RDieHarder: An R interface to the DieHarder RNG test suite

Dirk Eddelbuettel[1] Robert G. Brown[2]

[1]Debian Project

[2]Duke University

UseR! 2007
Iowa State University
August 8 - 10, 2007

Outline

Quote

"The generation of random numbers is too important to be left to chance."

– Robert R. Coveyou

Another Quote

“Anyone who attempts to generate random numbers by deterministic means is, of course, living in a state of sin.”

– John von Neuman

Why does RNG quality matter?

Random numbers play an ever-increasing role in statistics via

- estimation: Monte Carlo Markov Chain, randomizing methods, permutations
- inference as the Bootstrap has become a standard tool
- various simulation methods (also used for estimation)

Outside of statistics, random numbers are critically important for encryption and secure communications protocols, but we completely ignore that aspect here.

We want to ensure that our RNG is behaving consistently in producing 'random' (i.e. unpredictable, without decernable patterns) numbers.

Basic idea of RNG testing

A conceptual algorithm

The core idea is as follows:

- assume we have an RNG that produces random integers
- conduct one 'experiment' and draw M random numbers
- arrange these M integers as a binary vector of bit length N
- as there should be as many zeros as ones, a test statistic is to measure the number of ones in the vector which should be normally distributed with mean $N/2$ and variance $\sqrt{N}$
- so we can compute a p-value for this experiment of M draws

Hence, given an RNG and a test statistic, we obtain one p-value.

The second key idea is that the p-value itself should be uniformly distributed – so we can test a series of these p-values against departures from a uniform distribution.

Basic idea of test algorithm

Going back to Kendall and Babington-Smith

Slightly more formally stated:

- formulate H_0 which assumes that the RNG to be tested is perfect
- select a test statistic that can operate on a sequence of RNG draws and has a 'known' distribution
- perform an experiment by drawing M random numbers and evaluating the test statistic to obtain a p-value under H_0
- repeat previous step n times to obtain a p-value vector of size n
- test resulting vector of size n for uniform distribution using e.g. a KS test
- the p-value of this last test provides the result for the RNG test
- evaluate H_0 using this p-value and possibly reject it

The original DieHard test library

George Marsaglia

DieHard by George Marsaglia is often seem as the ‘gold standard’ of RNG testing with his ’diehard’ battery of tests. However, there are some issues:

- Written in Fortran without comments, not particularly extensible or modular
- no copyright notice or license
- target statistics and distribution derived via state-of-the-art simulations ... of fifteen years ago
- fixed file-based inputs requiring fixed format – of rather limited size given the cpu speed and memory size of today’s computers
- no user-selectable parameters

The DieHarder re-implementation and extensions

Robert G. Brown

Over the last few years, Brown has written DieHarder:

- reimplemented to be extensible
- rewritten in modular, portable, standard C
- wraps around over sixty RNGs from the GNU GSL
- additional test statistics from NIST's STS suite (with a crypto focus) and by Brown, including a timer
- implemented as core library plus command-line frontend
- released under GNU GPL

So what's not to like?

