Dhaka Univ. J. Sci. 67(2): 91- 98, 2019 (July)

A Comparison Study of Finding Efficient Methods for Generating Normal Random Numbers

Anamul Haque Sajib and Syeda Fateha Akter

Department of Statistics, Dhaka University, Dhaka-1000, Bangladesh

(Received: 11 February 2019; Accepted: 23 June 2019)

Abstract

Normal distribution is one of the most commonly used non-uniform distributions in applications involving simulations. Advanced computing facilities make the simulation tasks simple but the challenge is to meet the increasingly stringent requirements on the statistical quality of the generated samples. In this paper, we examine performances of different existing methods available to generate random samples from normal distribution based on statistical quality of the generated samples (randomness and normality) and computational complexities. From the simulation study, it is observed that CDF approximation based method and acceptance-rejection method devised by Rao et al.[12] and Sigman[14] are the fastest and the slowest respectively among all algorithms considered in this paper while generated samples produced by all methods satisfy randomness and normality properties. An application involving simulation from normal distribution is shown by considering a Monte Carlo integration problem.

Keywords: Normal distribution, Monte Carlo integration, Ljung-Box test, Bootstrapping.

I. Introduction

Normal random variables are widely used in Statistics, Computer Science, and other disciplines for several purposes. One of the main common purposes across the different disciplines is to carry out statistical inference of certain population parameter, which includes evaluating performances of estimation and test procedures by conducting a simulation study. In particular, normal random variables are required in certain application in Bayesian statistics, such as Monte Carlo method[12]. For example, suppose we wish to evaluate the integral $I = \int_{-\infty}^{\infty} \frac{\theta}{1+\theta} \mathrm{e}^{-0.5(\theta-5)^2} d\theta$ whose analytical solution is impossible.This type of computational problem is very common in both the frequentist and the Bayesian approaches. The Monte Carlo is one of the possible methods to approximate I by $\hat{I} = n^{-1}(2\pi)^{0.5} \sum_{i=1}^{n} \frac{\theta_i}{1+\theta_i}$, where $\theta_i, i = 1, 2, \cdots\cdots, n$, are generated sample produced by any MCMC scheme is more correlated compare to that of sample produced by any acceptance-rejection based sampling technique.

There are several algorithms available in the literatures which can be used to generate normal random numbers but all are scattered in the literatures. To the best of our knowledge, there are two comparative studies to date regarding finding an efficient method for generating normal random numbers conducted by Kundu, Gupta and Manglick[8] and Rao, Boiroju and Reddy[12], respectively. In both of the studies, the efficiency comparison over different methods for generating normal random numbers was measured based on the properties of randomness and normality. However, the computing time to generate samples for each method was ignored in both of their comparison studies, but computing time is one of the core components to decide how good analgorithm is in modern computational statistics. Therefore, in addition to check randomness and normality properties of the generated samples during the comparison of different methods, there is a need to incorporate computing time as well.

Motivated by the computational aspects of an algorithm, our aims are to conduct a study by covering all the existing methods for generating normal random numbers, and compare their performances based on the criteria of randomness, normality and computational time.

The rest of the paper is organized as follows: Section 2 presents the description of normal distribution and some relevant terminologies used in this paper such as Monte Carlo method, Ljung-Box test and Bootstrapping while each of the methods is discussed in section 3. Section 4 presents the results and the discussion which will be followed by an application and future work presented in sections 5 and 6, respectively.

II. Normal Distribution and Related Terminologies

Normal distribution, also known as Gaussian distribution, is one of the most widely used distributions in Statistics. The probability density function of a normal distribution is defined as

$$f(x;\ \mu, \sigma^2) = (2\pi\sigma^2)^{-0.5} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2};\ -\infty < x < \infty$$

where $\mu, -\infty < \mu < \infty$, and $\sigma > 0$ are the location (mean) and scale (variance) parameters of this distribution respectively. When $\mu = 0$ and $\sigma = 1$, this distribution is called the standard normal distribution which has some characteristics: (i) Symmetric about $\mu = 0$ and mean = median =mode =0 (ii) Skewness and Kurtosis are $\beta_1 = 0$ and $\beta_2 = 3$ respectively (iii) First and third quartiles are -0.6745 and 0.6745 respectively while second quartile is the median. A normal random variable (rv) X with mean μ and variance σ^2 can be obtained from standard normal variable Z using the transformation $X = \mu + \sigma Z$.

