A remark on NIST SP 800-22 serial test

Corina-Elena Bogos[*], Razvan Mocanu[†], Emil Simion [‡]

February 14, 2022

Abstract

This paper represents a cumulative review of the serial statistical test over the canonical values used in testing and freely generated values. Also in this paper, we study by simulation, the variation of second type error, depending on certain factors: the range of p1,the length of the bit string represented by n and the m-bit pattern.

Keywords: pseudo-random generators, serial test, second type error.

1 Introduction

1.1 Serial Test

The focus of this test is the frequency of all possible overlapping m-bit patterns across the entire sequence. The purpose of this test is to determine whether the number of occurrences of the 2^m m-bit overlapping patterns is approximately the same as would be expected for a random sequence. Random sequences have uniformity; that is, every m-bit pattern has the same chance of appearing as every other m-bit pattern.

1.2 Pseudo-random Generators

Given an initial seed, a PRNG produces a sequence of bits indistinguishable from a sequence produced by a real random source.
Indistinguishable means that there is no algorithm executable in polynomial time on a probabilistic Turing machine that can decide if the given sequence is random or calculated. That is, no randomised algorithm can say if a string produced by a PRNG was calculated deterministically or extracted by a random source. Hence, here it is a definition of PRNG: it's an algorithm executable in polynomial time on a deterministic Turing Machine that calculates a function G such that:

[*]Faculty of Computer Science, Alexandru Ioan Cuza University of Iasi, Email:corina.iftinca.v@gmail.com

[†]Faculty of Computer Science, Alexandru Ioan Cuza University of Iasi, Email: mocanuraz-van123@gmail.com

[‡]Politehnica University of Bucharest, Email: emil.simion@upb.ro

$$G{:}\{0,1\}^k \longrightarrow \{0,1\}^{l(k)}$$

with l as a monotonically increasing function. That means the output is always longer than the input (seed).
Truly random numbers are unpredictable in advance and must be produced by a random physical process, such as radioactive decay. Series of such numbers are available on magnetic tape or published in books, but they are extremely clumsy to use and are generally insufficient in both number and accuracy for serious calculations.

2 Math Background

A null hypothesis is a type of hypothesis used in statistics that proposes that there is no difference between certain characteristics of a population.

2.1 Null Hypothesis (First type error)

First type error (also known as the significance level), which is the probability of rejecting the null hypothesis when it is true: $\alpha = \mathrm{P}(\text{reject } H0|H0 \text{ is true})$. In the case of the serial test the p-values are calculated according to this formula:

$$\text{P-value1} = \text{igmac}(2^{m-2}, \tfrac{\nabla\psi_m^2}{2})$$
$$\text{P-value2} = \text{igmac}(2^{m-3}, \tfrac{\nabla^2\psi_m^2}{2})$$

Where igmac represents the regularized upper incomplete gamma function, defined as:

$$\Gamma(x,s) = \int_{\text{inf}}^{x} t^{s-1}e^{-t}\,dt,$$

s is a complex parameter, such that the real part of s is positive.

2.2 Null Hypothesis (Second type error)

Second type error represents the probability of failing to reject the null hypothesis when it is false: $\beta = \mathrm{P}(\text{accept } H0|H0 \text{ is false})$.
The (generalized) serial test represents a battery of procedures based on testing the uniformity of distributions of patterns of given lengths. Specifically, for $i_1, \ldots, i_m$ running through the set of all 2^m possible 0,1 vectors of length m, let $\nu_{i1\ldots i_m}$ denote the frequency of the pattern $(i_1, \ldots, i_m)$ in the "circularized" string of bits $(\epsilon_1, \ldots, \epsilon_n, \epsilon_1, \ldots, \epsilon_{m-1})$.

$$\psi_m^2 = \tfrac{2^m}{n}\textstyle\sum_{i_1,\ldots,i_m}(v_{i_1,\ldots,i_m} - \tfrac{n}{2^m})^2 = \tfrac{2^m}{n}\sum_{i_1,\ldots,i_m} v_{i_1,\ldots,i_m}^2 - n$$

Thus, ψ_m^2 is a χ^2 -type statistic, but it is a common mistake to assume that ψ_m^2 has the χ^2 -distribution. Indeed, the frequencies $\nu_{i_1\ldots i_m}$ are not independent.
For the second type error, the serial test with the parameter m, verifies the uniformity of distributions of patterns of given length m.
The input size recommended in [nis21] should verify m < $[log_2(n)]$ - 2.
Regarding the second error value, the results for the serial test from [GS17] are:

$$\beta(p1) \approx \chi^2((\tfrac{p_0}{p_1})^m\chi^2_{1-\alpha}(2^m-1) + \tfrac{n}{m}\tfrac{(p_0^m-p_1^m)^2}{(p_0^m p_1^m)}, 2^m-1)$$

3 Implementation

3.1 Serial test implementation

For running the experiments we used our own implementation[Bog22], explained in [SLL+10] that had as a starting point the steps described in NIST. Because we wanted to benefit from the multiple libraries given by Python, we thought it was the perfect choice.
In the next lines we will make a short description for the main functions used in our script.

