRy: massive matrices in (possibly shared) memory.

Description

bigmemoRy implements massive matrices in C (optionally, in shared memory) and supports the manipulation and exploration of these objects. It provides a framework for the development of C code for interactive use with R.

Details
Multi-gigabyte data sets challenge and frustrate R users even on well-equipped hardware. C programming can provide memory efficiency and speed improvements, but is cumbersome for interactive data analysis and lacks the flexibility and power of R’s rich statistical programming environment. **bigmemoRy** bridges this gap, implementing persistent massive objects in memory (managed in R but implemented in C) and supporting the manipulation and exploration of these objects. It provides a framework for the development of C code for interactive use with R. In Unix environments, it supports the use of shared memory, allowing different R sessions (threads) on the same machine to share access to the same `BigMatrix`.

**Author(s)**

John W. Emerson and Michael J. Kane

Maintainer: Jay Emerson <john.emerson@yale.edu>

**See Also**

`BigMatrix`, `which.bm`, `colmean`, `biglm`

**Examples**

```r
# Our examples are all trivial in size, rather than burning huge amounts
# of memory simply to demonstrate the package functionality.

x <- BigMatrix(5, 2, type="integer", init=0, colnames=c("alpha", "beta"))
x x[,] x[,1] <- 1:5 x[,] mean(x) colmean(x)
```

---

`colmean, colmin, colmax, colrange, colvar, colsd`

*Basic statistics for “BigMatrix” objects.*

**Description**

These functions are implemented for each column of the `BigMatrix`. 
Usage

```r
colmean(x, cols, na.rm)
colmin(x, ..., na.rm)
colmax(x, ..., na.rm)
colrange(x, ..., na.rm)
colvar(x, cols, na.rm)
colsd(x, cols, na.rm)
```

Arguments

- `x`: a `BigMatrix`.
- `cols`: a scalar or vector of column(s) to be summarized.
- `na.rm`: if `TRUE`, remove `NA` values before summarizing.
- `...`: additional arguments.

Details

These functions essentially apply summary functions to each column (or each specified column) of the `BigMatrix` in turn.

The `cols` argument works for each of these, although for some of the functions the argument works via `...` instead of being an explicit argument. This is for consistency with the associated functions in base.

Value

For `colrange`, a matrix with two columns and `length(cols)` rows; column 1 contains the minimum, and column 2 contains the maximum for that column. The other functions return vectors of length `length(cols)`.

Author(s)

John W. Emerson and Michael J. Kane

See Also

`BigMatrix`

Examples

```r
x <- as.BigMatrix(matrix(sample(1:10, 20, replace=TRUE), 5, 4))
x[,]
mean(x)
colmean(x)
colmin(x)
colmax(x)
colsd(x)
colrange(x)
```
deepcopy.bm  
*Produces a physical copy of a “BigMatrix”*

Description

This is needed to make a duplicate of a `BigMatrix`, because traditional R syntax would only copy the R object (the pointer to the `BigMatrix` rather than the `BigMatrix` itself).

Usage

```r
deepcopy.bm(x, shared = FALSE)
```

Arguments

- `x` a `BigMatrix`.
- `shared` if TRUE, make the new copy a shared memory object.

Details

This is needed to make a duplicate of a `BigMatrix`, because traditional R syntax would only copy the R object (the pointer to the `BigMatrix` rather than the `BigMatrix` itself).

Value

a `BigMatrix`, possibly in shared memory.

Author(s)

John W. Emerson and Michael J. Kane

See Also

- `BigMatrix`

Examples

```r
x <- as.BigMatrix(matrix(1:30, 10, 3))
y <- deepcopy.bm(x)
x
y
head(x)
head(y)
```
Description

Create a hash into a \texttt{BigMatrix} based on the values of the specified column.

Usage

\texttt{hash.mat.bm(x, col)}

Arguments

\begin{itemize}
\item \texttt{x} \hspace{1cm} a \texttt{BigMatrix} sorted by column \texttt{col}.
\item \texttt{col} \hspace{1cm} an integer or name of the target column; \texttt{BigMatrix} \texttt{x} must be sorted on this column.
\end{itemize}

Details

When a column of a \texttt{BigMatrix} contains many duplicated values, it can be useful (and efficient) to access subsets of the matrix using a hash table. To do this, the matrix must first be sorted based on the entries in the desired column, and the code is designed for integer-valued \texttt{BigMatrix} objects. Ideally, the values in the specified column should range from 1 to some maximum value that is considerably less than \texttt{nrow(x)}.

Value

a two-column matrix, where the values in row \texttt{i} provide the range of indices of \texttt{x} containing the value \texttt{i} in the specified column, \texttt{col}.

Author(s)

John W. Emerson and Michael J. Kane

See Also

\texttt{BigMatrix}

Examples

\begin{verbatim}
x <- as.BigMatrix(matrix(sample(1:10, 100, replace=TRUE), 25, 4))
theorder <- order(x[,1])
for (i in 1:ncol(x)) x[,i] <- x[theorder,i]
thehash <- hash.mat.bm(x, 1)
x[,]
thehash

# The following will produce all rows with entries 5 or 9 in the first column:
x[c(thehash[5,1]:thehash[5,2], thehash[9,1]:thehash[9,2]),]
\end{verbatim}
is.shared  

Check the shared memory status of a “BigMatrix”

Description

Check to see if a BigMatrix is in shared memory.