Monte Carlo Method

To explain the Monte Carlo method, we don't cover its detailed theory. Instead we discuss here only how this

*Author for correspondence. e-mail: sajibstat@yahoo.co.uk

method works for a particular problem by considering a suitable example, which is the main aim in the context of this paper. Monte Carlo method is a technique that can be used to solve a mathematical or statistical problem, whose analytical solution is impossible or quite challenging to compute, by using the stochastic simulation[13]. In the introduction section, we have seen that analytical solution of I is impossible and Monte Carlo method can be used to approximate I. Monte Carlo method approximates the integral I in a three steps procedures: (i) Express I in the form of an expectation of some function $h(\theta)$ of a random variable θ with a probability density function (pdf) $p(\theta)$ as $I = \int_{-\infty}^{\infty} h(\theta)p(\theta)d\theta = \int_{-\infty}^{\infty} \left[\sqrt{2\pi}\frac{\theta}{1+\theta}\right]\left[\frac{1}{\sqrt{2\pi}}\mathrm{e}^{-0.5(\theta-5)^2}\right]d\theta$, where the function $h(\theta) = \left[\sqrt{2\pi}\frac{\theta}{1+\theta}\right]$ and $p(\theta) = \left[\frac{1}{\sqrt{2\pi}}\mathrm{e}^{-0.5(\theta-5)^2}\right]$ (ii) Simulate a sample $\theta_1, \theta_2, \cdots\cdots, \theta_n$ from the pdf $p(\theta)$ which is $N(5,1)$ in this problem (iii) Finally the Monte Carlo estimator $\hat{I}$ of I is $n^{-1}\sum_{i=1}^{n} h(\theta_i)$ which is unbiased and converges to the true integral almost surely[11] (proofs are omitted here).

Ljung-Box Test

Ljung-Box test is widely used in Econometrics and other applications of time series analysis to test the randomness of a time series. Ljung and Box[9] jointly developed this test which works by following the three steps: (i) H_0 : the data are independently distributed against H_a : the data possess some serial correlation up to a certain lag h (ii) The quantity $Q = n(n+1)\sum_{k=1}^{h}[(n-k)^{-1}r_k^2]$, , which is a function of sample autocorrelation r at lag k and sample size n, denotes the test statistic (iii) $Q \sim \chi^2_{(h)}$ under H_0 and reject the null hypothesis if $Q > \chi^2_{(1-\alpha,h)}$ where $\chi^2_{(1-\alpha,h)}$ is the $(1-\alpha)^{th}$ quintile of the χ^2 distribution with h degrees of freedom.

Bootstrapping

In Statistics, inference about certain population characteristics can be drawn based on observed data (sample). For example, inference of the parameters (μ, σ^2) of normal distribution can be made using the sample mean $(\overline{X})$ and sample variance (S^2), respectively. After making inference regarding unknown population characteristics through estimators, which are a function of sample data, there is a need to evaluate the accuracy of the estimators. The accuracy of the estimators can be evaluated in terms of bias and variance of the estimators along with their confidence interval (C.I.). Sometimes, such measures of accuracy can be derived analytically but often require to estimate them numerically.

