Package 'matter'

October 18, 2017

Type Package

Title A framework for rapid prototyping with binary data on disk

Version 1.2.0

Date 2016-10-11

Author Kylie A. Bemis <k.bemis@northeastern.edu>

Maintainer Kylie A. Bemis <k.bemis@northeastern.edu>

Description Memory-efficient reading, writing, and manipulation of structured binary data on disk as vectors, matrices, and arrays.

License Artistic-2.0

Depends methods, stats, biglm

Imports BiocGenerics, irlba, utils

Suggests BiocStyle, testthat

Collate matterGenerics.R utils.R drle.R atoms.R matter.R matter_vec.R matter_mat.R stats.R apply.R bigglm.R prcomp.R

biocViews Software, Infrastructure

URL https://github.com/kuwisdelu/matter

NeedsCompilation yes

R topics documented:

apply *Apply Functions Over "matter" Matrices*

Description

An implementation of apply for matter_mat matrices.

Usage

```
## S4 method for signature 'matter_mat'
apply(X, MARGIN, FUN, ...)
```

Arguments

X	A matter_mat object.
MARGIN	Must be 1 or 2 for matter_mat matrices, where '1' indicates rows and '2' indicates columns. The dimension names can also be used if X has dimnames set.
FUN	The function to be applied.
...	Additional arguments to be passed to FUN.

Details

Because FUN must be executed by the interpreter in the appropriate R environment, the full row or column will be loaded into memory. The chunksize of X is ignored. For summary statistics, functions like colMeans,matter_mat-method and rowMeans,matter_mat-method offer greater control over memory pressure.

Value

See apply for details.

Author(s)

Kylie A. Bemis

See Also

apply

Examples

```
x <- matter(1:100, nrow=10, ncol=10)

apply(x, 2, summary)
```

bigglm *Using "biglm" with "matter"*

Description

This method allows matter_mat matrices to be used with the bigglm function from the "biglm" package.

Usage

```
## S4 method for signature 'formula,matter_mat'
bigglm(formula, data, ..., chunksize = NULL, fc = NULL)
```

Arguments

formula	A model formula.
data	A matter matrix with column names.
chunksize	An integer giving the maximum number of rows to process at a time. If left NULL, this will be calculated by dividing the chunksize of data by the number of variables in the formula.
fc	Either column indices or names of variables which are factors.
...	Additional options passed to bigglm.

Value

An object of class bigglm.

Author(s)

Kylie A. Bemis

See Also

bigglm

Examples

```
set.seed(1)

x <- matter_mat(rnorm(1000), nrow=100, ncol=10)

colnames(x) <- c(paste0("x", 1:9), "y")

fm <- paste0("y ~ ", paste0(paste0("x", 1:9), collapse=" + "))
fm <- as.formula(fm)

fit <- bigglm(fm, data=x, chunksize=50)
coef(fit)
```

delayed-ops *Delayed Operations on "matter" Objects*

Description

Some arithmetic operations are available as delayed operations on `matter` objects. With these operations, no data is changed on disk, and the operation is only executed when elements of the object are actually accessed.

Details

Currently the following operations are supported:

'Arith': '+', '-', '*', '/', '^'

'Math': 'exp', 'log', 'log2', 'log10'

Delayed operations are applied at the C++ layer immediately after the elements are read from disk. This means that operations that are implemented in C and/or C++ for efficiency (such as summary statistics) will also reflect the execution of the delayed operations.

Value

A new `matter` object with the registered delayed operation. Data on disk is not modified; only object metadata is changed.

Author(s)

Kylie A. Bemis

See Also

`Arith`, `Math`

Examples

```
x <- matter(1:100, length=100)
y <- x + 1

x[1:10]
y[1:10]

mean(x)
mean(y)
```

drle-class *Delta Run Length Encoding*

Description

The drle class stores delta-run-length-encoded vectors. These differ from other run-length-encoded vectors provided by other packages in that they allow for runs of values that each differ by a common difference (delta).

Usage

```
## Instance creation
drle(x, cr_threshold = 0)

is.drle(x)
## Additional methods documented below
```

Arguments

x	An integer or numeric vector to convert to delta run length encoding for drle(); an object to test if it is of class drle for is.drle().
cr_threshold	The compression ratio threshold to use when converting a vector to delta run length encoding. The default (0) always converts the object to drle. Values of cr_threshold < 1 correspond to compressing even when the output will be larger than the input (by a certain ratio). For values > 1, compression will only take place when the output is (approximately) at least cr_threshold times smaller.

Value

An object of class drle.

Slots

values: The values that begin each run.

lengths: The length of each run.

deltas: The difference between the values of each run.

Creating Objects

drle instances can be created through drle().

