Intro to Parallel Random Number Generation with RevoScaleR

Posted on June 6, 2013 by **Joseph Rickert** in Uncategorized | 0 Comments

by Joseph Rickert

Random number generation is fundamental to doing computational statistics. As you might expect, R is very rich in random number resources. The R base code provides several high quality random number generators including: Wichmann-Hill, Marsaglia-Multicarry, Super-Duper, Mersenne-Twister, Knuth-TAOCP-2002 and L'Ecuyer-CMRG. (See Random for details.) And, there are at least three packages, rspring, rlecuyer, and rstream for generating independent streams of random numbers in parallel. Independent streams are essential for parallel computations because even though small numbers of random draws done on different processors might show little correlation, as the the number of random draws approaches a significant fraction of the period length of the random number generator there are sure to be significant correlations among two or more streams.

High quality, independent, parallel random number streams are essential for doing statistical analyses with data sets having billions of observations. Random sampling, bootstrapping and ensemble techniques require generating independent random number streams in distributed environments. The RevoScaleR package in Revolution R Enterprise 6.2 handles this task by providing access to several of the parallel random number generators included in Intel's MKL Vector Statistical Library. RevoScaleR lets users select from among the following VSL random number generators:

- MCG31m1 – a 32-bit multiplicative congruential generator: fast and good for testing but the period length is not sufficient for working with big data
- R250 – a generalized feedback shift register generator: used commonly in physics applications but fails a number of tests.
- MRG32k3a – a combined multiple recursive generator: meets modern requirements for modern random number generators and is optimized for Intel architectures
- MCG59 – a 59 bit multiplicative congruential generator: implemented in the Numerical Algorithms Group, but quirky
- MT19937 – A Merseene Twiister with period length 2^19937

Most viewed posts (weekly)

R Shiny in Life Sciences – Top 7 Dashboard Examples
How to use functional programming for ggplot
February 2022: "Top 40" New CRAN Packages
5 Ways to Subset a Data Frame in R
How I analyze 100+ ggplots at once
R – Sorting a data frame by the contents of a column
Date Formats in R

Sponsors

- MT2203 – This set of 6024 random number generators provides mutual independence of the random number streams. It has a period length of 2^2203 and is the default generator for RevoScaleR's rxRngNewStream package
- SFMT19937 – is a Single Instruction Multiple Data Merseene Twister with a period length of (2^199370) – 1
- SOBOL – a quasi-random number generator
- NIEDERREITER – quasi-random number generator with excellent asymptotic properties

The following code shows how easily two independent random number streams can be generated with RevoScaleR's rxExec function. rxExec acts much like R's apply family of functions, but in a parallel or distributed computing environment. The first line tells the code underlying the RevoScaleR's parallel infrastructure to use 2 parallel workers. The second line "rxSetComputeContext("localpar)" instructs rxExec to use my laptop as the "compute" context. (If I had access to a cluster, either a Microsoft HPC cluster or a Linux LFS cluster this line of code would be a little more elaborate, but it would be the only line of code that would have to change to use a different, more powerful compute context.) The call to rxExec in line in line 20 instructs each core on my laptop generate a random number stream. The rest of the code performs some simple tests on resulting random number streams.

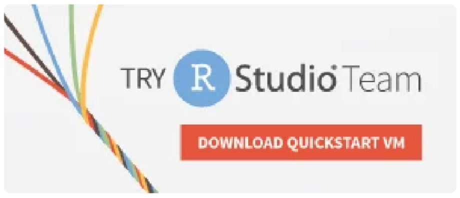
```
# Code for Random Numbers Blog Post
# Joseph Rickert
# Revolution Analytics
# June 2013
library(ggplot2)            # For plotting
#
#--------------------------------------------------------------------------
# Using the vectorized random number generators
# See the Intel Math Kernel Library, Vector Statistical Library Notes.
# http://software.intel.com/sites/products/documentation/doclib/mkl_sa/11/vslr
# for an explanation of the various random number generators
rkind = c("MCG31", "R250", "MRG32K3A", "MCG59", "MT19937", "MT2203", "SFMT1993

rxOptions(numCoresToUse=2)            # 2 cores on my Dell XPS, i7 laptop
rxSetComputeContext("localpar")           # Set the compute context to local
N <- 10000                  # Length of random number stream
# Generate random number stream
ranNums <- rxExec(runif, n = rxElemArg(c(N, N)), RNGkind = rkind[6])

ranNumsDF <- data.frame(ranNums)        # put the nmbers in a data frame
names(ranNumsDF) <- c("stream.1","stream.2")
length(unique(ranNumsDF$stream.1))
length(unique(ranNumsDF$stream.2))

cor(ranNumsDF)                    # Look at the correlation of the two streams
                         # See how the random numbers cover the plan
# If we were generating numbers from the Reals, then the
# differences of the sorted numbers we generate should be unique
uniqueDif <- function(x){
    # Function suggested by john.angell@csueastbay.edu
    x = sort(x)
    xu <- x[2:N] - x[1:N-1]
    len <- list(length(xu),length(unique(xu)))
    return(len)
    }
uniqueDif(ranNumsDF$stream.1)
uniqueDif(ranNumsDF$stream.2)
# Look at a scatter plot of the random number streams
ggplot(ranNumsDF, aes(x=stream.1, y=stream.2)) +
    geom_point(shape=1,colour="red") +     # Use hollow circles
    geom_smooth() +             # Add a loess smoothed fit curve with confidence
    ggtitle("Two Random Number Streams with Loess Estimator")
```