Bootstrap is such a technique that can be used to estimate their accuracy numerically from a single data set which is originally proposed by Efron and Tibishirani[5]. Usually, the true error in a sample statistic against its population value is not measurable as population is unknown. The bootstrap method alleviates this problem by resampling the sample data and performing inference about a sample from resampled data. In this case, the sample data plays the role as population which is known and hence the quality of inference of the true sample from resampled data is measurable. There are two types of bootstrapping: parametric and non-parametric bootstrapping. In the light of this paper, we will only require parametric bootstrapping which works by assuming observed sample comes from a known probability distribution $p(x;\Theta)$ (the parameter Θ value is unknown). Using the observed sample parameter Θ is estimated by $\hat{\Theta}$ using any suitable method (preferably MLE) and then plugging in $p(x;\hat{\Theta})$. Finally, N samples are drawn from $p(x;\hat{\Theta})$ and estimate Θ from each of the samples. Suppose $\hat{\Theta}_i^*, i = 1,2,\cdots,N,$ are the estimates of Θ for each sample respectively then the empirical distribution of $\hat{\Theta}^* = (\hat{\Theta}_1^*, \hat{\Theta}_2^*, \cdots, \hat{\Theta}_N^*)$ gives the approximate distribution of $\hat{\Theta}$.

Bootstrap Estimates of Bias, Variance and C.I.

The theoretical bias and variance of an estimator $\hat{\Theta}$ against the parameter Θ are Bias $(\hat{\Theta}) = E(\hat{\Theta} - \Theta) = E(\hat{\Theta}) - \Theta$ and Var $(\hat{\Theta}) = E\{(\hat{\Theta} - \Theta)^2\}$, respectively. The bootstrap estimates of bias and variance can be obtained by replacing both Θ and $\hat{\Theta}$ by $\hat{\Theta}$ and $\hat{\Theta}^*$ in their respective expressions i.e. Bias $(\hat{\Theta}) \approx E(\hat{\Theta}^* - \hat{\Theta}) = \frac{1}{N}\sum_{i=1}^{N}\hat{\Theta}_i^* - \hat{\Theta}$ and Var $(\hat{\Theta}) \approx (N-1)^{-1}\sum_{i=1}^{N}\left\{\left(\hat{\Theta}_i^* - \hat{\Theta}^*\right)^2\right\}$. From the bootstrap distribution (approximate distribution) of true $\hat{\Theta}$, confidence interval can be constructed using several methods. The most commonly used methods are "basic", "percentile" and "normal" methods. As the percentile method works well in symmetrical case and we believe that the shape of the bootstrap distribution is symmetrical in the context of our problem, we will use "percentile" method to construct confidence interval in this paper[4,5]. According to them, the percentile confidence interval is $(\hat{\Theta}^*_{(\alpha/2)}, \hat{\Theta}^*_{(1-\alpha/2)})$, where $\hat{\Theta}^*_{(1-\alpha/2)}$ denotes the $(1-\alpha)^{th}$ percentile of the bootstrap distribution.

In this paper, the bootstrap confidence interval is used to test the hypothesis that the sample data comes from a standard normal distribution against the alternative that the data comes from a different distribution. When the bootstrap confidence interval for certain estimator contains its true parameter, we accept the null hypothesis that sample data comes from a standard normal distribution[1]. For example, if the bootstrap confidence interval for sample mean contains the true parameter $\mu = 0$ then we accept the null hypothesis.

Acceptance-Rejection Method

Suppose $f(x)$ and $F(x)$ be the pdf and cdf of a random variable X, respectively. Generating X from $f(x)$ using inversion method requires $F_X^{-1}(U)$, where $U \sim Uniform(0,1)$. The inversion method can't be applied to generate X from $f(x)$ if $F(x)$ is not invertible. Under this circumstances, other available methods such as acceptance-rejection method can be applied to generate X

from $f(x)$. In some situations, there may be other available methods which are more efficient than inversion method and in such cases it is recommended to use these alternative methods. Acceptance-rejection algorithm is a well-known data generation technique from an arbitrary probability density function. Under this framework, a suitable proposal density needs to be designed from which a candidate samples can be drawn. Suppose $g(y)$ be the proposal density which is chosen in such a way that $g(y)$ is very close to $f(x)$, and efficient data generation method is available for $g(y)$. A candidate value $Y$from $g(y)$ is accepted (considered to follow $f(x)$) under acceptance-rejection method if $U \leq \frac{f(y)}{Mg(y)}$, where $Sup_x \left\{\frac{f(y)}{g(y)}\right\} \leq M$ and $U \sim Uniform(0, 1)$. The detailed procedure of acceptance-rejection method to simulate a sample of size n is discussedin Algorithm 2.