Methods

Standard generic methods:

x[i]: Get or set elements of the uncompressed vector.

length(x): Get the length of the uncompressed vector.

c(x, ...): Combine vectors.

Author(s)

Kylie A. Bemis

See Also

[base]{rle}

Examples

```
## Create a drle vector
x <- c(1,1,1,1,1,6,7,8,9,10,21,32,33,34,15)
y <- drle(x)

# Check that their elements are equal
x == y[]
```

---

matter-class *Vectors, Matrices, and Arrays Stored on Disk*

Description

The matter class and its subclasses are designed for easy on-demand read/write access to binary on-disk data structures, and working with them as vectors, matrices, and arrays.

Usage

```
## Instance creation
matter(...)

## Additional methods documented below
```

Arguments

...	Arguments passed to subclasses.

Value

An object of class matter.

Slots

data: This slot stores the information about locations of the data on disk and within the files.

datamode: The storage mode of the accessed data when read into R. This should a 'character' vector of length one with value 'integer' or 'numeric'.

paths: A 'character' vector of the paths to the files where the data are stored.

filemode: The read/write mode of the files where the data are stored. This should be 'rb' for read-only access, or 'rb+' for read/write access.

chunksize: The maximum number of elements which should be loaded into memory at once. Used by methods implementing summary statistics and linear algebra. Ignored when explicitly subsetting the dataset.

`length`: The length of the data.

`dim`: Either 'NULL' for vectors, or an integer vector of length one of more giving the maximal indices in each dimension for matrices and arrays.

`names`: The names of the data elements for vectors.

`dimnames`: Either 'NULL' or the names for the dimensions. If not 'NULL', then this should be a list of character vectors of the length given by 'dim' for each dimension. This is always 'NULL' for vectors.

`ops`: Delayed operations to be applied on atoms.

Creating Objects

`matter` is a virtual class and cannot be instantiated directly, but instances of its subclasses can be created through `matter()`.

Methods

Class-specific methods:

`atomdata(x)`: Access the 'data' slot.

`adata(x)`: An alias for atomdata(x).

`datamode(x)`, `datamode(x) <- value`: Get or set 'datamode'.

`paths(x)`, `paths(x) <- value`: Get or set 'paths'.

`filemode(x)`, `filemode(x) <- value`: Get or set 'filemode'.

`chunksize(x)`, `chunksize(x) <- value`: Get or set 'filemode'.

Standard generic methods:

`length(x)`, `length(x) <- value`: Get or set 'length'.

`dim(x)`, `dim(x) <- value`: Get or set 'dim'.

`names(x)`, `names(x) <- value`: Get or set 'names'.

`dimnames(x)`, `dimnames(x) <- value`: Get or set 'dimnames'.

Author(s)

Kylie A. Bemis

See Also

`matter_vec`, `matter_mat`

Examples

```
## Create a matter_vec vector
x <- matter(1:100, length=100)
x[]

## Create a matter_mat matrix
x <- matter(1:100, nrow=10, ncol=10)
x[]
```

matter_ex-data *Examples for "matter" package*

Description

Example data for the "matter" package for use in vignettes.

Usage

```
data(matter_ex)
```

Value

None. Loads objects required to build vignettes.

matter_mat-class *Matrices Stored on Disk*

Description

The matter_mat class implements on-disk matrices.

Usage

```
## Instance creation
matter_mat(data, datamode = "double", paths = NULL,
            filemode = ifelse(is.null(paths), "rb+", "rb"),
            offset = c(0, cumsum(sizeof(datamode) * extent)[-length(extent)]),
            extent = if (rowMaj) rep(ncol, nrow) else rep(nrow, ncol),
            nrow = 0, ncol = 0, rowMaj = FALSE, dimnames = NULL, ...)

## Additional methods documented below
```

Arguments

data	An optional data vector which will be initially written to the data on disk if provided.
datamode	A 'character' vector giving the storage mode of the data on disk. Allowable values are 'short', 'int', 'long', 'float', and 'double'.
paths	A 'character' vector of the paths to the files where the data are stored. If 'NULL', then a temporary file is created using tempfile.
filemode	The read/write mode of the files where the data are stored. This should be 'rb' for read-only access, or 'rb+' for read/write access.
offset	A vector giving the offsets in number of bytes from the beginning of each file in 'paths', specifying the start of the data to be accessed for each file.
extent	A vector giving the length of the data for each file in 'paths', specifying the number of elements of size 'datamode' to be accessed from each file.
nrow	An optional number giving the total number of rows.

ncol	An optional number giving the total number of columns.
rowMaj	Whether the data should be stored in row-major order (as opposed to column-major order) on disk. Defaults to 'FALSE', for efficient access to columns. Set to 'TRUE' for more efficient access to rows instead.
dimnames	The names of the matrix dimensions.
...	Additional arguments to be passed to constructor.

