Package 'biganalytics'

July 2, 2014

Version 1.1.1

Date 2012-09-20

Title A library of utilities for big.matrix objects of 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)

LinkingTo BH, bigmemory

Suggests foreach, biglm

Description This package extends 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

OS_type unix

URL http://www.bigmemory.org

LazyLoad yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2013-04-01 16:17:04

R topics documented:

`biganalytics-package` *biganalytics: A library of utilities for big.matrix objects of package bigmemory. Some work with R matrices, independently of bigmemory.*

Description

This package extends 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 R objects, as well, providing gains in speed and memory-efficiency.

Details

Package:	biganalytics
Type:	Package
Version:	1.1.1
Date:	2012-09-20
License:	LGPL-3
Copyright:	(C) 2013 John W. Emerson and Michael J. Kane
URL:	http://www.bigmemory.org
LazyLoad:	yes

The **bigmemory** package contains the core `big.matrix` support; **biganalytics** contains tools for exploratory data analysis as well as more advanced analytics on `big.matrix` objects. Sister packages **synchronicity**, **bigtabulate**, and **bigalgebra** provide additional functionality.

Author(s)

John W. Emerson and Michael J. Kane

Maintainer: Michael J. Kane <bigmemoryauthors@gmail.com>

References

The Bigmemory Project: `http://www.bigmemory.org/`.

See Also

For example, see big.matrix, biglm, bigkmeans, binit, colmean.

Examples

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

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-methods *apply() for big.matrix objects.*

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 partically significant if you are applying over lengthy rows or columns.

Methods

apply signature(x = "big.matrix"): apply() where MARGIN may only be 1 or 2, but otherwise conforming to what you would expect from apply().

Examples

```
library(bigmemory)

x <- big.matrix(5, 2, type="integer", init=0,
                dimnames=list(NULL, c("alpha", "beta")))
x[,] <- round(rnorm(10))
apply(x, 1, mean)
```

bigkmeans *The Bigmemory Project's 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)
```

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).

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` random starting points. Even so, there are efficiencies, doing a comparison of each result to the previous best result (rather than saving everything 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.

Author(s)

John W. Emerson <bigmemoryauthors.@gmail.com>

References

Hartigan, J. A. and Wong, M. A. (1979). A K-means clustering algorithm. *Applied Statistics* **28**, 100–108.

MacQueen, J. (1967) Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, eds L. M. Le Cam & J. Neyman, **1**, pp. 281–297. Berkeley, CA: University of California Press.

See Also

`big.matrix`, `foreach`

Examples

```
# Simple example (with one processor):

library(bigmemory)

x <- big.matrix(100000, 3, init=0, type="double")
x[seq(1,100000,by=2),] <- rnorm(150000)
x[seq(2,100000,by=2),] <- rnorm(150000, 5, 1)
head(x)
ans <- bigkmeans(x, 1)              # One cluster isn't always allowed
                                    # but is convenient.
ans$centers
ans$withinss
ans$size
apply(x, 2, mean)
ans <- bigkmeans(x, 2, nstart=5)    # Sequential multiple starts.
class(ans)
names(ans)
ans$centers
ans$withinss
ans$size
```

```
  # To use a parallel backend, try something like the following,
  # assuming you have at least 3 cores available on this machine.
  # Each processor does incur memory overhead for the storage of
  # cluster memberships.
  ## Not run:
    library(doSNOW)
    cl <- makeCluster(3, type="SOCK")
    registerDoSNOW(cl)
    ans <- bigkmeans(x, 2, nstart=5)

## End(Not run)

  # Both the following are run iteratively, but with less memory overhead
  # using bigkmeans().  Note that the gc() comparisons aren't completely
  # fair, because the big.matrix objects aren't reflected in the gc()
  # summary.  But the savings is there.
  gc(reset=TRUE)
  time.new <- system.time(print(bigkmeans(x, 2, nstart=5)$centers))
  gc()
  y <- x[,]
  rm(x)
  gc(reset=TRUE)
  time.old <- system.time(print(kmeans(y, 2, nstart=5)$centers))
  gc()
  # The new kmeans() centers should match the old kmeans() centers, without
  # the memory overhead amd running more quickly.
  time.new
  time.old
```

`biglm.big.matrix, 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

```
biglm.big.matrix( formula, data, chunksize=NULL, ..., fc=NULL,
  getNextChunkFunc=NULL)
bigglm.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 or bigglm functions
getNextChunkFunc	a function which retrieves chunk data

Details

See **biglm** package for more information; chunksize defaults to max(floor(nrow(data)/ncol(data)^2), 10000).

Value

an object of class biglm.

References

Algorithm AS274 Applied Statistics (1992) Vol. 41, No.2

Thomas Lumley (2005). biglm: bounded memory linear and generalized linear models. R package version 0.4.

See Also

biglm, big.matrix

Examples

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

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

silly.biglm <- biglm.big.matrix(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)

## End(Not run)
```

`binit` *Count elements appearing in bins of one or two variables.*

Description

This provides preliminary counting functionality to eventually support graphical exploration or as an alternative to `table`. Note the availability of **bigtabulate**.

Usage

```
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.

Author(s)

John W. Emerson and Michael J. Kane

See Also

`big.matrix`

Examples

```
y <- matrix(rnorm(40), 20, 2)
y[1,1] <- 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")
```

---

colmean, colmin, min, colrange, colvar, colsd, colprod, colsum, colna, etc...

Basic statistics for "big.matrix" objects.

Description

These functions operate on columns of a `big.matrix` object.

Usage

```
colmean(x, cols, na.rm)
colmin(x, cols, na.rm)
min(x, ..., na.rm)
colmax(x, cols, na.rm)
max(x, ..., na.rm)
colrange(x, cols, na.rm)
range(x, ..., na.rm)
colvar(x, cols, na.rm)
colsd(x, cols, na.rm)
colsum(x, cols, na.rm)
sum(x, ..., na.rm)
colprod(x, cols, na.rm)
prod(x, ..., na.rm)
colna(x, cols)
```

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.
...	options associated with the correspoding 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).

Author(s)

John W. Emerson and Michael J. Kane

See Also

bigmemory

Examples

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

Index