Algorithm 1: Algorithm of parametric bootstrapping

Input: Observed data $x_1, x_2, \cdots \cdots x_n$ from $p(x; \Theta)$

Output:Approximate pdf of $\hat{\theta}$

Begin

- Estimate Θ by its MLE $\hat{\theta}$
- Fix bootstrap re-sample number, say N

For $i = 1, 2, \cdots \cdots, N$**do**

- Generate $x^* = (x_1^*, x_2^*, \cdots x_n^*)$ from $p(x; \hat{\theta})$
- Estimate Θ from x^* and denote it by $\hat{\theta}_i^*$

End For loop

- Empirical pdf of $\hat{\theta}_i^*$ mimics the true pdf of $\hat{\Theta}$

End Begin

Algorithm 2: Acceptance-Rejection algorithm

Input: Proposed value Y from $g(y)$

Output: Produce X from the target density $g(x)$

Begin

For $i = 1, 2, \cdots \cdots, n$ **do**

1. Generate Y from $g(y)$
2. Calculate $Sup_y \left\{\frac{f(y)}{g(y)}\right\} \leq M$
3. Generate $U \sim Uniform(0, 1)$
4. **If** $U \leq \frac{f(y)}{Mg(y)}$ **then**
   - $X = Y$

   Else
   - Go back to step 1

   End If

End For loop

- Return all $X_1, X_2 \cdots, X_n$ as a desired sample

End Begin

III. Methods for Generating Standard Normal Random Numbers

In this section, we discuss the most commonly used methods to generate standard normal random numbers. Extensive list of different available methods for generating standard normal random numbers is available in the text book written by Johnson, Kotz and Balakrishnan[7].

Central Limit Theorem (CLT) Based Approach

Generating standard normal random numbers using the idea of central limit theorem is simple. The idea behind this approach is to generate k independent identically distributed (i.i.d.) $U_1, U_2, \cdots, U_k \sim Uniform(0, 1)$ and define their sum by $S = \sum_{i=1}^{k} U_i$, which approximately follows $N\left(\frac{k}{2}, \frac{k}{12}\right)$. The more $Uniform(0, 1)$ variables are used to define S, the better the approximation. Finally, standardize S by $Z = \frac{S - {}^{k}/_{2}}{\sqrt{{}^{k}/_{12}}}$ which approximately follows $N(0, 1)$. Considering $k = 12$ in the right hand of Z simplifies the expression to $Z = S - 6$ although there is a question regarding the approximation which only takes only 12 $Uniform(0, 1)$ variables. The detailed procedure of CLT based approach to simulate n standard normal random numbers is discussed in Algorithm 3.

Algorithm 3: Algorithm of CLT method to generate Z

Input: $U_1, U_2, \cdots, U_k \sim Uniform(0, 1)$

Output: Produce $Z \sim N(0,1)$

Begin

For $i = 1, 2, \cdots \cdots, n$ **do**

- $U_1, U_2, \cdots, U_k \sim Uniform(0, 1)$
- $Z_i = \sum_{i=1}^{k} U_i - 6$

End For loop

- Return all $Z_1, Z_2 \cdots, Z_n$ as a desired sample

End Begin

Box-Muller Method

Box and Muller[3]proposed a method which generates two independent standard normal random numbers using two $Uniform(0, 1)$ variables. The detailed procedure of Box-Muller method to simulate n (even) standard normal random numbers is discussed in Algorithm 4.

