Rabbit and NIST Test Batteries in TestU01

Prof. Michael Mascagni

Department of Computer Science
Department of Mathematics
Department of Scientific Computing
Graduate Program in Molecular Biophysics
Florida State University, Tallahassee, FL 32306 **USA**
E-mail: mascagni@fsu.edu
URL: http://www.cs.fsu.edu/∼mascagni

Why randomness testing matters

- Random number generators are used in simulation, cryptography, games, and scientific computing.
- A generator can pass simple visual checks and still fail deeper statistical tests.
- Test batteries are designed to expose structure, bias, and hidden correlations.

What is TestU01?

TestU01 is a software library for statistical testing of random number generators.

- Developed by Pierre L'Ecuyer and Richard Simard.
- Provides a collection of tests organized into batteries.
- Common batteries include **SmallCrush**, **Crush**, and **BigCrush**.
- Also supports other test suites and custom experimental workflows.

Core idea

A battery is a collection of tests with increasing strength and cost.

How to read a randomness test

- Each test outputs a statistic and a *p-value*.
- Under the null hypothesis of true randomness, p-values should be roughly uniform on $[0, 1]$.
- Very small or very large p-values suggest structure, bias, or dependence.
- One failure may happen by chance; repeated failures across many tests are more serious.

Rule of thumb

A generator is not proved random by passing tests, but a weak generator is often exposed by them.

The Rabbit battery

The **Rabbit** battery is a lighter-weight collection of tests in the TestU01 ecosystem.

- Designed for quicker screening than the largest batteries.
- Useful when you want a fast check before running heavier suites.
- Helps detect obvious flaws such as short-period structure, bias, and basic dependence.
- Often used as an intermediate step between exploratory testing and stronger batteries.

Practical role

Rabbit is good for early diagnostics; it is not the final word on generator quality.

What Rabbit is good at

- Catching low-hanging defects quickly.
- Screening candidate RNGs before expensive testing.
- Identifying generators with obvious linear structure or poor mixing.
- Providing faster feedback during implementation and tuning.

Interpretation

If a generator fails Rabbit, it is usually not worth trusting in more demanding applications.

The NIST test battery

The **NIST Statistical Test Suite (STS)** is a widely known set of randomness tests standardized by NIST.

- Focuses on evaluating binary sequences.
- Includes tests such as frequency, block frequency, runs, longest run, rank, FFT, and approximate entropy.
- Often used for hardware RNGs, cryptographic sources, and certification-related evaluation.
- Emphasizes practical validation of observed bit streams.

Difference from TestU01

NIST STS is a separate suite, but it is often discussed alongside TestU01 because both are used to assess randomness.

Example NIST-style tests

- **Frequency test**: checks whether 0s and 1s appear equally often.
- **Runs test**: checks the number and length of streaks.
- **Longest run of ones**: looks for unusually long clusters.
- **FFT test**: looks for periodic structure in the sequence.
- **Approximate entropy**: measures local pattern complexity.

Why this matters

A stream can look balanced overall but still fail because of local patterns or repeated structure.

Rabbit vs. NIST

Aspect	Rabbit / NIST
Goal	Quick screening vs. bit-level validation
Scope	Broader generator diagnostics vs. binary stream tests
Cost	Usually lighter than the biggest TestU01 batteries vs. moderate
Best use	Early detection of obvious flaws vs. validating bit sequences
Limitations	Not a proof of security or true randomness
Also not a proof; p-values can mislead	

How to interpret failures

- A single surprising p-value can happen by chance.
- Repeated failures across related tests are a red flag.
- Very strong batteries can expose weaknesses that weaker tests miss.
- Passing a battery does not make a generator cryptographically secure.

Important caution

Statistical randomness is not the same as unpredictability against an adversary.

A practical workflow

1. Generate a large output sample from the RNG.
2. Run a quick screen such as Rabbit.
3. If the generator looks promising, move to stronger testing.
4. Use NIST STS when evaluating bit streams or hardware sources.
5. Interpret results in context: application, sample size, and threat model matter.

Examples of RNGs that get tested

- Linear congruential generators.
- Mersenne Twister and related pseudorandom generators.
- Cryptographic DRBGs such as HMAC-DRBG or ChaCha20-based generators.
- Hardware entropy sources and mixed system RNGs.

Why this is useful

Even good-looking output can hide weaknesses that only a battery of tests reveals.

What students should remember

- Rabbit is a faster screening battery.
- NIST STS is a classic bit-stream test suite.
- Both are statistical tools, not proofs.
- Strong RNG design still depends on theory, implementation, and threat model.

Summary

- TestU01 provides a framework for serious RNG testing.
- Rabbit helps catch obvious flaws quickly.
- NIST STS evaluates binary sequences with a well-known standardized suite.
- Passing tests is necessary for confidence, but not sufficient for security.

Questions?