Value

An object of class matter_mat.

Slots

data: This slot stores the information about locations of the data on disk and within the files.

datamode: The storage mode of the accessed data when read into R. This should a 'character' vector of length one with value 'integer' or 'numeric'.

paths: A 'character' vector of the paths to the files where the data are stored.

filemode: The read/write mode of the files where the data are stored. This should be 'rb' for read-only access, or 'rb+' for read/write access.

chunksize: The maximum number of elements which should be loaded into memory at once. Used by methods implementing summary statistics and linear algebra. Ignored when explicitly subsetting the dataset.

length: The length of the data.

dim: Either 'NULL' for vectors, or an integer vector of length one of more giving the maximal indices in each dimension for matrices and arrays.

names: The names of the data elements for vectors.

dimnames: Either 'NULL' or the names for the dimensions. If not 'NULL', then this should be a list of character vectors of the length given by 'dim' for each dimension. This is always 'NULL' for vectors.

Extends

matter

Creating Objects

matter_mat instances can be created through matter_mat() or matter().

Methods

Standard generic methods:

x[i,j], x[i,j] <- value: Get or set the elements of the matrix.

x %*% y: Matrix multiplication. At least one matrix must be an in-memory R matrix (or vector).

crossprod(x, y): Alias for t(x) %*% y.

tcrossprod(x, y): Alias for x %*% t(y).

cbind(x, ...), rbind(x, ...): Combine matrices by row or column.

t(x): Transpose a matrix. This is a quick operation which only changes metadata and does not touch the on-disk data.

Author(s)

Kylie A. Bemis

See Also

matter

Examples

```
x <- matter_mat(1:100, nrow=10, ncol=10)
x[]
```

---

matter_vec-class *Vectors Stored on Disk*

Description

The matter_vec class implements on-disk vectors.

Usage

```
## Instance creation
matter_vec(data, datamode = "double", paths = NULL,
           filemode = ifelse(is.null(paths), "rb+", "rb"),
           offset = 0, extent = length, length = 0L, names = NULL, ...)

## Additional methods documented below
```

Arguments

data	An optional data vector which will be initially written to the data on disk if provided.
datamode	A 'character' vector giving the storage mode of the data on disk. Allowable values are 'short', 'int', 'long', 'float', and 'double'.
paths	A 'character' vector of the paths to the files where the data are stored. If 'NULL', then a temporary file is created using tempfile.
filemode	The read/write mode of the files where the data are stored. This should be 'rb' for read-only access, or 'rb+' for read/write access.
offset	A vector giving the offsets in number of bytes from the beginning of each file in 'paths', specifying the start of the data to be accessed for each file.
extent	A vector giving the length of the data for each file in 'paths', specifying the number of elements of size 'datamode' to be accessed from each file.
length	An optional number giving the total length of the data across all files, equal to the sum of 'extent'. This is ignored and calculated automatically if 'extent' is specified.
names	The names of the data elements.
...	Additional arguments to be passed to constructor.

Value

An object of class matter_vec.

Slots

`data`: This slot stores the information about locations of the data on disk and within the files.

`datamode`: The storage mode of the *accessed* data when read into R. This is a 'character' vector of length one with value 'integer' or 'numeric'.

`paths`: A 'character' vector of the paths to the files where the data are stored.

`filemode`: The read/write mode of the files where the data are stored. This should be 'rb' for read-only access, or 'rb+' for read/write access.

`chunksize`: The maximum number of elements which should be loaded into memory at once. Used by methods implementing summary statistics and linear algebra. Ignored when explicitly subsetting the dataset.

`length`: The length of the data.

`dim`: Either 'NULL' for vectors, or an integer vector of length one of more giving the maximal indices in each dimension for matrices and arrays.

`names`: The names of the data elements for vectors.

`dimnames`: Either 'NULL' or the names for the dimensions. If not 'NULL', then this should be a list of character vectors of the length given by 'dim' for each dimension. This is always 'NULL' for vectors.

Extends

matter

Creating Objects

`matter_vec` instances can be created through `matter_vec()` or `matter()`.

Methods

Standard generic methods:

`x[i], x[i] <- value`: Get or set the elements of the vector.

`c(x, ...)`: Combine vectors.

`t(x)`: Transpose a vector (to a row matrix). This is a quick operation which only changes metadata and does not touch the on-disk data.

Author(s)

Kylie A. Bemis

See Also

matter

Examples

```
x <- matter_vec(1:100, length=100)
x[]
```

prcomp *Principal Components Analysis for "matter" Matrices*

Description

This method allows computation of a truncated principal components analysis of a matter_mat matrix using the implicitly restarted Lanczos method irlba.