Algorithm 4: Box-Muller[3] algorithm to generate Z

Input: $U_1, U_2 \sim Uniform(0, 1)$

Output: Produce $Z_1, Z_2 \sim N(0,1)$

Begin

For $i = 1, 2, \cdots \cdots, {}^{n}/_{2}$ **do**

- Generate $U_1, U_2 \sim Uniform(0, 1)$
- Calculate $Z_1 = \sqrt{-2\, logU_1} \cos(2\pi U_2)$

and $Z_2 = \sqrt{-2\, logU_1} \sin(2\pi U_2)$

End For loop

- Return all $Z_1, Z_2 \cdots, Z_n$ as a desired sample

End Begin

To generate n^* (odd) standard normal random numbers, run the above algorithm and discard one standard normal random number.

Polar Method

George Marsaglia[10] modified the Box-Muller method which is known as Polar method. Because of Marsaglia's modification, there is no need to use Sin and Cos function in Box-Muller method. The modified version of Box-Muller method to produce a sample of size n is summarized in Algorithm 5.

Algorithm 5: Polar[10] algorithm to generate Z

Input: $U_1, U_2 \sim Uniform(0,1)$
Output: Produce $Z_1, Z_2 \sim N(0,1)$

Begin
For $i = 1, 2, \cdots\cdots, {}^{n}/_{2}$ **do**
• Generate $U_1, U_2 \sim Uniform(0,1)$
• Set $V_1 = 2U_1 - 1$, $V_2 = 2U_2 - 1$ and $S = V_1^2 + V_2^2$
• **While** $S > 1$ **then**
• Repeat first two steps to calculate S again
• **End While**
• Calculate $Z_j = \sqrt{\frac{-2\log S}{S}} V_j, j = 1, 2$
End For loop
• Return all $Z_1, Z_2 \cdots, Z_n$ as a desired sample
End Begin

Hastings CDF Approximation Based Method

Hastings[6]developed an algorithm to generate a standard normal random numbers using the inversion method, where an approximation function of standard normal CDF is considered.

Algorithm 6: Hastings[6] algorithm to generate Z

Input: $U \sim Uniform(0,1)$, and the seven constants $a_0 = 2.515, a_1 = 0.8028, a_2 = 0.0103, b_0 = 1, b_1 = 1.432, b_2 = 0.189,\ b_3 = 0.001$
Output: Produce $Z \sim N(0,1)$

Begin
For $i = 1, 2, \cdots\cdots, n$ **do**
• Generate $U \sim Uniform(0,1)$
• $W = \sqrt{-2\ln(U)}$
If $U \leq 0.5$ **then**
• $Z = -W + \frac{\sum_{i=0}^{2} a_i w^i}{\sum_{j=0}^{3} b_j w^j} = \varphi^{-1}(U)$
Else
• $Z = -\varphi^{-1}(1 - U)$
End If
End For Loop
• Return all $Z_1, Z_2 \cdots, Z_n$ as a desired sample
End Begin

The detailed procedure of Hastings's method for generating a sample of size n is described in Algorithm 6.

Rao et al. CDF Approximation Based Method

Rao et al.[12] used the same idea like Hastings's CDF approximation method but they used logistic distribution to approximate the CDF of standard normal distribution. Equating $\varphi(Z) = (1 + e^{-1.702Z})^{-1} = U$ yields $Z = -(1.702)^{-1}\ln(u^{-1} - 1)$. The detailed procedure of Rao et. al[12] method is described in Algorithm 7.

