Masaryk University
Faculty of Informatics

The automated testing of randomness with multiple statistical batteries

Master's Thesis

Ľubomír Obrátil

Brno, Spring 2017

Replace this page with a copy of the official signed thesis assignment and the copy of the Statement of an Author.

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Ľubomír Obrátil

Advisor: RNDr. Petr Švenda, Ph.D.

Acknowledgement

TODO

Abstract

TODO

Keywords

TODO

Contents

1 Introduction

- Randomness, why should we test it (defects, low entropy, etc...)
- Statistical testing of randomness

2 Used third-party statistical software

2.1 Terminology

In this thesis we are using certain expressions in the context of Randomness Testing Toolkit and the tools it uses. We list these expressions along with their explanations here.

(Statistical) Battery
Program developed by a third party serving as a tool for evaluation of randomness of arbitrary binary data. Statistical battery usually contains one or multiple statistical tests. Final result of the evaluation is based on the results of the tests. Examples of statistical batteries are NIST Statistical Test Suite, Dieharder or TestU01.

(Statistical) Test
Single unit in statistical battery that measures some property of the tested binary stream (e.g. number of zeroes). Based on this measurement, first level p-value of the test is calculated. The test can be repeated, resulting in multiple p-values. From these p-values are obtained one or multiple second level p-values. Result of the test can then be either first level p-value or one or more second level p-values called statistics.

Variant of a test
Many tests can be parametrized in some ways, possibly giving different results with the same input data. We don't treat multiple executions of single test with different settings as separate units but rather as a variants of that test.

Subtest
Some tests, even when executed only once, may measure multiple properties of the data thus providing multiple results. For example, Serial Test from Dieharder battery will measure frequencies of all 1, 2, .., 16-bit patterns in the data. We treat these measurements separately - as a subtests of the test. Therefore subtest is smallest unit in the battery, providing only single result.

Statistic
Value obtained by certain calculation from first level p-values. Multiple statistics can be obtained from one set of p-values e.g. when testing the set for uniformity we can use Kolmogorov-Smirnovov Test or Chi-Square test.

p-value
We refer to first level p-values as to simply p-values and second level p-values as to statistics. P-value is obtained as a result of single execution of a test. This p-value tells us the probability that the input data was truly random and it is value between 0 and 1. Hence obtaining p-value very close to 1 means that the data was almost certainly produced by TRNG. When

the p-value is close to 0 it means that it is almost impossible to obtain such data from TRNG; the data is probably not random.

Alpha - α
Critical value based on which we evaluate certain p-value. If p-value falls into interval specified by α (e.g. $[0, \alpha)$) we can say that the result is failure or success when the p-value is outside of the interval.

2.2 Batteries supported by RTT

2.2.1 NIST Statistical Test Suite

The battery of statistical tests was developed by National Institute of Standards and Technology (cit.). The battery implements 15 statistical tests for evaluating randomness of input data.

The reference implementation is not used in RTT because it is considerably slower than its optimized counterparts. The optimized version of NIST STS used in RTT was developed by Zdeněk Říha and Marek Sýs(cit.).

2.2.2 Dieharder

Dieharder is a battery developed by Robert G. Brown at the Duke University (cit.). The battery features user friendly console interface with possibility of fine-grain modification of the test parameters. The fact that Dieharder is included in repositories of some Linux distributions (cit manpage) adds to its popularity and ease of use. Dieharder includes all tests from older statistical battery Diehard (cit.), three tests from NIST STS and several other tests implemented by the author.

Since original Dieharder implementation doesn't output all of the information we needed for interpretation and evaluation of the results, we had to modify source code of the battery. RTT uses this modified Dieharder.

2.2.3 TestU01

This library of statistical batteries was developed at Université de Montréal by Pierre L'Ecuyer et al (cit.). It contains wide range of tests from NIST STS, Dieharder and literature. It also implements numerous pseudo-random number generators. The statistical tests are grouped into multiple categories each intended for different use-case scenario. We will treat these categories of tests as separate batteries. TestU01 includes following 10 batteries.

Small Crush, Crush, Big Crush
Small Crush battery is very fast and needs relatively small amount of data

to run - around 250 megabytes. Small Crush is also the only battery that can be natively used for analysis of data in binary file, for use of Crush and Big Crush, the user has to implement PRNG with TestU01 interface. Crush and Big Crush batteries are more stringent and need gigabytes of data and a few hours to finish while Big Crush is more time and data demanding.

Rabbit, Alphabit, Block Alphabit
Tests in these batteries are suited for testing hardware bit generators and are applicable to arbitrary amount of data. Data can be provided either as a binary file or PRNG implementing TestU01 interface.

PseudoDIEHARD, FIPS_140_2
Tests in PseudoDIEHARD imitate DIEHARD battery and FIPS_140_2 battery implements small suite of tests in NIST standard(cit.) Randomness Testing Toolkit doesn't support these two batteries since they are subsets of other supported batteries.

Since TestU01 is available only as a ANSI C library, we developed a console interface for it. The interface implements a dummy number generator that provides data from a supplied binary file to the battery. This allows us to apply batteries to arbitrary binary data even when the batteries don't support this feature natively.

2.3 Known errors in the batteries

To examine the boundary behavior of the above-listed tools, we used them to process extremely non-random data streams. The data streams that were used as the input to the batteries were two binary data files consisting only of zeroes and ones respectively. The settings of the batteries remained set to default.

Below we list observed behavior that differ from execution of the batteries with non-extreme input.

NIST Statistical Test Suite

Tests Random Excursions and Random Excursions Variant are not applicable to all possible data streams. In a normal run with reasonably varied data, this doesn't matter much, as the tests are repeated multiple times and the final result will simply be calculated from lesser number of p-values.

