Package ‘biganalytics’

February 18, 2016

Version 1.1.14
Date 2016-02-17
Title Utilities for 'big.matrix' Objects from Package 'bigmemory'
Author John W. Emerson <jayemerson@gmail.com> and Michael J. Kane
   <kaneplusplus@gmail.com>
Maintainer Michael J. Kane <bigmemoryauthors@gmail.com>
Contact Jay and Mike <bigmemoryauthors@gmail.com>
Depends methods, stats, utils, bigmemory (>= 4.0.0), foreach, biglm
LinkingTo Rcpp, bigmemory, BH
Description Extend the 'bigmemory' package with various analytics.
   Functions 'bigkmeans' and 'binit' may also be used with native R objects.
   For 'tapply'-like functions, the bigtabulate package may also be helpful.
   For linear algebra support, see 'bigalgebra'. For mutex (locking) support
   for advanced shared-memory usage, see 'synchronicity'.
License LGPL-3 | Apache License 2.0
Copyright (C) 2013 John W. Emerson and Michael J. Kane
URL http://www.bigmemory.org
LazyLoad yes
Biarch yes
RoxygenNote 5.0.1
NeedsCompilation yes
Repository CRAN
Date/Publication 2016-02-18 06:51:28

R topics documented:

   biganalytics-package .......................................................... 2
   apply.big.matrix-method ..................................................... 3
   bigglm.big.matrix ............................................................. 3
biganalytics-package

Utilities for big.matrix objects of package bigmemory

Description

Extend the *bigmemory* package with various analytics. In addition to the more obvious summary statistics (see `colmean`, etc...), *biganalytics* offers `biglm.big.matrix`, `bigglm.big.matrix`, `bigkmeans`, `binit`, and `apply` for `big.matrix` objects. Some of the functions may be used with native \texttt{R} objects, as well, providing gains in speed and memory-efficiency.

Author(s)

Michael J. Kane and John W. Emerson.

Maintainers Michael J. Kane <bigmemoryauthors@gmail.com>

References

http://www.bigmemory.org

Examples

```r
library(bigmemory)

x <- big.matrix(5, 2, type="integer", init=0,
                 dimnames=list(NULL, c("alpha", "beta")))
x
x[,]
x[,1] <- 1:5
x[,]
mean(x)
colmean(x)
summary(x)
apply(x, 1, mean)
```
apply.big.matrix-method

Description

apply for big.matrix objects. Note that the performance may be degraded (compared to apply with regular R matrices) because of S4 overhead associated with extracting data from big.matrix objects. This sort of limitation is unavoidable and would be the case (or even worse) with other "custom" data structures. Of course, this would only be partially significant if you are applying over lengthy rows or columns.

Usage

## S4 method for signature 'big.matrix'
apply(X, MARGIN, FUN, ...)

Arguments

- **X**: a big.matrix object.
- **MARGIN**: the margin. May be only be 1 or 2, but otherwise conforming to what you would expect from apply().
- **FUN**: the function to apply.
- **...**: other parameters to pass to the FUN parameter.

Examples

library(bigmemory)
options(bigmemory.typecast.warning=FALSE)
x <- big.matrix(5, 2, type="integer", init=0,
dimnames=list(NULL, c("alpha", "beta")))
x[,] <- round(rnorm(10))
biganalytics::apply(x, 1, mean)

bigglm.big.matrix

Use Thomas Lumley's "biglm" package with a "big.matrix"

Description

This is a wrapper to Thomas Lumley’s biglm package, allowing it to be used with massive data stored in big.matrix objects.
Usage

bigglm.big.matrix(formula, data, chunksize = NULL, ..., fc = NULL,
getNextChunkFunc = NULL)

biglm.big.matrix(formula, data, chunksize = NULL, ..., fc = NULL,
getNextChunkFunc = NULL)

Arguments

formula a model formula.
data a big.matrix.
chunksize an integer maximum size of chunks of data to process iteratively.
fc either column indices or names of variables that are factors.
... options associated with the biglm
getNextChunkFunc a function which retrieves chunk data

Value

an object of class biglm

Examples

## Not run:
library(bigmemory)
x <- matrix(unlist(iris), ncol=5)
colnames(x) <- names(iris)
x <- as.big.matrix(x)
head(x)

silly.biglm <- bigglm.big.matrix(Sepal.Length ~ Sepal.Width + Species,
data=x, fc="Species")
summary(silly.biglm)

y <- data.frame(x[,1])
y$Species <- as.factor(y$Species)
head(y)

silly.lm <- lm(Sepal.Length ~ Sepal.Width + Species, data=y)
summary(silly.lm)

## End(Not run)
bigkmeans  Memory-efficient k-means cluster analysis

Description

k-means cluster analysis without the memory overhead, and possibly in parallel using shared memory.