Acceptance–Rejection Method

Exponential density with rate $\lambda = 1$, $g(x; \lambda) = e^{-x}, x \geq 0$, is chosen as a proposal density to generate standard normal variables by Sigman[14]. Considering exponential proposal density (rate $\lambda = 1$) actually produces samples from the distribution of $|Z|$ instead of Z, which has density $f(x) = 2(2\pi)^{-0.5}e^{-x^2/2}, x \geq 0$. Under this setting, the value of $\sup_x[f(x)/g(x)]$ is $M \leq \sqrt{2e/\pi} = 1.315$, and a proposed value is set to $|Z|$if $U \leq [f(x)/Mg(x)] = e^{-\frac{1}{2}(x-1)^2}$, where $U \sim Uniform(0,1)$. Finally, using the symmetry property of normal distribution $|Z|$ is transformed to Z, and this can be done by setting $Z = |Z|$ if $U \leq 0.5$ and $Z = -|Z|$ if $U \geq 0.5$, where $U \sim Uniform(0,1)$. As $M = 1.315$, so using $g(x; \lambda) = e^{-x}, x \geq 0$ proposal density requires on average 1.315 draws to generate one standard normal variate. Generating a sample of size n according to Sigman[14] method is summarized in Algorithm 8a.

Algorithm 7: Rao et al.[12] algorithm to generate Z

Input: $U \sim Uniform(0,1)$
Output: Produce $Z \sim N(0,1)$

Begin
For $i = 1, 2, \cdots\cdots, n$ **do**
• Generate $U \sim Uniform(0,1)$
• Calculate $Z = -(1.702)^{-1}\ln(U^{-1} - 1)$
End For loop
• Return all $Z_1, Z_2 \cdots, Z_n$ as a desired sample
End Begin

Algorithm 8a: Sigman[14] algorithm to generate Z

Input: Proposed value Y from $g(y)$
Output: Produce X from the target density $g(x)$

Begin
For $i = 1, 2, \cdots\cdots, n$ **do**
1. Generate $Y \sim \exp(1)$
2. Find $sup_y\left\{\frac{2(2\pi)^{-0.5}e^{-y^2/2}}{e^{-y}}\right\} = \sqrt{\frac{2e}{\pi}} \leq c$
3. Generate $U \sim Uniform(0,1)$
4. **If** $U \leq \frac{f(y)}{cg(y)} = e^{-0.5(y-1)^2}$ **then**
• $|X| = Y$
• Generate $U \sim Uniform(0,1)$
• **If** $U \leq 0.5$ **then**
• $X = |X|$
• **Else**
• $X = -|X|$
End If
Else
• Go back to step 1
End If
End For loop
• Return all $X_1, X_2 \cdots, X_n$ as a desired sample
End Begin

Using the relation $-logU \sim \exp(1)$ where $U \sim Uniform(0,1)$, step 4 of Algorithm 8a can be simplified as follows: (i) generate two independent variables $Y_1, Y_2 \sim \exp(1)$ (ii) if $Y_2 \geq 0.5(Y_2 - 1)^2$ then set $|X| = Y_1$. The simplified version of Algorithm 8a is presented in Algorithm 8b.

Algorithm 8b: Sigman[14] algorithm to generate Z

Input: Proposed value Y from $g(y)$

Output: Produce X from the target density $g(x)$

Begin

For $i = 1, 2, \cdots\cdots, n$ **do**

1. Generate $Y \sim \exp(1)$
2. Find $sup_y \left\{ \frac{2(2\pi)^{-0.5} e^{-y^2/2}}{e^{-y}} \right\} = \sqrt{\frac{2e}{\pi}} \leq c$
3. Generate $U \sim Uniform(0,1)$
4. **If** $U \leq \frac{f(y)}{cg(y)} = e^{-0.5(y-1)^2}$ **then**
   - $|X| = Y$
   - Generate $U \sim Uniform(0,1)$
   - **If** $U \leq 0.5$ **then**
     - $X = |X|$
   - **Else**
     - $X = -|X|$
   - **End If**