Hm, maybe the 'research triangle, 1980s' look and feel?

```
edd@ron:~> dieharder -g 17 -d 12 -p 500 -S 0
#==================================================================
#         Diehard Minimum Distance (2d Circle) Test
# It does this 100 times::   choose n=8000 random points in a
# square of side 10000.  Find d, the minimum distance between
# the (n^2-n)/2 pairs of points.  If the points are truly inde-
# pendent uniform, then d^2, the square of the minimum distance
# should be (very close to) exponentially distributed with mean
# .995 .  Thus 1-exp(-d^2/.995) should be uniform on [0,1) and
# a KSTEST on the resulting 100 values serves as a test of uni-
# formity for random points in the square. Test numbers=0 mod 5
# are printed but the KSTEST is based on the full set of 100
# random choices of 8000 points in the 10000x10000 square.
#
# This test uses a fixed number of samples -- tsamples is ignored.
# It also uses the default value of 100 psamples in the final
# KS test, for once agreeing precisely with Diehard.
#==================================================================
#                        Run Details
# Random number generator tested: ran0
# Samples per test pvalue = 8000 (test default is 8000)
# P-values in final KS test = 500 (test default is 100)
#==================================================================
#                Histogram of p-values
# Counting histogram bins, binscale = 0.100000
#    120|    |    |    |    |    |    |    |    |    |    |
#       |    |    |    |    |    |    |    |    |    |    |
#    108|    |    |    |****|    |    |    |    |    |    |
#       |    |    |    |****|    |    |    |    |    |    |
#     96|    |    |    |****|    |    |    |    |    |    |
#       |    |    |    |****|    |    |    |    |    |    |
#     84|    |    |    |****|    |    |    |    |    |    |
#       |    |    |    |****|    |    |    |    |    |    |
#     72|    |    |    |****|    |    |    |****|    |    |
#       |    |    |    |****|    |    |    |****|    |    |
#     60|    |    |    |****|    |    |    |****|    |    |
#       |    |    |    |****|    |    |    |****|    |    |
#     48|****|    |    |****|    |    |    |****|    |****|
#       |****|    |    |****|****|****|    |****|****|****|
#     36|****|    |    |****|****|****|    |****|****|****|
#       |****|    |****|****|****|****|    |****|****|****|
#     24|****|****|****|****|****|****|****|****|****|****|
#       |****|****|****|****|****|****|****|****|****|****|
#     12|****|****|****|****|****|****|****|****|****|****|
#       |****|****|****|****|****|****|****|****|****|****|
#       |--------------------------------------------------
#       | 0.1| 0.2| 0.3| 0.4| 0.5| 0.6| 0.7| 0.8| 0.9| 1.0|
#==================================================================
#                          Results
Kuiper KS: p = 0.000000
Assessment: FAILED at < 0.01% for Diehard Minimum Distance (2d Circle) Test
edd@ron:~>
```

RDieHarder

A port to R

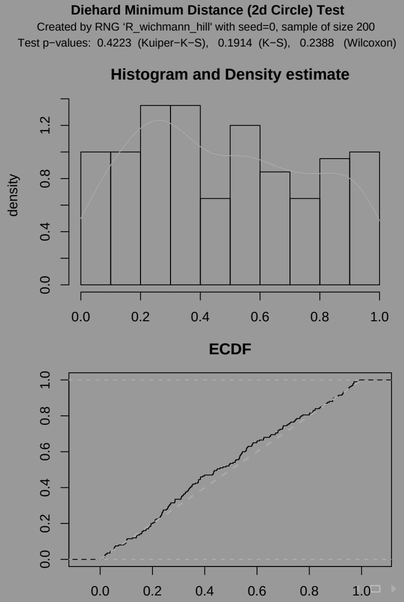
Straightforward 'port' to R given the layout of dieharder

- R package provides access to dieharder library
- Access to all RNGs in dieharder, and all test statistics
- The `dieharder` function replaces the `dieharder` command-line interface ...
- **but** also returns results data to R for further analysis, visualization, and different tests by ...
- returning a `dieharder` object with `print`, `summary` and `plot` methods.

At the same time, also 'ported' R's RNGs to dieharder's framework of RNG further extending the set of RNGs in dieharder.

