CORE Package for Target inference from collections of genomic intervals (Version 1.1)

Alexander Krasnitz, Guoli Sun

June 23, 2013

1 Installation

To install `CORE` package, caution should be taken with installing the dependency packages, `parallel` and `Rmpi`, in advance. `parallel` is necessary for all operating systems. `Rmpi` is optional, it is needed when you are working on grid engine which has MPI software installed, and wish to accelerate the analysis.

parallel

From R version 2.14.0, `parallel` is already a buildin package, no need to install it. This package builds on the work done for CRAN packages `multicore` and `snow`. `http://stat.ethz.ch/R-manual/R-devel/library/parallel/doc/parallel.pdf`

Rmpi

Although `Rmpi` is available from CRAN, installation of `Rmpi` package may require manually adding path to your MPI environment.`http://www.stats.uwo.ca/faculty/yu/Rmpi` Make sure you have OPENMPI software before installing `Rmpi`.Depending on your operating system (Mac, Windows, Linux), there are different instructions. Here's an example on how to do it under Linux. Before installing, make sure you can load MPI environment using linux command line like this,

```
module load openmpi
```

Then to install the package, use command lines with extra configure settings:

```
R CMD INSTALL -l LIBDIR Rmpi_0.6-3.tar.gz -configure-args="
-with-Rmpi-include=MPIDIR/include/openmpi
-with-Rmpi-libpath=MPIDIR/openmpi/lib
-with-Rmpi-type=OPENMPI"
```

,where `LIBDIR` is your R library directory, where you want to put your `Rmpi` package.`MPIDIR` is the directory of your openmpi software. Any problems

occur with `Rmpi`, please contact `Rmpi` package maintainer, Hao Yu <hyu at stats.uwo.ca>.

As long as your R library has "parallel" and "Rmpi"(optional), you are ready for `CORE` package installation. `CORE` can be installed directly using R function

`install.packages("CORE")`

in an open R session, or using one command line under Linux, with the downloaded package.

`R CMD INSTALL CORE_1.0.tar.gz`

To load CORE (suppose you install it into default R library using above operations):

```
> library(CORE)
```

2 Examples

For arguments in function CORE, please refer to the manual page. This vignette only focuses on examples choosing the right option of `distrib` argument. There are four options,`"vanilla"`,`"Rparallel"`,`"mpi.ge"`.

vanilla

A single-processor mode of execution is chosen. In practice, it can only be used if the randomizations `nshuffle` is small.

```
> data(testInputCORE)
> data(testInputBoundaries)
> myCOREobj<-CORE(dataIn=testInputCORE,maxmark=10,nshuffle=1,
+ boundaries=testInputBoundaries,seedme=123)
> names(myCOREobj)

 [1] "input"         "call"          "minscore"
 [4] "maxmark"       "pow"           "assoc"
 [7] "coreTable"     "seedme"        "boundaries"
[10] "shufflemethod" "nshuffle"      "tiny"
[13] "simscores"     "p"

> dim(myCOREobj$coreTable)

[1] 10  4

> dim(myCOREobj$simscores)

[1] 10  1
```

Rparallel

`"Rparallel"` can perform multi-core tasks on your local machine. The number of cores is the minimum between number of local processors and the argument `njobs` (when the number of shuffles is larger than the above

two). Then the number of shuffles each task will perform, is roughly the number of (add-on) shuffles ÷ number of cores.

```
> #randomization can be performed with add-on shuffles
> #10 shuffle example based on object with three shuffles already
> newCOREobj<-CORE(dataIn=myCOREobj,keep=c("maxmark","seedme",
+ "boundaries"),nshuffle=10,distrib="Rparallel",njobs=2)
> dim(newCOREobj$simscores)

[1] 10 10

> #example without add-on shuffles
> myCOREobj<-CORE(dataIn=testInputCORE,maxmark=10,nshuffle=10,
+ boundaries=testInputBoundaries,seedme=123,distrib="Rparallel",
+ njobs=2)
> dim(myCOREobj$simscores)

[1] 10 10
```

mpi.ge

`"mpi.ge"` uses MPI (message passing interface) implementation in grid engine environment. This option outperforms `"Rparallel"` when dealing with large number of shuffles per processor. Unlike `"Rparallel"`, `njobs` when using `"mpi.ge"` gives the number of host machines. Number of shuffles on each host will then be accomplished by as many slave processors as possible from different machines. The results are gathered by each host through message passing functions. If the number of shuffles per slave processor is small, `"Rparallel"` is faster because there is overhead system time to activate MPI and submit grid engine job using `"mpi.ge"`. Run the example below only when you installed `Rmpi` and grid engine environment is available.(delete the # when you run the example)

```
> data(testInputCORE)
> data(testInputBoundaries)
> ##different parallel methods, time efficiency
> ##don' t run
> #system.time(myCOREobj<-CORE(dataIn=testInputCORE,maxmark=10,
> #nshuffle=40,boundaries=testInputBoundaries,distrib="mpi.ge",
> #seedme=123,njobs=2))
> #system.time(myCOREobj<-CORE(dataIn=testInputCORE,maxmark=10,
> #nshuffle=40,boundaries=testInputBoundaries,distrib="Rparallel",
> #seedme=123,njobs=2))
```

References

Alexander Krasnitz, Guoli Sun, Peter Andrews and Michael Wigler.(2013). Target inference from collections of genomic intervals. *PNAS 2013 : 1306909110v1-201306909.*