Usage

bigkmeans(x, centers, iter.max = 10, nstart = 1, dist = "euclid")

Arguments

x  a big.matrix object.

centers  a scalar denoting the number of clusters, or for k clusters, a k by ncol(x) matrix.

iter.max  the maximum number of iterations.

nstart  number of random starts, to be done in parallel if there is a registered backend (see below).

dist  the distance function. Can be "euclid" or "cosine".

Details

The real benefit is the lack of memory overhead compared to the standard kmeans function. Part of the overhead from kmeans() stems from the way it looks for unique starting centers, and could be improved upon. The bigkmeans() function works on either regular R matrix objects, or on big.matrix objects. In either case, it requires no extra memory (beyond the data, other than recording the cluster memberships), whereas kmeans() makes at least two extra copies of the data. And kmeans() is even worse if multiple starts (nstart>1) are used. If nstart>1 and you are using bigkmeans() in parallel, a vector of cluster memberships will need to be stored for each worker, which could be memory-intensive for large data. This isn’t a problem if you use are running the multiple starts sequentially.

Unless you have a really big data set (where a single run of kmeans not only burns memory but takes more than a few seconds), use of parallel computing for multiple random starts is unlikely to be much faster than running iteratively.

Only the algorithm by MacQueen is used here.

Value

An object of class kmeans, just as produced by kmeans.
Note
A comment should be made about the excellent package `foreach`. By default, it provides `foreach`, which is used much like a `for` loop, here over the `nstart` and doing a final comparison of all results).

When a parallel backend has been registered (see packages `doSNOW`, `doMC`, and `doMPI`, for example), `bigkmeans()` automatically distributes the `nstart` random starting points across the available workers. This is done in shared memory on an SMP, but is distributed on a cluster *IF* the `big.matrix` is file-backed. If used on a cluster with an in-RAM `big.matrix`, it will fail horribly. We’re considering an extra option as an alternative to the current behavior.

---

### binit

*Count elements appearing in bins of one or two variables*

#### Description
Provides preliminary counting functionality to eventually support graphical exploration or as an alternative to `table`. Note the availability of `bigtabulate`.

#### Usage
```r
binit(x, cols, breaks = 10)
```

#### Arguments
- **x**: a `big.matrix` or a `matrix`.
- **cols**: a vector of column indices or names of length 1 or 2.
- **breaks**: a number of bins to span the range from the maximum to the minimum, or a vector (1-variable case) or list of two vectors (2-variable case) where each vector is a triplet of min, max, and number of bins.

#### Details
The user may specify the number of bins to be used, of equal widths, spanning the range of the data (the default is 10 bins). The user may also specify the range to be spanned along with the number of bins, in case a summary of a subrange of the data is desired. Either univariate or bivariate counting is supported.

The function uses left-closed intervals `[a,b)` except in the right-most bin, where the interval is entirely closed.

#### Value
A list containing (a) a vector (1-variable case) or a matrix (2-variable case) of counts of the numbers of cases appearing in each of the bins, (b) description(s) of bin centers, and (c) description(s) of breaks between the bins.
Examples

```r
y <- matrix(rnorm(40), 20, 2)
y[,1] <- NA
x <- as.big.matrix(y, type="double")
x[,]
  binit(y, 1:2, list(c(-1,1,5), c(-1,1,2)))
  binit(x, 1:2, list(c(-1,1,5), c(-1,1,2)))

  binit(y, 1:2)
  binit(x, 1:2)

  binit(y, 1:2, 5)
  binit(x, 1:2, 5)

  binit(y, 1)
  binit(x, 1)

x <- as.big.matrix(matrix(rnorm(400), 200, 2), type="double")
x[,1] <- x[,1] + 3
x.binit <- binit(x, 1:2)
  filled.contour(round(x.binit$rowcenters,2), round(x.binit$colcenters,2),
                  x.binit$counts, xlab="Variable 1",
                  ylab="Variable 2")
```