Else

- Go back to step 1

End If

End For loop

- Return all $X_1, X_2 \cdots, X_n$ as a desired sample

End Begin

Using Generalized Exponential Distribution

Kundu et al[8]. proposed an algorithm to generate standard normal random numbers using two parameters generalized exponential (GE) distribution. The distribution function of GE density is $F(x;\, \alpha, \lambda) = \left(1 - e^{-\lambda x}\right)^{\alpha}$, where α and λ are the shape and scale parameters, respectively. Simulating a random variable X from GE density can be made using the inversion method as its distribution function is analytically invertible. They also observed that for a certain ranges of shape parameter of GE density the log-normal and GE are very close to each other i.e. one can be approximated by other. Furthermore, if X follows log-normal then $\log X$ is distributed as normal. Using these ideas, Kundu et al.[8] generated standard normal variables as follows: (i) Generate X from GE density using inversion method for $\alpha = 12.9$ and $\lambda = 1$ (ii) Transform X which is approximately distributed as log-normal to standard normal distribution. Algorithm 9 describes the Kundu et al.[8]method to generate a sample of size n.

Algorithm 9: Kundu et al.[8] algorithm to generate Z

Input: $U \sim Uniform(0,1)$

Output: Produce $Z \sim N(0,1)$

Begin

For $i = 1, 2, \cdots\cdots, n$**do**

- Generate $U \sim Uniform(0,1)$
- Calculate $X = -\log\left(1 - U^{1/12.9}\right)$
- Calculate $Z = \frac{\log X - 1.0821}{0.3807}$

End For loop

- Return all $Z_1, Z_2 \cdots, Z_n$ as a desired sample

End Begin

Bol'shev Method

Bol'shev[2] proposed a method to generate standard normal random numbers. The detailed procedures of Bol'shev method to generate a sample of size n is summarized in Algorithm 10

Boiruju et al. Method

Boiroju et al.[12] proposed an algorithm to generate standard normal random numbers. The detail procedures of Boiroju et al.[12] method to generate n standard normal variates is summarized in Algorithm 11.

Algorithm 10: Bol'shev[2] algorithm to generate Z

Input: $U \sim Uniform(0,1)$

Output: Produce $Z \sim N(0,1)$

Begin

For $i = 1, 2, \cdots\cdots, n$ **do**

- Generate $U_1, U_2, U_3, U_4, U_5 \sim Uniform(0,1)$
- Calculate $X = \frac{1}{5}\sum_{i=0}^{n}\left[\sqrt{3}(2U_i - 1)\right]$
- Calculate $Z = X - 0.01(3X - X^3)$

End For loop

- Return all $Z_1, Z_2 \cdots, Z_n$ as a desired sample

End Begin

Algorithm 11: Boiruju et al.[12] method to generate Z

Input: $U \sim Uniform(0,1)$

Output: Produce $Z \sim N(0,1)$

Begin

For $i = 1, 2, \cdots\cdots, n$ **do**

- Generate $U \sim Uniform(0,1)$
- Calculate :

$$X_1 = \tanh(-31.356 + 28.771U)$$
$$X_2 = \tanh(-2.571 - 31.163U)$$
$$X_3 = \tanh(3.949 - 1.668U)$$
$$X_4 = \tanh(2.312 + 1.842U)$$
$$Z = .466 + 90.721X_1 - 89.369X_2 - 96.554X_3 + 97.363X_4$$

End For loop

- Return all $Z_1, Z_2 \cdots, Z_n$ as a desired sample

End Begin

IV. Simulation Study and Discussions

In this section, we have presented the simulation results obtained under different methods considered in section 3. Here all numerical computations are computed in- R on a Samsung XI machine with an Intel (R) Core (TM) i7-4900 (single) processor running at 3.60 GHz. By using the CLT method, Box-Muller (BM) method, George Marsaglia (GM) method, Hastings CDF (H_CDF) based method, Rao et al. CDF (R_CDF) based method, Sigman accept-reject (S_AR) method, Kundu generalized exponential (K_GED) method, Bol'shev method (B_SHEV) and Boiroju methods (B_ROJU), a sample of size 1 million is generated from standard normal distribution. All the results presented here are produced using random seed number, and we have found that using different seed number produces similar kind of results. The efficiency of

each methods is determined based on statistical quality of thegenerated samples (randomness and normality properties) and computing time required to generate these samples. Table 1 shows the average computing (CPU) time required to generate a sample of size 1 million for each method.

