RcppZiggurat: Faster Random Normal Draws

Dr. Dirk Eddelbuettel
dirk.eddelbuettel@R-Project.org
edd@debian.org
@eddelbuettel

useR! 2014
UCLA, 1 July 2014
Outline

1. Context
   • Overview
   • Ziggurat
R and RNGs

R has “batteries included”:

- Excellent support for random number generation
- Documentation isn’t all that easy to find: `RNGkind`
- There are four Normal RNGs in R (and six Uniforms)
library(microbenchmark)
res <- microbenchmark(
  {RNGkind("Kinderman-Ramage"); rnorm(1e6)},
  {RNGkind("Ahrens-Dieter"); rnorm(1e6)},
  {RNGkind("Box-Muller"); rnorm(1e6)},
  {RNGkind("Inversion"); rnorm(1e6)},
  times=100)
R Normal RNGs

Time for 100 times 1e6 normal draws

- AH
- KR
- Inv
- BM

Time in msec

AH KR Inv BM

Dirk Eddelbuettel
RcppZiggurat
Marsaglia and Tsang, JSS, 2000

- Introduce a very small, very fast generator
- Just a few lines of C code (in MACROS)
- Plus two support functions
- But 32-bit only (which was ok at the time...)
#include <math.h>

static unsigned long jz, jsr=123456789;

#define SHR3 (jz=jsr, jsr^=(jsr<<13), jsr^=(jsr>>17), \
            jsr^=(jsr<<5), jz+jsr)
#define UNI (.5 + (signed) SHR3*.2328306e-9)
#define IUNI SHR3

static long hz;
static unsigned long iz, kn[128], ke[256];
static float wn[128], fn[128], we[256], fe[256];

#define RNOR (hz=SHR3, iz=hz&127, \
            (fabs(hz)<kn[iz])? hz*wn[iz] : nfix())
Three page note extending Ziggurat
Show a flaw in the underlying U(0,1) generator SHR3
Plugging in KISS, another Marsaglia U(0,1), as fix
This correction is not always used (!!)
And is still 32-bit only
Widely used Open Source generators

- *Lots* of other scripting languages have Ziggurat
- It is sometimes their default
- GNU GSL has a widely used implementation by Voss
- GNU Gretl borrows from it, as does QuantLib
2. RcppZiggurat
   - Base
   - Derived
   - Performance
```cpp
#include <cmath>
#include <cstdint.h>  // cstdint needs C++11

namespace Ziggurat {

class Zigg {
public:
    virtual ~Zigg() {};
    virtual void setSeed(const uint32_t s) = 0;
    // no getSeed() as GSL has none
    virtual double norm() = 0;
};
}
```
Our package provides several implementations

Our default Ziggurat class implements LZLLV in 32 and 64 bit mode

For comparison, we also have implementations

- ZigguratMT of Marsaglia and Tsang
- ZigguratLZLLV of Leong, Zhang et al (also the default)
- ZigguratGSL using the GNU GSL (by linking to GSL)
- ZigguratGt using the GNU Gretl generator (as an adapted copy)
- ZigguratQL using the QuantLib generator (as an adapted copy)
Ziggurat RNGs

Time for 100 times 1e6 normal draws
Outline

3 Accuracy
- Standard Test
- Normal Test
- Chi-Square Test
Use standard test for any Uniform generator

- Invert each Ziggurat draw from $N(0,1)$ to $U(0,1)$
- $n$ draws from a $U(0,1)$; compute sum of the $n$ values.
- Repeat $m$ times to create $m$ sums of uniform RNGs.
- With $n$ large, the $m$ results converge towards $N(n/2, \sqrt{n}/12)$ (Irwin-Hall distribution of sum of uniformly distributed values).
- Construct a $p$-value $p_i$ for each of $m$ values using the inverse of the Normal using the known mean and standard dev. from the Irwin-Hall distribution.
- With $m$ uniformly distributed values: standard tests such as Kolmogorov-Smirnow or Wilcoxon can be used to test for departures from uniform.
Standard test results

Ziggurat

MT

LZLLV

GSL

QL

Gretl

Draws: 5e+09
Repeats: 100
Seed: 123456789
Created at: 2014–06–18 18:59:41
Version: 0.1.2.2
Suggested new test for $N(0,1)$ generator

- $n$ from a $N(0,1)$, then compute sum of the $n$ values.
- Repeats $m$ times to create $m$ sums of $N(0,1)$ draws.
- With $n$ large, the $m$ results converge to $N(0, \sqrt{n})$.
- Construct a $p$-value $p_i$ for each of the $m$ values using the inverse of the Normal distribution.
- With $m$ uniformly distributed values: standard tests such as Kolmogorov-Smirnov or Wilcoxon can be used to test for departures from uniform.
Normal test results

Ziggurat

MT

LZLLV

GSL

QL

Gretl

pKS: 0.4365
pWil.: 0.4301

pKS: 0
pWil.: 0.0399

pKS: 0.4365
pWil.: 0.4301

pKS: 0.4003
pWil.: 0.3704

pKS: 0.0378
pWil.: 0.0602

pKS: 0.7187
pWil.: 0.5236

Draws: 5e+09 Repeats: 100 Seed: 123456789 Created at: 2014-06-18 10:24:03 Version: 0.1.2.2
Divide real line into $B$ equally spaced bins such that no $N(0, 1)$ draw should exceed furthest.

Follow Leong et al: range from -7 to 7 with 200 bins.

Large number of $N(0, 1)$ drawn; for each of a counter in the bin corresponding to the draw is increased.

After $N$ draws, the empirical distribution is compared to the theoretical (provided by the corresponding value of the Normal density function) using a standard $\chi^2$ test.
Chi–square test results

Ziggurat

MT

LZLLV

GSL

QL

Gretl

Total draws: 1e+10 Bins: 200 Seed: 123456789 Steps: 50 Created at: 2014–06–18 07:03:45 Version: 0.1.2.2
We posted an Rcpp Gallery article ‘Timing normal RNGs’ comparing the Normal RNG generators of R (via Rcpp), Boost and C++. We can now include the (default) Ziggurat method:

R> print(res[,1:4])

<table>
<thead>
<tr>
<th>test</th>
<th>replications</th>
<th>elapsed</th>
<th>relative</th>
</tr>
</thead>
<tbody>
<tr>
<td>zrnorm (n)</td>
<td>500</td>
<td>1.397</td>
<td>1.000</td>
</tr>
<tr>
<td>cxx11Normals (n)</td>
<td>500</td>
<td>4.917</td>
<td>3.520</td>
</tr>
<tr>
<td>boostNormals (n)</td>
<td>500</td>
<td>5.114</td>
<td>3.661</td>
</tr>
<tr>
<td>rcppNormals (n)</td>
<td>500</td>
<td>6.166</td>
<td>4.414</td>
</tr>
</tbody>
</table>

R>
5 Summary
RcppZiggurat implements updated Ziggurat generators which now work for 32- and 64-bit OSs.

- Simple use: Include a single header (per generator)
- Generator is fast, small, tested and has minimal state
- This makes it particularly useful for thread-local use in parallel simulations.
- Package has been on CRAN for a few months, will get update in the next few weeks.