

An Introduction to the bigmemoryExtras Package

Peter M. Haverty

April 10, 2013

1 Introduction

This package defines a "BigMatrix" ReferenceClass which adds safety and convenience features to the file-backed `big.matrix` class from the `bigmemory` package. `BigMatrix` protects against segfaults by monitoring and gracefully restoring the connection to on-disk data. We provide utilities for using `BigMatrix`-derived classes as `assayData` matrices within the `Biobase` package's `eSet` family of classes. `BigMatrix` provides some optimizations related to attaching to, and indexing into, file-backed matrices with `dimnames`. Additionally, the package provides a "BigMatrixFactor" class, a file-backed matrix with factor properties.

```
> library(bigmemoryExtras)
> data.file = file.path(tempdir(),"bigmat","ds")
> x = matrix(1:9,ncol=3,dimnames=list(letters[1:3],LETTERS[1:3]))
> ds = BigMatrix(x,data.file)
> ds[,1] = 3:1
> ds[,1]
```

```
a b c
3 2 1
```

2 Re-attaching to on-disk data as necessary

When a `big.matrix` object is attached to its on-disk data, an external pointer is used to connect the R object to a C++ data structure. When a `big.matrix` object is not attached, like when it is loaded from an `RData` file, this pointer is `nil`. Any access to this `nil` pointer will crash R. The `bigmemoryExtras` package provides a `BigMatrix` class that prevents such a crash by controlling access to the external pointer. Additionally, `BigMatrix` objects remember the location of their on-disk components and automatically re-attach themselves as necessary.

This kind of thing would be helpful if you, for example, chose to save your new `BigMatrix` object to disk for later use. You might save your object using R's built in `save` or `saveRDS` functions.

```
> ds$descpath

[1] "/private/tmp/RtmpTEnxIj/bigmat/ds.desc.rds"

> saveRDS(ds,file=file.path(tempdir(),"foo.rds"))
> new.ds = readRDS(file=file.path(tempdir(),"foo.rds"))
> new.ds[1:2,2:3]

  B C
a 4 7
b 5 8
```

3 Files

The bigmemory package creates two files for each object: one binary file for the matrix data (the “backingfile”) and one text file with meta-data about its matrix (the “descriptor” file). The functions `dget` and `dput` are used to access this meta-data. For large matrices with `dimnames`, this can take a while. `BigMatrix` replaces the descriptor file with a binary one that uses `readRDS` and `saveRDS`, which requires about 1/10 the time to read when it’s time to attach an object to the backingfile.

4 S4 Style Access

The `BigMatrix` class uses R’s Reference Class system. Any change to the matrix portion of the data has on-disk side effects, so it seems natural that any other changes to the object should have the same behavior. In order to give `BigMatrix` the same API as a base matrix or `big.matrix` class, certain S4-style methods are provided. ReferenceClass objects are relatively new to R and unfamiliar to many users, so you may want to review the `ReferenceClasses` help page.

```
> nrow(ds)
[1] 3
> ds$nrow()
[1] 3
> ncol(ds)
[1] 3
> ds$ncol()
[1] 3
> dim(ds)
[1] 3 3
> ds$dim()
[1] 3 3
> dimnames(ds)
[[1]]
[1] "a" "b" "c"
[[2]]
[1] "A" "B" "C"
> ds$dimnames()
[[1]]
[1] "a" "b" "c"
[[2]]
[1] "A" "B" "C"
```

```
> length(ds)

[1] 9

> ds$length()

[1] 9
```

5 BigMatrixFactor

The bigmemoryExtras package adds a “BigMatrixFactor” class to provide a means to store large matrices of characters. On the file system, these are stored as the C type char or int (8 or 32 bits), depending on the number of levels in the factor. Subsetting a BigMatrixFactor returns a base matrix that is also a factor. This provides a convenient way to convert between integer and character representations of the data.

```
> data.file = file.path(tempdir(),"bigmat","fs")
> x = matrix( c(rep(1,5),rep(2,4)) ,ncol=3,dimnames=list(letters[1:3],LETTERS[1:3]))
> fs = BigMatrixFactor(x,data.file,levels=c("AA","BB"))
> fs[,]

  A  B  C
a AA AA BB
b AA AA BB
c AA BB BB
Levels: AA BB

> as(fs,"factor")

  A  B  C
a AA AA BB
b AA AA BB
c AA BB BB
Levels: AA BB

> as(fs,"matrix")

  A B C
a 1 1 2
b 1 1 2
c 1 2 2

> fs$levels

[1] "AA" "BB"

> levels(fs)

[1] "AA" "BB"
```

However, you may want to set the S3 class of this matrix/factor to ‘matrix’ in some cases, like when you need to maintain its shape.

```
> x = as(fs,"factor")
> x
```

```

      A  B  C
a AA AA BB
b AA AA BB
c AA BB BB
Levels: AA BB

> x == "BB"

[1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE

> y = as(fs,"matrix")
> y

      A B C
a 1 1 2
b 1 1 2
c 1 2 2

> y == 2

      A      B      C
a FALSE FALSE TRUE
b FALSE FALSE TRUE
c FALSE  TRUE TRUE

> rowMeans( y == 2 )

      a      b      c
0.3333333 0.3333333 0.6666667

```

6 Use with Biobase and eSet-derived Classes

Either class can be used as an assayDataElement in the assayData slot of the familiar BioConductor eSet-derived classes. We provide utility functions to deal with relocated BigMatrix files and to attach all of an eSet's BigMatrix assayDataElements. Of course, you can also just let them attach themselves as necessary.

```

> library(Biobase)
> eset = ExpressionSet()
> assayDataElement(eset,"exprs") = ds
> exprs(eset)[1:2,2:3]

      B C
a 4 7
b 5 8

> new.dir = file.path(tempdir(),"newbigmat")
> dir.create(new.dir,showWarnings=FALSE)
> file.copy(ds$descpath, new.dir)

[1] TRUE

> file.copy(ds$datapath, new.dir)

[1] TRUE

```

```

> updateAssayDataElementPaths( assayData(eset), new.dir )
> assayDataElement(eset,"exprs")$descpath

[1] "/tmp/RtmpTEnxIj/newbigmat/ds.desc.rds"

> attachAssayDataElements( eset )
>

```

7 Optimal Usage

Using dimension names can slow down access to a `BigMatrix`/`big.matrix` object considerably. Indexing into a `big.matrix` by `dimname` is considerably slower indexing by integer. `BigMatrix` has an optimization to reduce this penalty.

Reading from a `BigMatrix`/`big.matrix` with `dimnames` is also slower, as transferring `dimnames` from `big.matrix`'s C++ representation to R and adding them to the numeric data from the matrix takes some time. Typically, this takes much more time than just reading the numeric data.

Future optimizations may remove these penalties without changing the API, but when speed is critical, you will want to avoid indexing by character values and you may want to avoid having `dimnames` altogether.