Using the doRNG package

doRNG package – Version 1.2.1

Renaud Gaujoux

March 9, 2012

Research reproducibility is an issue of concern, in particular in bioinformatics [3, 6, 4]. Some analyses require multiple independent runs to be performed, or are amenable to a split-and-reduce scheme. For example, some optimisation algorithms are run multiple times from different random starting points, and the result that achieves the least approximation error is selected. The *foreach* package[1] [1] provides a very convenient way to perform parallel computations, with different parallel environments such as MPI or Redis, using a transparent loop-like syntax:

```
# load and register parallel backend for multicore computations
library(doParallel)

## Loading required package: foreach

## Loading required package: iterators

## Loading required package: codetools

## Loading required package: parallel

registerDoParallel(2)

# perform 5 tasks in parallel
x <- foreach(i = 1:5) %dopar% {
    i + runif(1)
}
unlist(x)

## [1] 1.887 2.662 3.943 4.019 5.487
```

For each parallel environment a *backend* is implemented as a specialised `%dopar%` operator, which performs the setup and pre/post-processing specifically required by the environment (e.g. export of variable to each worker). The `foreach` function and the `%dopar%` handle the generic parameter dispatch and reducing step – when the results are returned to the master worker.

When stochastic computations are involved, special random number generators must be used to ensure that the separate computations are indeed statistically independent – unless otherwise wanted – and that the loop is reproducible. A random number generator commonly used to achieve this is the combined multiple-recursive generator from L'Ecuyer [5]. This generator can generate independent random streams, from a 6-length numeric seed.

The *doRNG* package[2] [2] provides a convenient way to implement reproducible parallel `foreach` loops, independently of the parallel backend used to perform the computation. We illustrate its use, showing how non-reproducible loops can be made reproducible, even when rerun with the

[1] http://cran.r-project.org/package=foreach
[2] http://cran.r-project.org/package=doRNG

tasks not scheduled in the same way in two separate set of runs, e.g. when the workers do not get to compute the same number of tasks or the number of workers is different.

```
# load the doRNG package
library(doRNG)
```

```
## Loading required package: methods
```

```
# with standard %dopar%: loops are not reproducible
set.seed(123)
res <- foreach(i = 1:5) %dopar% {
    runif(3)
}
set.seed(123)
res2 <- foreach(i = 1:5) %dopar% {
    runif(3)
}
identical(res, res2)
```

```
## [1] FALSE
```

```
# using %dorng%: reproducible
res <- foreach(i = 1:5, .options.RNG = 123) %dorng% {
    runif(3)
}
res2 <- foreach(i = 1:5, .options.RNG = 123) %dorng% {
    runif(3)
}
identical(res, res2)
```

```
## [1] TRUE
```

```
# even when the tasks are not scheduled in the same way
res <- foreach(i = 1:5, .combine = rbind, .options.RNG = 123) %dorng%
    {
        c(pid = Sys.getpid(), val = runif(1))
    }

cl <- makeCluster(3)
registerDoParallel(cl)
res2 <- foreach(i = 1:5, .combine = rbind, .options.RNG = 123) %dorng%
    {
        c(pid = Sys.getpid(), val = runif(1))
    }
stopCluster(cl)

# task schedule is different
pid <- rbind(res1 = res[, 1], res2 = res2[, 1])
storage.mode(pid) <- "integer"
pid
```

```
##      result.1 result.2 result.3 result.4 result.5
## res1     8163     8164     8163     8164     8163
## res2     8166     8175     8184     8166     8166
```

```
# results are identical
identical(res[, 2], res2[, 2])
```

```
## [1] TRUE
```

Session information

```
R version 2.14.2 (2012-02-29)
Platform: x86_64-pc-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_ZA.UTF-8       LC_NUMERIC=C               LC_TIME=en_ZA.UTF-8
 [4] LC_COLLATE=en_ZA.UTF-8     LC_MONETARY=en_ZA.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=C                 LC_NAME=C                  LC_ADDRESS=C
[10] LC_TELEPHONE=C             LC_MEASUREMENT=en_ZA.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] methods   parallel  stats     graphics  grDevices utils     datasets  base

other attached packages:
[1] doRNG_1.2.1      doParallel_1.0.0 foreach_1.3.2    codetools_0.2-8  iterators_1.0.5
[6] knitr_0.3.5

loaded via a namespace (and not attached):
 [1] compiler_2.14.2 digest_0.5.1    evaluate_0.4.1  formatR_0.3-4   highlight_0.3.1
 [6] parser_0.0-14   plyr_1.7.1      Rcpp_0.9.9      stringr_0.6     tools_2.14.2
```

References

[1] Revolution Analytics. *foreach: Foreach looping construct for R*, 2011. R package version 1.3.2.

[2] Renaud Gaujoux. *doRNG: Generic Reproducible Parallel Backend for foreach Loops*, 2010. R package version 1.2.1.

[3] Torsten Hothorn and Friedrich Leisch. Case studies in reproducibility. *Briefings in bioinformatics*, January 2011.

[4] John P A Ioannidis, David B Allison, Catherine A Ball, Issa Coulibaly, Xiangqin Cui, Aedín C Culhane, Mario Falchi, Cesare Furlanello, Laurence Game, Giuseppe Jurman, Jon Mangion, Tapan Mehta, Michael Nitzberg, Grier P Page, Enrico Petretto, and Vera Van Noort. The reproducibility of lists of differentially expressed genes in microarray studies. *Nature Genetics*, 41(2):149–155, 2008.

[5] Pierre L'Ecuyer. Good parameters and implementations for combined multiple recursive random number generators. *Operations Research*, 47(1), 1999.

[6] Victoria C Stodden. The Digitization of Science: Reproducibility and Interdisciplinary Knowledge Transfer, 2011.