Usage

is.shared(x)

Arguments

x  
a BigMatrix.

Details

None.

Value

TRUE if a BigMatrix is in shared memory, and FALSE otherwise.

Author(s)

John W. Emerson and Michael J. Kane

See Also

bigmemoR, BigMatrix

Examples

x <- BigMatrix(10, 2, init=-5, colnames=c("alpha", "beta"))
is.BigMatrix(x)
head(x)
y <- as.BigMatrix(matrix(1:20, 10, 2))
y[,]
cbind(x[,1], y[,2])
y[,1] <- x[,2]
head(y)
# New examples:
# z <- BigSharedMatrix(7, 3, type='integer', init=3, filename='test.txt')
# z[,]
# dim(z)
# z[1,1] <- 2
# z[,]
# y <- AttachBigSharedMatrix('test.txt')
# y[,
which.bm

Provides “which”-like functionality for a “BigMatrix”

Description

Implements which-like functionality for a BigMatrix, with additional options for efficient comparisons executed in C rather than R.

Usage

which.bm(x, cols, vals, comps, op = 'AND')

Arguments

x a BigMatrix.
cols a vector of column indices or names.
vals a list (one component for each of cols) of vectors of length 1 or 2; length 1 is used to test equality (or not equal), while vectors of length 2 are used for checking values in the range (-Inf and Inf are allowed). If a scalar or vector of length 2 instead of a list, it will be replicated length(cols) times.
comps a list of operators, including 'eq', 'neq', 'le', 'lt', 'ge' and 'gt'. If a single operator, it will be replicated length(testCol) times.
op the comparison operator for combining the results of the individual tests, either 'AND' or 'OR'.

Details

To avoid the creation of massive vectors in R when doing comparisons, which.bm executes column-by-column comparisons of values to the specified values or ranges, and then returns the row indices satisfying the comparison using the op operator. More advanced comparisons are then possible (and memory-efficient) in R by doing set operations (union and intersect, for example) on the results of multiple which.bm calls.

Note that NA is a valid argument in conjunction with 'eq' or 'neq', replacing traditional is.na() calls. And both -Inf and Inf can be used for one sided inequalities.

Value

a vector of row indices satisfying the criteria.

Author(s)

John W. Emerson and Michael J. Kane
See Also

BigMatrix, which

Examples

x <- as.BigMatrix(matrix(1:30, 10, 3))
x[,,]
x[which.bm(x, 1:2, list(c(2,3), c(11,17)),
     list(c('ge','le'), c('gt', 'lt')), 'OR'),]

x[which.bm(x, 1:2, list(c(2,3), c(11,17)),
     list(c('ge','le'), c('gt', 'lt')), 'AND'),]

x[1,1] <- NA
which.bm(x, 1:2, NA, 'eq', 'OR')
which.bm(x, 1:2, NA, 'neq', 'AND')

# Column 1 equal to 4 and/or column 2 less than or equal to 16:
which.bm(x, 1:2, list(4, 16), list('eq', 'le'), 'OR')
which.bm(x, 1:2, list(4, 16), list('eq', 'le'), 'AND')

# Column 2 less than or equal to 15:
which.bm(x, 2, 15, 'le')

# No NAs in either column, and column 2 strictly less than 15:
which.bm(x, c(1:2,2), list(NA, NA, 15), list('neq', 'neq', 'lt'), 'AND')

write.bm, read.bm File interface for a "BigMatrix"

Description

Create a BigMatrix by reading from a suitably-formatted ASCII file, or write the contents of a BigMatrix object to a file.

Usage

write.bm(bigMat, fileName = NA, row.names = FALSE, col.names = FALSE)
read.bm(fileName, sep = ',', header = FALSE, row.names = NULL, col.names = NULL,
        type = NA, skip = 0, shared = FALSE)

Arguments

bigMat a BigMatrix object.
fileName the name of an input/output file.
sep a field delimiter.
header if TRUE, the first line (after a possible skip) should contain column names.
row.names if TRUE, use the first column of the file for row names; if a vector of names, use them even if row names appear to exist in the file.
col.names if TRUE, use the first row of the file for column names; if a vector of names, use them even if column names exist in the file.
type preferably specified, 'integer' for example.
skip number of lines to skip at the head of the file.
shared if TRUE, load the object into shared memory.

Details

Files contain only one atomic type (all integer, for example). Once we implement BigDataFrame, this assumption will be relaxed.

When reading from a file, if type is not specified we try to make a reasonable guess for you without making any guarantees at this point. The same is true for the field separator. Warning messages will be printed to alert you of this. Unless you have really large integer values, we strongly recommend you consider 'short'. If you have something that is essentially categorical, you might even be able use 'char', with huge memory savings in large data sets.

Value

a BigMatrix object is returned by read.bm, while write.bm creates an output file in the present working directory.

Author(s)

John W. Emerson and Michael J. Kane

See Also

BigMatrix

Examples

# Without specifying the type, this BigMatrix x will hold integers.
x <- as.BigMatrix(matrix(1:10, 5, 2))
x[2,2] <- NA
x[,] <- NA
write.bm(x, "foo.txt")

# Just for fun, I'll read it back in as character (1-byte integers):
y <- read.bm("foo.txt", type="char")
y[,]