Making randomness tests more robust

Alexander Shen

Making randomness tests more robust

Alexander Shen*

Abstract

In this note we suggest a simple change in practical randomness tests that could make them more robust (not dependent on the asymptotic formulas and etalon random sources).

1 What is a randomness test?

Imagine that we want to check some "null hypothesis", e.g., the hypothesis of a fair coin (all bits are independent with probabilities 1/2 for head and tail). The standard way is to choose some *test*, a subset $F \subset \{0,1\}^n$ of the set of all possible outcomes that has small probability, and agree that an appearance of a sequence from F will be interpreted as a failure (the hypothesis is rejected).[1] The probability of F, usually called the *significance level*, is the probability of rejecting a sequence that is produced according to the null hypothesis. Its choice depends on the level of certainty we need when rejecting the null hypothesis. For example, a simple test could consists of all strings of some length N where the frequency of ones exceeds some threshold $t > 1/2$. The significance level is then the fraction of N-bit strings that contain more than tN ones, and this fraction should be small to make the rejection meaningful.

*LIRMM CNRS & University of Montpellier. Supported by RaCAF ANR grant.

[1]There are many philosophical and practical questions related to this procedure. Normally we should choose a test before the experiment, but what can be done if the experiment already happened? What are the "reasonable" tests that can be meaningful in such a situation? Algorithmic information theory considers "simple" tests and identifies "randomness" with incompressibility, but how can we test incompressibility? Does it make any sense to test algorithms that claim that they produce a "pseudorandom" sequence — without a seed, like the binary expansion of π, or with a seed? (This latter setting corresponds to the Yao–Blum–Micali definition of pseudo-random number generators.) We do not discuss these questions here.

More general, we may consider any *test statictic*, i.e., a random variable ξ that is a real-valued function defined on the outcomes (n-bit strings in our example), and for every outcome c consider the *p-value* that corresponds to this outcome, i.e., the probability of the event $\{x \mid \xi(x) \geq \xi(c)\}$. In this way we get a family of tests: for every $\varepsilon > 0$ we get a test T_ε that consist of all outcomes that have p-value at most ε. The probability of T_ε is at most ε (more precisely, it is equal to the maximal p-value that does not exceed ε). So different observers may look at the p-value for the tested string, compare this p-value with their favorite significance level and decide whether to reject the null hypothesis based on the test result (this happens if p-value is below the significance level) or not. Popular collections of randomness tests (like `diehard` [1] and more extensive `dieharder` [2]) use this approach: they do not say just "rejected" or "accepted" but provide the p-value for many different tests (random variables ξ in our notation).

2 Secondary tests

An important part of mentioned test collections [1, 2] are "secondary" tests based on the following observation. Let ξ be an arbitrary random variable; consider the new random variable $\eta = p(\xi)$, where p is the p-value function constructed for the variable ξ. The main observation: *the distribution of this random variable is close to the uniform distribution*. It would be exactly the uniform distribution if ξ were a continuous random variable such that every individual value has probability 0. In general, the cumulative distribution function $\Pr[\xi \leq c]$ equals the maximal p-value that does not exceed c. The distance between the consecutive p-values is the probability of an individual outcome for ξ, so for almost all tests the distribution is very close to the uniform one.

So we can apply the test to non-overlapping (and therefore independent) segments of the input sequence, and then test the sequence of p-values obtained for them against the uniform distribution. The test collections we mentioned [1, 2] use the Kolmogorov–Smirnov test for this comparison. And this is important: in many cases the p-values obtained by the primary tests are not very close to zero; the significant discrepancy appears only at the second stage.

3 Problems with secondary tests

The first problem is already mentioned: the distribution of p-values is not exactly uniform. But the difference usually is negligible, since the distance between empirical and theoretical cumulative distributions is much bigger anyway. More dangerous are the assumptions that are made in the tests. The birthday spacing test from the original collection uses the overlapping sequences of bits for several tests, so there are no reasons to claim that the p-values will have uniform distribution.[2] The same problem appears with 6×8 matrix rank test from `diehard` (Kolmogorov–Smirnov test is applied to dependent distrubutions).