Neither of the tests is applicable to a stream full of zeroes or ones. This causes absence of results when analyzing such data. The user can find out the fact that the tests are not applicable to provided stream after he inspects logs of the program, otherwise the interpretation of the missing results is left to him.

Dieharder

When processing extremely non-random data with Dieharder, we observed various erroneous events. The events are summarized in Table 2.1.

Observed errors

- **No result** Test didn't provide any p-values that would be used to calculate the final result of the test. User is not notified about this and the final result of the test statistic is based on default value (1.0).
- **Invalid result** Test provided resulting statistic and there was no indication of error other than that the result was again default value of statistic (1.0). Following definition of p-value, the interpretation of such result is that the analyzed stream was almost certainly generated by TRNG. This is obviously not true, as both streams are just repeating ones or zeroes respectively.
- **Stuck execution** Tests froze at a certain point in execution, did not produce any results and we were forced to manually kill the processes.

Test name	Stream of zeroes	Stream of ones
STS Runs	No result	No result
DAB Monobit 2	Invalid result	Invalid result
Diehard Minimum Distance (2D Circle)	Stuck execution	-
Diehard 3D Sphere (Minimum Distance)	Stuck execution	-
Marsaglia and Tsang GCD	Stuck execution	-
RGB Generalized Minimum Distance	Stuck execution	-
RGB Kolmogorov-Smirnov	Stuck execution	-
Diehard Craps	-	Stuck execution
RGB Bit Distribution	-	Stuck execution

Table 2.1: Erroneous tests in Dieharder battery

TestU01

Similarly to Dieharder battery, TestU01 battery also contains tests that has undocumented behaviour. The errors are summarized in Table 2.2.

Observed errors

- **No results** Tests reported warning and ended without any result. This is probably caused by the tests not being applicable to provided data.
- **All results invalid** All statistics of the test reported p-value very close to 1.0. This could lead user to the interpretation that the test reports the data as produced by almost perfect TRNG.

- **Some results invalid** Similar situation to the previous one but not all statistics of the test are close to 1.0. Results of tests statistics are either close to 0.0 or to 1.1.
- **Stuck execution** Tests froze at a certain point of execution. In some cases this is preceeded by an issued warning. Tests didn't produce any results and had to be killed manually.

Test name	Stream of zeroes	Stream of ones
sstring_Run	No result	No result
sknuth_Gap	No result	-
svaria_SampleProd	All results invalid	All results invalid
svaria_AppearenceSpacings	All results invalid	All results invalid
scomp_LinearComp	All results invalid	All results invalid
scomp_LempelZiv	All results invalid	All results invalid
svaria_SampleCorr	-	All results invalid
sknuth_MaxOft	Some results invalid	Some results invalid
svaria_SampleMean	Some results invalid	Some results invalid
sspectral_Fourier3	Some results invalid	Some results invalid
sstring_HammingWeight2	Some results invalid	Some results invalid
sstring_AutoCor	Some results invalid	Some results invalid
smultin_MultinomialBitsOver	Some results invalid	Some results invalid
sstring_LongestHeadRun	-	Some results invalid
snpair_ClosePairs	Stuck execution	Stuck execution
snpair_ClosePairsBitMatch	-	Stuck execution
svaria_SumCollector	-	Stuck execution
smarsa_GCD	-	Stuck execution

Table 2.2: Erroneous tests in TestU01 library

2.3.1 Preventive measures in RTT

Since we need to use the batteries in RTT even in cases of extreme data, we implemented following measures that mitigate above-mentioned errors.

- Tests that don't produce any results are ignored and treated as if never executed.
- Because some tests give 1.0 as a result of their statistics when the data are clearly not random, we will evaluate result as failure either when is too close to 0 or too close to 1. More specifically, all p-values that falls into interval $\left[\frac{\alpha}{2}, \frac{\alpha}{2}\right]$ will be considered success and failure otherwise.
- Each test is executed with timeout. If the test doesn't finish within defined time limit, it is automatically terminated. The test is then treated as if it didn't produce any results.

3 Randomness Testing Toolkit

- Motivation - unified interface to batteries, ease of use, unified result format/representation
- Local execution of RTT - battery and toolkit configuration, installation, brief implementation and interface overview - more thorough in documentation and comments
- Local result format - either database or file output storage
- Remote execution of RTT - toolkit deployed on server infrastructure, system overview (database, frontend, backend(s), storage), accessible through ssh on limited system or via web interface (django), results in database
- Remote results of RTT - email notification, webpage layout
- Interpretation of results of RTT
  - Grouping subtests together - eliminating intertest bias
  - How grouping works - theory, Sidak correction, partial p-value, fail/pass of a test

4 Analysis of outputs of cryptographic functions, comparison with EACirc

- How the data were tested
- List functions
- List interesting (differing) results - Dieharder, NIST STS, TestU01, EACirc, polynomials(???)

5 Analysis of DIEHARDER results on quantum random data

- Statistical intro, uniformity, first vs. second level p-value, etc...
- Two experiments - continuous p-values, blocks of 2nd level
- Results - non-uniform, where it will begin to show on 2nd level results

6 Conclusions

- Developed user-friendly tool for easy analysis of arbitrary binary data - Randomness Testing Toolkit
- Interpretation of results
- Comparison of batteries with EACirc, polynomials
- Defects in Dieharder, their relevance, etc...
- Future work, same analysis on TestU01, dependence between tests(?), continuous development of RTT, call for flawless statistical battery (:))

A An appendix

TODO