Both streams contain 10,000 unique numbers. The correlation matrix shows very little correlation between the two streams.



Beginner's Guide to
Spatial, Temporal and Spatial-Temporal Ecological Data Analysis with R-INLA
Zuur, Ieno, Saveliev

```
           stream.1    stream.2
stream.1 1.000000000 0.002468557
stream.2 0.002468557 1.000000000
```

The unique difference test, which was brought to my attention by John Angell of Cal State East Bay, is very interesting. If we were actually dealing with random Real numbers then we would expect the differences between adjacent numbers in a sorted list to be unique. However, even with high quality random number generators we must live with the reality of finite precision arithmetic.

```
uniqueDif(ranNumsDF$stream.1)
#[[1]]
#[1] 9999
#
#[[2]]
#[1] 9952
```

Finally, a scatter plot shows good coverage of the plane.

But, if you really want to put random number generators through their paces look at the diehard site and have a go at the tests in the RDieHarder package.

This post presents just the very basics. In future posts, I hope to show some parallel random number applications and results from running them on clusters.

Related

future 1.19.1 - Making Sure Proper Random Numbers are Produced in Parallel Processing Parallel 'Digital Rain' by Jahobr After two-and-a-half months, future 1.19.1 is now on CRAN. As usual, there are some bug fixes September 22, 2020 In "R bloggers"	future 1.19.1 - Making Sure Proper Random Numbers are Produced in Parallel Processing Parallel 'Digital Rain' by Jahobr After two-and-a-half months, future 1.19.1 is now on CRAN. As usual, there are some bug fixes September 22, 2020 In "R bloggers"	rTRNG: Advanced Parallel RNG in R Following on the recent CRAN release of rTRNG, it's time to show its features and usage more in detail. rTRNG is a new package June 10, 2019 In "R bloggers"

Share Tweet

To **leave a comment** for the author, please follow the link and comment on their blog: **Revolutions**.

R-bloggers.com offers **daily e-mail updates** about R news and tutorials about learning R and many other topics. Click here if you're looking to post or find an R/data-science job.

Want to share your content on R-bloggers? click here if you have a blog, or here if you don't.

← Previous post **Next post →**

Recent Posts

What is a horizon chart?
How I analyze 100+ ggplots at once
Track Shiny App User Activity With the RStudio Connect Server API
COVID-19 Data Hub Paper Published in Nature Scientific Data
R Shiny in Life Sciences – Top 7 Dashboard Examples

Search through your ecological data with the ‘grep()’ function
Using R to detect the pressure wave from the 2022 Hunga Tonga eruption in personal weather station data
Recreating the Storytelling with Data look with ggplot
How to download Kobotoolbox data in R
scikit-learn models in R with reticulate
rsnps 0.5.0: New ncbi_snp_query() Features
Simulating time-to-event outcomes with non-proportional hazards
Sylhet R User Group in Bangladesh Hopes to Get Back on Track with Physical Events
Nuclear Threat Projection with Neural Network Time Series Forecasting
Kadane’s algorithm – finding maximum sum in contigous sub-array

Jobs for R-users

Junior Data Scientist / Quantitative economist
Senior Quantitative Analyst
R programmer
Data Scientist – CGIAR Excellence in Agronomy (Ref No: DDG-R4D/DS/1/CG/EA/06/20)
Data Analytics Auditor, Future of Audit Lead @ London or Newcastle

python-bloggers.com (python/data-science news)

How to launch Jupyter notebooks from Windows
Python Exceptions – What, Why, and How?
How to Use R and Python Together? Try These 2 Packages
Python List Print – 7 Different Ways to Print a List You Must Know
How to get the most and least Volatile Cryptocurrencies
Firing Up Firestore
How to Get Cryptocurrency Data from Kraken API in Python

Full list of contributing R-bloggers

Archives

Select Month

Other sites

SAS blogs
Jobs for R-users