---

**colmin**

*Basic summary statistics for "big.matrix" objects*

**Description**

Functions operate on columns of a `big.matrix` object

**Usage**

```r
colmin(x, cols = NULL, na.rm = FALSE)
```
colprod(x, cols = NULL, na.rm = FALSE)

## S4 method for signature 'big.matrix'
colprod(x, cols = NULL, na.rm = FALSE)

## S4 method for signature 'big.matrix'
prod(x, ..., na.rm = FALSE)

colsum(x, cols = NULL, na.rm = FALSE)

## S4 method for signature 'big.matrix'
colsum(x, cols = NULL, na.rm = FALSE)

## S4 method for signature 'big.matrix'
sum(x, ..., na.rm = FALSE)

colrange(x, cols = NULL, na.rm = FALSE)

## S4 method for signature 'big.matrix'
colrange(x, cols = NULL, na.rm = FALSE)

## S4 method for signature 'big.matrix'
range(x, ..., na.rm = FALSE)

colmean(x, cols = NULL, na.rm = FALSE)

## S4 method for signature 'big.matrix'
colmean(x, cols = NULL, na.rm = FALSE)

## S4 method for signature 'big.matrix'
mean(x, ...)

colvar(x, cols = NULL, na.rm = FALSE)

## S4 method for signature 'big.matrix'
colvar(x, cols = NULL, na.rm = FALSE)

colsd(x, cols = NULL, na.rm = FALSE)

## S4 method for signature 'big.matrix'
colsd(x, cols = NULL, na.rm = FALSE)

colna(x, cols = NULL)

## S4 method for signature 'big.matrix'
colna(x, cols = NULL)
## S4 method for signature 'big.matrix'

`summary(object)`

### Arguments

- `x`: a `big.matrix` object.
- `cols`: a scalar or vector of column(s) to be summarized.
- `na.rm`: if `TRUE`, remove NA values before summarizing.
- `object`: a `big.matrix` object.
- `...`: options associated with the corresponding default R function.

### Details

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

### 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).

### Examples

```r
x <- as.big.matrix(
  matrix(sample(1:10, 20, replace=TRUE), 5, 4,
            dimnames=list(NULL, c("a", "b", "c", "d"))
  )
mean(x)
colmean(x)
colmin(x)
colmin(x, 1)
colmax(x)
colmax(x, "b")
colsum(x)
colrange(x)
range(x)
colprod(x)
```
Index

*Topic package
  biganalytics-package, 2

apply, 3
apply.big.matrix-method, 3

big.matrix, 3–7, 9
biganalytics (biganalytics-package), 2
biganalytics-package, 2
bigglm.big.matrix, 2, 3
bigkmeans, 2, 5
biglm, 3, 4
biglm.big.matrix, 2
biglm.big.matrix (bigglm.big.matrix), 3
binit, 2, 6

colmax (colmin), 7
colmax.big.matrix-method (colmin), 7
colmean, 2
colmean (colmin), 7
colmean.big.matrix-method (colmin), 7
colmin, 7
colmin.big.matrix-method (colmin), 7
colmin, min, colmax, max, colprod, prod, colsum, sum, colrange, range, colmean, mean, colvar, var, colsd, sd, colna, summary
  (colmin), 7
colna (colmin), 7
colna.big.matrix-method (colmin), 7
colprod (colmin), 7
colprod.big.matrix-method (colmin), 7
colrange (colmin), 7
colrange.big.matrix-method (colmin), 7
colvar (colmin), 7
colvar.big.matrix-method (colmin), 7

to application, 3

kmeans, 5

matrix, 6
max.big.matrix-method (colmin), 7
mean.big.matrix-method (colmin), 7
min.big.matrix-method (colmin), 7
prod.big.matrix-method (colmin), 7
range.big.matrix-method (colmin), 7
sum.big.matrix-method (colmin), 7
summary.big.matrix-method (colmin), 7

foreach, 6
formula, 4