Parallel R Loops for Windows and Linux

*January 17, 2012*

By [Vik Paruchuri](http://www.r-bloggers.com/author/vik-paruchuri/" \o "Posts by Vik Paruchuri)

Parallel computation may seem difficult to implement and a pain to use, but it is actually quite simple to use. The foreach package provides the basic loop structure, which can utilize various parallel backends to execute the loop in parallel. First, let's go over the basic structure of a foreach loop. To get the foreach package, run the following command: 

install.packages("foreach")

Then, initialize the library: 

library(foreach)

A basic, nonparallel, foreach loop looks like this: 

foreach(i=1:10) %do% {  
  
#loop contents here  
  
}

To execute the loop in parallel, the %do% command must be replaced with %dopar%: 

foreach(i=1:10) %dopar% {  
  
#loop contents here  
  
}

To capture the return values of the loop: 

list<-foreach(i=1:10) %do% {  
i  
}

Note that the foreach loop returns a list of values by default. The foreach package will always return a result with the items in the same order as the counter, even when running in parallel. For example, the above loop will return a list with indices 1 through 10, each containing the same value as their index(1 to 10).  
  
In order to return the results as a matrix, you will need to alter the .combine behavior of the foreach loop. This is done in the following code: 

matrix<-foreach(i=1:10,.combine=cbind) %do% {  
i  
}

This will return a matrix with 10 columns, with values in order from 1 to 10.  
  
Likewise, this will return a matrix with 10 rows: 

matrix<-foreach(i=1:10,.combine=rbind) %do% {  
i  
}

This can be done with multiple return values to create n x k matrices. For example, this will return a 10 x 2 matrix: 

matrix<-foreach(i=1:10,.combine=rbind) %do% {  
c(i,i)  
}

**Parallel Backends**  
  
In order to run the foreach loop in parallel(using the %dopar% command), you will need to install and register a parallel backend. Because windows does not support forking, the same backend that works a linux or an OS X environment will not work for windows. Under linux, the doMC package provides a convenient parallel backend.  
  
Here is how to use the package(of course, you need to install doMC first): 

library(foreach)  
library(doMC)  
registerDoMC(2) #change the 2 to your number of CPU cores   
  
foreach(i=1:10) %dopar% {  
  
#loop contents here  
  
}

Under windows, the doSNOW package is very convenient, although it has some issues. I do not recommend the doSMP package, as it has significant issues. 

library(doSNOW)  
library(foreach)  
cl<-makeCluster(2) #change the 2 to your number of CPU cores  
registerDoSNOW(cl)  
  
foreach(i=1:10) %dopar% {  
  
#loop contents here  
  
}

Edit:  Thanks to an alert reader, I noticed that I neglected to add in the code to stop the clusters.  This will need to be run after you finish executing all of your parallel code if you are using doSNOW.

stopCluster(cl)

Also please note that you will need to set the parameter in the makeCluster and registerDoMC functions to the number of CPU cores that your computer possesses, or less if you do not want to use all of your CPU cores.   
  
I hope that this has been a good introduction to parallel loops in R. The new version of R(2.14), also includes the parallel package, which I will discuss further in a later post. You can find more information on the packages mentioned in this article on CRAN. [Foreach](http://cran.r-project.org/web/packages/foreach/index.html" \t "_blank), [doSNOW](http://cran.r-project.org/web/packages/doSNOW/index.html" \t "_blank), and [doMC](http://cran.r-project.org/web/packages/doMC/index.html" \t "_blank) can all be found there.