The bigger problem arises because for most of these tests the distribution is not really computed exactly; instead, some heuristic assumptions are used (based on asymptotic convergence, or some experimental measurements[3]). So we do not have proofs that the approximate p-values coincide with the correct ones. The problem becomes even more serious when the secondary tests are used: in a primary test we may believe for a good reason that the approximation provides some upper bound for a correct p-value. However, for the secondary tests bounds are not enough, and we have no reason to believe in the correctness of the tests.

4 "True randomness" calibration

The natural idea would be to check first that the tests are valid by applying them to sequences that are believed to be random. If some test systematically rejects sequences that (we believe) are produces by a reliable source of random bits, then the test should be discarded. The `dieharder` documentation says about this problem:

> Many dieharder tests, despite our best efforts, are numerically unstable or have only approximately known target statistics or are

[2] "The first test uses bits 1–24 (counting from the left) from integers in the specified file. Then the file is closed and reopened, then bits 2–25 of the same integers are used to provide birthdays, and so on to bits 9–32. Each set of bits provides a p-value, and the nine p-values provide a sample for a KSTEST" (`cdbday.c` from `diehard`).

[3] The comments for one of the tests (OQSO) in `diehard` say "The mean is based on theory; sigma comes from extensive sumulation".

> straight up asymptotic results, and will eventually return a failing result even for a gold-standard generator (such as AES), or for the hypercautious the XOR generator with AES, threefish, kiss, all loaded at once and xor'd together. ⟨...⟩ Failure with numbers of psamples within an order of magnitude of the AES thresholds should probably be considered possible test failures, not generator failures. Failures at levels significantly less than the known gold standard generator failure thresholds are, of course, probably failures of the generator.

So some tests in [2] are labeled as suspect tests that consistently fail "good" generators. It would be desirable to use some more reliable approach, and there is an obvious possibility. Instead of using Kolmogorov–Smirnov test to compare the p-values distribution with the hypothetical uniform distribution, we may use two-sample Kolmogorov–Smirnov test (or any other test for comparing two distrubutions) to check whether the p-values obtained for the generator under testing are indistinguishable (up to some significance level) from the p-values obtained for the "gold standard generator" (assume for now that we have one).

The additional advantage of this approach is that now we do not need to compute the p-values, since only the ordering of them matters, so we can directly work with the values of the underlying test variable and do not need any information about its distribution. The price we pay is that we need more data. Indeed, we need the p-values both for generator we test and for the etalon generator. Also we need more samples for the same precision, since we now compare two empirical distributions and each has some sampling error.

5 What if do not have "true randomness"?

The trick mentioned in the preceding section assumes that we have some generator that we trust to be random. This assumption is crucial: If we use some faulty generator as an etalon for comparison, we may easily reject the good generator or accept a bad one. The second case may happen anyway for every test, but the first situation is hardly acceptable.

However, there is an easy way to get around this problem. Instead of using N test runs (strings $x_1, \dots, x_N$) of our generator and comparing then with N test runs ($e_1, \dots, e_N$) of an "etalon" (presumably good) generator, we may take $2N$ runs $x_1, \dots, x_{2N}$ of our generator and N test runs $e_1, \dots, e_N$ of the etalon

generator, and then compare the distribution of test values for $x_1, \ldots, x_N$ and for $x_{N+1} \oplus e_1, \ldots, x_{2N} \oplus e_N$ (here $\oplus$ stands for bitwise `xor`). This is a valid test *for every* $e_1, \ldots, e_N$ (assuming they are independent from $x_1, \ldots, x_{2n}$; we may use some fixed values since this guarantees independence). On the other hand, if we use true random generator to obtain $e_1, \ldots, e_N$, then the second half will contain true random variables, so the test is as powerful as before. The only disadvantage is that we need twice more bits for testing.

References

[1] G. Marsaglia, *DIEHARD: a battery of tests of randomness*. Archived: `http://web.archive.org/web/19971014105330/http://stat.fsu.edu:80/pub/diehard/cdrom/`

[2] R.G. Brown, *Dieharder: A Random Number Test Suite. Version 3.31.1*, available at `https://webhome.phy.duke.edu/~rgb/General/dieharder.php`.