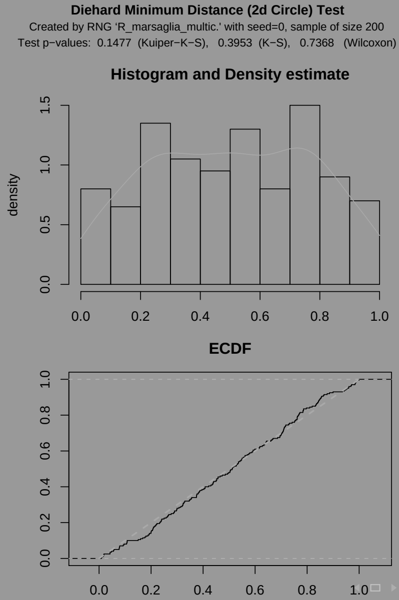
Example: R RNGs

Wichmann-Hill



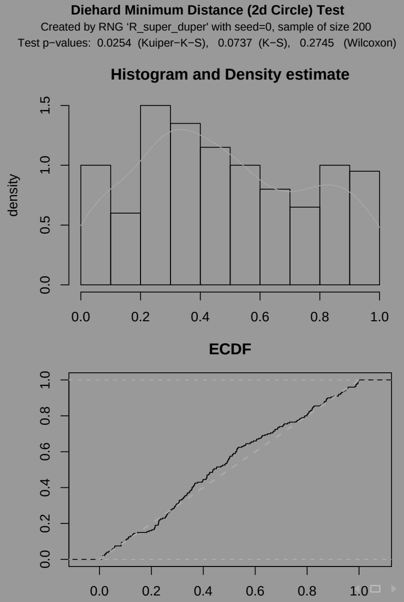
Example: R RNGs

Marsaglia Multicarry



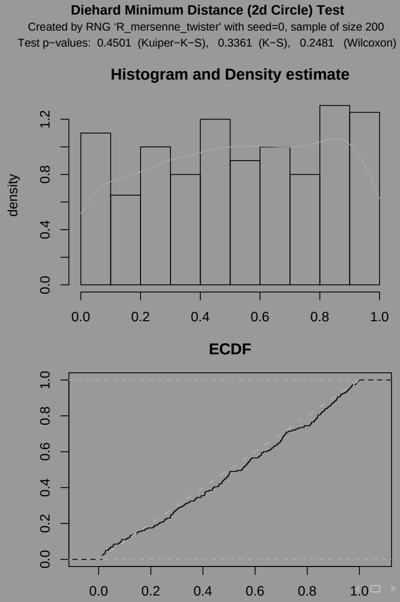
Example: R RNGs

Super Duper – failing Kuiper-K-S



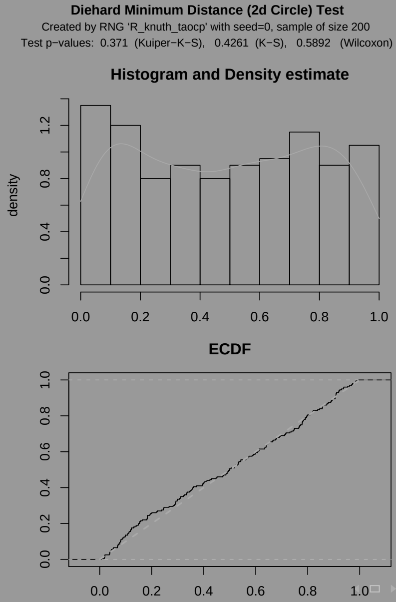
Example: R RNGs

Mersenne Twister



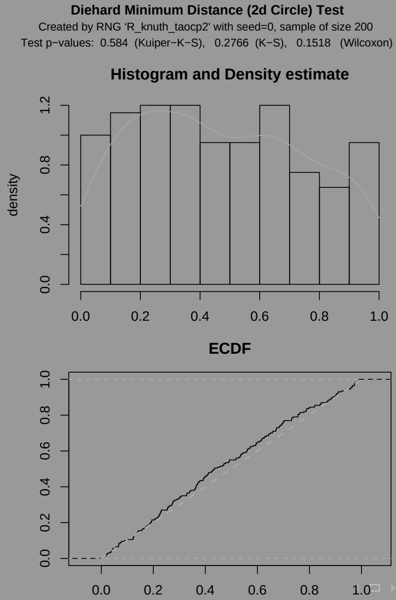
Example: R RNGs

Knuth TAOCP



Example: R RNGs

Knuth TAOCP2



RDieHarder contributions

RDieHarder ...

- brings the analytical framework of the DieHarder tests for random number generators to R
- allows the test statistics to be analysed further in R

Open venues for research

- more and better tests
- innovative use of the test results
- possibly more flexible architecture allowing callbacks into R

RDieHarder availability

On Debian or Ubuntu (now that RDieHarder is on CRAN)

```
$ sudo apt-get install libdieharder-dev
$ echo "install.packages('RDieHarder',
          '/usr/local/lib/R/site-library',
          'http://cran.us.r-project.org')" |
           sudo R --slave
```

On other Unix systems, download the dieharder sources from Robert G. Brown's site at Duke, do `configure; make; make install` and run `install.packages('RDieHarder')` from R as usual.

SVN access via http://code.google.com/p/rdieharder/

'Soon' http://dirk.eddelbuettel.com/code/rdieharder