- sub_strings() extracts the substrings of a given length from a string;
- dict_sequence() makes a dictionary having as a key a string and as a value the number of appearance;
- psi() for calculating the psi function;
- serial_test() the combination of the functions described above having as a result the serial test itself.

3.2 Second type error implementation

Regarding the tests for the second type error we transposed the code, presented in [GS17], from Matlab into Python. The second error β :

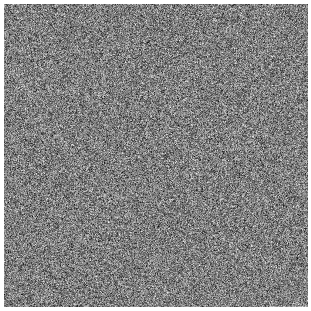
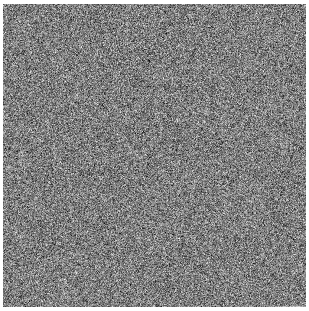
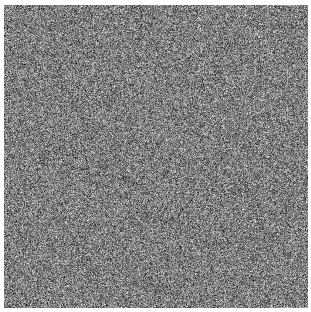
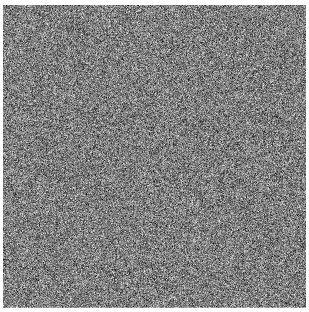
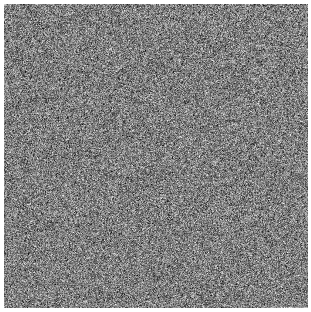
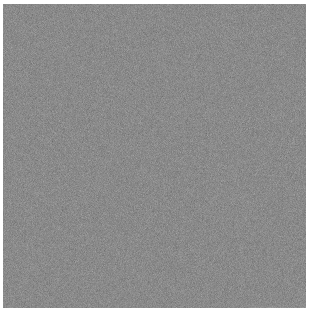
- 0.21062271194733564 for data.e
- 0.07112411541676428 for data.pi
- 0.01535927966491096 for data.sha1
- 0.02175406943058905 for data.sqrt2
- 0.0005832720701033665 for data.sqrt3

We can conclude that the probability for false negatives is very small for the serial test over the batch test

4 Experiments and Results

4.1 Batch test

For the batch test we have tested our algorithm over the following parameters and examples:

- 
  data.e is a random sequence
  with p1=0.8749140550999196 and p2=0.6109207198026381
- 
  data.sqrt3 is a random sequence
  with with p1=0.4037303479165052 and p2=0.22856489778962388
- 
  data.pi is a random sequence
  with p1=0.6372227521869827 and p2=0.4213740310465458
- 
  data.sha1 is a random sequence
  with p1=0.521894405425432 and p2=0.30962938652749694
- 
  data.sqrt2 is a random sequence
  with p1=0.5421009046833856 and p2=0.2790923129134414
- 
  data.bad_rng is not a random sequence
  with p1=0.0 and p2=0.0

4.2 Random functions testing

We have used the serial test for finding out about the qualities of several random functions from python and compared them to a true random source from random.org.

