ISSUE 37 | OCTOBER 2010

SOCIETY OF ACTUARIES Technology Section

CompAct

ELECTRONIC NEWSLETTER

CONTENTS

QUICK LINKS

R CORNER–MEMORY MANAGEMENT[1]

By Steve Craighead

Editor's note: R Corner is a series by Steve Craighead introducing readers to the "R" language used for statistics and modeling of data. See footnote 4 at the end of the article for information on installing and learning more about R.

Even though I'm still looking for all of my marbles, I'm going to talk about how to manage memory within R.

R has been built to run in 32-bit and 64-bit environments. Most of the time, the limits to memory are related to how R is built, but memory in the Windows' 32-bit R depends on what operating system is being used.

Usually, you know you have a memory issue when you get an error message like this: "Cannot allocate vector of ... "

Old Technology

Traditionally, I have managed the memory in R by using the two embedded properties "memory.size" and "memory.limit."

You use the memory.size command to determine how much current memory is available and how much can be made available.

For instance, "memory.size()" returns the current memory limit in that workbook, while "memory.size(TRUE)" returns the maximum size that the operating system can supply.

Council

Links of Interest

Fiction Contest

Howard Callif, Editor

SOA Staff
Meg Weber, Staff Partner

Sue Martz,
Section Specialist

Sam Phillips, Staff Editor

For instance, on my 32-bit Windows machine I get the following:

```
> memory.size()
[1] 20.47
> memory.size(TRUE)
[1] 23.44
>
```

Now, "memory.limit()" returns the memory limit, while "memory.limit(nnnn)," where "nnnn" is the number of megabtyes requested as a new limit. This is limited to 4096 Mb on 32-bit R builds.

On my machine, I get the following results:

```
>memory.limit()
[1] 1535
> memory.limit(3000)
[1] 3000
> memory.limit()
[1] 3000
```

So, now my memory limit has been increased to 3-Gig. When the memory.limit exceeds the max memory.size, R will still run, but the extra memory will be paged off and onto your hard drive. This means that you can use a large dataset, but processing time will go up dramatically since your hard drive doesn't run as fast as your computer's memory.

If you do use Windows, you may find that you can increase the operating system limit from 2Gig to 3Gig by using the "/3GB" switch in your "Boot.Ini" file. Please visit this Web address for more information.

I have used all of the above methods to great success. I was able to load million-record files into R memory through these techniques. If you review my earlier article on how to get data into R, you can also use the RODBC package to create SQL queries against large databases to extract what you need for your study.

New Technology–bigmemory[2]

Michael J. Kane and John W. Emerson developed a new matrix class called "big.matrix." The packages bigmemory, biganalytics,

synchronicity, bigalgebra, biglm and bigtabulate use this new class and allow you to manipulate larger and larger datasets within R. These packages are designed to run in parallel processes as well.

It took R about five seconds to create a five million row, three column double matrix with this command[3]:

```
Mymat
<- big.matrix(5000000,3,type='double',init=0,dimnames=list
(NULL,,c("first","second","third")))
```

It took less than a second to put 5 million random normal samples in the first column by this command:

```
Mymat[,"first"]<-rnorm(5000000,0,2)
```

It took a little bit longer to store 15 million random samples in the entire matrix by this command:

```
Mymat[,] <- rnorm(15000000,0,2)
```

You can read and write to big matrices as well. For instance, I'll first write Mymat out to a file "test.txt" and then I'll read it back in.

I was able to write this out in less than 30 seconds with this command:

```
write.big.matrix(Mymat,"test.txt")
```

The output file "test.txt" was comma delimited with three columns and 5 million rows.

Reading took less than 30 seconds when using this command:

```
MyMat2 <- read.big.matrix("test.txt",type="double")
```

There are many other commands within bigmemory that will allow you to share the same big.matrix object across multiple processors on your computer or across a cluster. I won't go into further detail regarding bigmemory, but if you are interested, please refer to Bigmemory.org for more information.

I also have loaded the biganalytics and the biglm packages to analyze Mymat.

Below are some summary statistics on my 15 million random normal draws:

```
> mean(Mymat)
[1] 0.0006737719
> colmean(Mymat)
```

first	second	third
0.0001679818	0.0021683848	-0.0003150509

```
> summary(Mymat)
```

	min	max	mean	NAs
first	-1.093267e+01	9.627050e+00	1.679818e-04	0.000000e+00
second	-1.088475e+01	9.889715e+00	2.168385e-03	0.000000e+00
third	-1.026170e+01	1.038852e+01	-3.150509e-04	0.000000e+00

I don't recommend this, but if you want the row averages you could run this command:

```
apply[Mymat,1,mean)
```

You can use the biglm function from the biglm package to determine a linear regression model on a big.matrix. For instance:

```
Mymat.biglm <- biglm.big.matrix(first ~ second + third,
data=Mymat)
```

This command ran in under 20 seconds and the results are:

Large data regression model: biglm(formula = formula, data = data, ...)

Sample size = 5000000

	Coef	(95%	CI)	SE	p
(Intercept)	2e-04	-0.0016	0.0020	9e-04	0.8500
second	-5e-04	-0.0014	0.0004	4e-04	0.3051
third	5e-04	-0.0004	0.0014	4e-04	0.2240

The above model is not at all significant, as was expected.

According to Kane and Emerson, you can expand these various packages to include other analytics.

I hope that you can find a use for big.matrix in your future work!

Footnotes

[1] Parts of this article are based on the "Memory Limits in R" R documentation. This documentation can be obtained in R by using the command: help("Memory-limits")

[2] Parts of this section are based on the R Documentation on bigmemory. Please use the command: help("bigmemory") to learn more about theses packages.

[3] If you copy these and subsequent commands that contain double quotes (") into R, you will need revise them to match the double quotes in R. This also applies to commands with single quotes.

[4] The first column of R Corner was published in the October 2008 issue, and explains how to download and install the package, as well as providing a basic introduction to the language. Refer to each CompAct issue since then for additional articles in the series. The introductory article can be found on p. 24 of the October 2008 issue on the SOA Web site. The R language is a very popular open source statistical modeling environment, which you can obtain from the webpage http://cran.r-project.org. If you are interested in building the toy model in this article, you will need to download the base R Binary for the operating system of your choice, (Windows, Linux or MAC). Also, you will need to download the contributed copula package. You can do this initially when you download the base system, or you can use the package installation facility when you start R.

Actuaries
Risk is Opportunity.®

475 North Martingale Road, Suite 600 Schaumburg, Illinois 60173
Phone: 847.706.3500 Fax: 847.706.3599 www.soa.org