Usage

```
## S4 method for signature 'matter_mat'
prcomp(x, n = 3, retx = TRUE, center = TRUE, scale. = FALSE, ...)
```

Arguments

x	A matter matrix.
n	The number of principal componenets to return, must be less than min(dim(x)).
retx	A logical value indicating whether the rotated variables should be returned.
center	A logical value indicating whether the variables should be shifted to be zero-centered, or a centering vector of length equal to the number of columns of x. The centering is performed implicitly and does not change the data-on-disk in x.
scale.	A logical value indicating whether the variables should be scaled to have unit variance, or a scaling vector of length equal to the number of columns of x. The scaling is performed implicitly and does not change the data-on-disk in x.
...	Additional options passed to irlba.

Value

An object of class 'prcomp'. See ?prcomp for details.

Note

The 'tol' truncation argument found in the default prcomp method is not supported. In place of the truncation tolerance in the original function, the argument n explicitly gives the number of principal components to return. A warning is generated if the argument 'tol' is used.

Author(s)

Kylie A. Bemis

See Also

bigglm

Examples

```
set.seed(1)

x <- matter_mat(rnorm(1000), nrow=100, ncol=10)

prcomp(x)
```

scale *Scaling and Centering of "matter" Matrices*

Description

An implementation of scale for matter_mat matrices.

Usage

```
## S4 method for signature 'matter_mat'
scale(x, center = TRUE, scale = TRUE)
```

Arguments

x	A matter_mat object.
center	Either a logical value or a numeric vector of length equal to the number of columns of 'x'.
scale	Either a logical value or a numeric vector of length equal to the number of columns of 'x'.

Details

See scale for details.

Value

A matter_mat object with the appropriate 'scaled:center' and 'scaled:scale' attributes set. No data on disk is changed, but the scaling will be applied any time the data is read. This includes but is not limited to loading data elements via subsetting, summary statistics methods, and matrix multiplication.

Author(s)

Kylie A. Bemis

See Also

scale

Examples

```
x <- matter(1:100, nrow=10, ncol=10)

scale(x)
```

summary-stats *Summary Statistics for "matter" Objects*

Description

These functions efficiently calculate summary statistics for `matter` objects. For matrices, they operate efficiently on both rows and columns.

Usage

```
## S4 method for signature 'matter'
mean(x, na.rm)
## S4 method for signature 'matter'
sum(x, na.rm)
## S4 method for signature 'matter'
sd(x, na.rm)
## S4 method for signature 'matter'
var(x, na.rm)
## S4 method for signature 'matter_mat'
colMeans(x, na.rm)
## S4 method for signature 'matter_mat'
colSums(x, na.rm)
## S4 method for signature 'matter_mat'
colSds(x, na.rm)
## S4 method for signature 'matter_mat'
colVars(x, na.rm)
## S4 method for signature 'matter_mat'
rowMeans(x, na.rm)
## S4 method for signature 'matter_mat'
rowSums(x, na.rm)
## S4 method for signature 'matter_mat'
rowSds(x, na.rm)
## S4 method for signature 'matter_mat'
rowVars(x, na.rm)
```

Arguments

`x`	A `matter` object.
`na.rm`	If `TRUE`, remove `NA` values before summarizing.

Details

These summary statistics methods operate on chunks of data (equal to the `chunksize` of `x`) which are loaded into memory and then freed before reading the next chunk.

For row and column summaries on matrices, the iteration scheme is dependent on the layout of the data. Column-major matrices will always be iterated over by column, and row-major matrices will always be iterated over by row. Row statistics on column-major matrices and column statistics on row-major matrices are calculated iteratively.

The efficiency of these methods is entirely dependent on the `chunksize` of `x`. Larger chunks will yield faster calculations, but greater memory usage. The row and column summary methods may be more or less efficient than the equivalent call to `apply`, depending on the chunk size.

Varsiance and standard deviation are calculated using a running sum of squares formula which can be calculated iteratively and is accurate for large floating-point datasets (see reference).

Value

For `mean`, `sum`, `sd`, and `var`, a single number. For the column summaries, a vector of length equal to the number of columns of the matrix. For the row summaries, a vector of length equal to the number of rows of the matrix.

Author(s)

Kylie A. Bemis

References

B. P. Welford, "Note on a Method for Calculating Corrected Sums of Squares and Products," Technometrics, vol. 4, no. 3, pp. 1-3, Aug. 1962.

See Also

`colSums`, `colMeans`, `rowSums`, `rowMeans`

Examples

```
x <- matter(1:100, nrow=10, ncol=10)

sum(x)
mean(x)
var(x)
sd(x)

colSums(x)
colMeans(x)
colVars(x)
colSds(x)

rowSums(x)
rowMeans(x)
rowVars(x)
rowSds(x)
```

Index