- randint from the random library
- randbelow from the secrets library
- urandom from the os library
- samples from random.org

Here are the tables that shows the qualities of the random tests done over 10 batches of 1000 generated numbers over 1, 1000000 range in python.

random_alg_gen	mean	standard deviation	range	mode	median
random_classic	96.7000	0.2160	0.6000	96.5000	96.6500
randbelow	96.4399	0.6449	2.2000	95.8999	96.5000
os_urandom	95.5099	0.4976	1.4000	96.0000	95.3500
random_org_request	98.4799	0.2573	0.8000	98.6000	98.5000

random_alg_gen	mean	standard deviation	range	mode	median
random_classic	96.5800	0.4685	1.5000	96.8999	96.6000
randbelow	96.8000	0.3590	1.0999	96.7000	96.7000
os_urandom	95.2500	0.6704	2.0000	94.8999	95.1499
random_org_request	98.5700	0.3591	1.3000	98.6000	98.5500

4.3 Second type error tests

In this section we study the variation of the second type error β with respect to *p1* and the length n of the bit stream. The figures presented below, generated using the scripts from [Bog22], compare the different values of β obtained in the following cases:

- n = 150, p1 in the range [0.3,0.7], m = 2;
- n = 200, p1 in the range [0.49,0.51], m taking the values 2,3,4;
- n = 5300, p1 in the range [0.48,0.52], m = 8;
- n in [5200,7700] and p1 in [0.48,0.52].

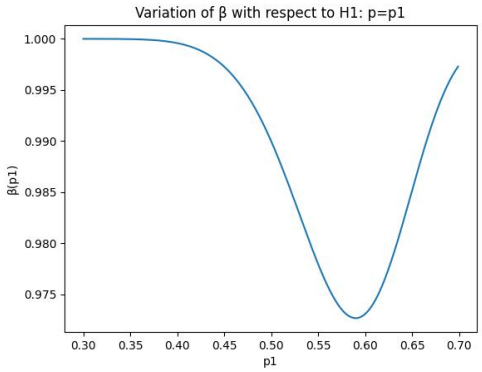


Figure 1: n=150.

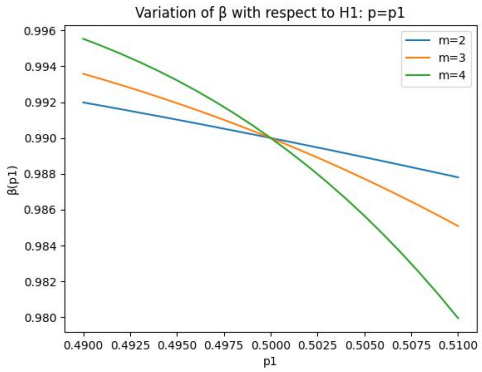


Figure 2: n=200.

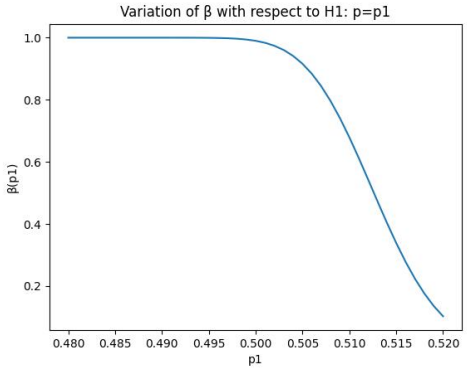


Figure 3: n=5300.

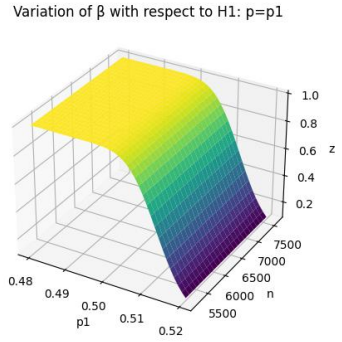


Figure 4: $5200 \leq n \leq 7700$.

5 Conclusion

In conclusion the serial test is reliable and robust statistical test that can help evaluate pseudorandom generators, but it is advisable to use multiple NIST statistical tests to have a higher amount of certainty about the randomness of an algorithm.

References

[Bog22] Corina Bogos. Nist serial tests. https://github.com/corina1906/nist_serial_test, 2022.

[GS17] Carmina Georgescu and Emil Simion. New results concerning the power of nist randomness tests. *Proceedings of The Romanian Academy, Series A*, 18:381–388, 2017.

[nis21] NIST standards. :http://www.nist.gov/,http://www.csrc.nist.gov/., 2021.

[SLL+10] Elaine Barker Smid, Stefan Leigh, Mark Levenson, Mark Vangel, AlanHeckert DavidBanks, and SanVo JamesDray. A statistical test suite for random and pseudorandom number generators for cryptographic applications. *Her research interest includes Computer security, secure operating systems, Access control, Distributed systems, Intrusion detection systems*, 2010.