Table 1. Average computing time required to generate a sample of size 1 million from standard normal distribution using different methods.

Method	Average Computing Time (in second)
CLT	0.840
BM	0.104
GM	0.557
H_CDF	0.119
R_CDF	**0.063**
S_AR	**0.965**
K_GED	0.167
B_SHEV	0.855
B_ROJU	0.335

We have computed the CPU time consumed to execute the whole program using the system.time command and noticed that consumed CPU time varies one run to another run by 1 to 2 percent. Although varied amount is not too high but we take into account it by reporting the average CPU time (50 replications). As computing time is one of the core components to measure the efficiency of each methods, an emphasized has been placed on coding as efficiently as possible. From Table 1, we see that Rao et al. CDF (R_CDF) based method is the fastest, which requires only 0.063 second, while Sigman accept-reject (S_AR) based method is the slowest (requires 0.965 second) among all methods considered here. In other word, R_CDF method is 15.31 times faster than S_AR method. It is also observed that average computing time for sample sizes $n = 100, 1000$ and 50000 reduced linearly relative to the time shown in Table 1 but not shown here. To test the randomness of the generated samples obtained under different methods, we have used both graphical technique (ACF plot) and Ljung-Box test. Figure 1 shows the ACF plots of generated samples obtained under CLT and BM methods respectively. From Figure 1, it is observed that some of the ACF at lag around 12, 32 and 63 for CLT method and at lag around 17, 41 and 80 for BM method are beyond the significance confidence bands (95%). However, it does not guarantee the presence of autocorrelation, and may happen because of sampling error. We haven't considered the ACF of generated samples obtained under other methods to plot here as they have similar patterns. Ljung-Box test is carried out to confirm the presence of autocorrelation and results are shown in Table 2.

Table 2. Ljung-Box test statistic and their corresponding *P* values in parenthesis at different lags.

Method	Lag 10	Lag 20	Lag 30	Lag 60	Lag 80
CLT	6.886 (0.736)	14.504 (0.804)	31.779 (0.378)	53.947 (0.695)	72.358 (0.716)
BM	3.673 (0.961)	13.383 (0.860)	21.002 (0.888)	60.914 (0.443)	87.619 (0.262)
GM	4.573 (0.918)	13.987 (0.831)	24.468 (0.750)	43.669 (0.944)	76.687 (0.584)
H_CDF	11.935 (0.289)	15.659 (0.738)	32.770 (0.333)	65.338 (0.297)	99.489 (0.069)
R_CDF	11.933 (0.289)	23.809 (0.251)	36.048 (0.207)	74.259 (0.102)	85.307 (0.322)
S_AR	10.587 (0.391)	15.477 (0.748)	22.597 (0.831)	48.590 (0.854)	78.791 (0.517)
K_GED	10.534 (0.395)	25.362 (0.188)	37.274 (0.169)	62.315 (0.394)	84.239 (0.351)
B_shev	4.871 (0.899)	13.617 (0.849)	17.826 (0.961)	36.911 (0.992)	57.264 (0.974)
B_roju	6.834 (0.741)	14.953 (0.779)	27.217 (0.612)	44.622 (0.931)	54.353 (0.988)

From Table 2, we observe that the *P* Values of Ljung-Box statistic at different lags under all methods considered here are greater than $\alpha = 0.05$, which support the null hypothesis of randomness of generated samples obtained under all methods. Bootstrap confidence intervals for μ and σ for all methods are presented in Table 3, whereby we can conclude whether the generated samples satisfy the normality assumption. As true μ (0) and σ (1) are contained in their respective bootstrap intervals for all methods, generated samples produced by all methods satisfy the normality assumption. However, bootstrap confidence interval for σ under R_CDF and B_roju methods don't contain exactly 1 but both the limits are very close to 1.

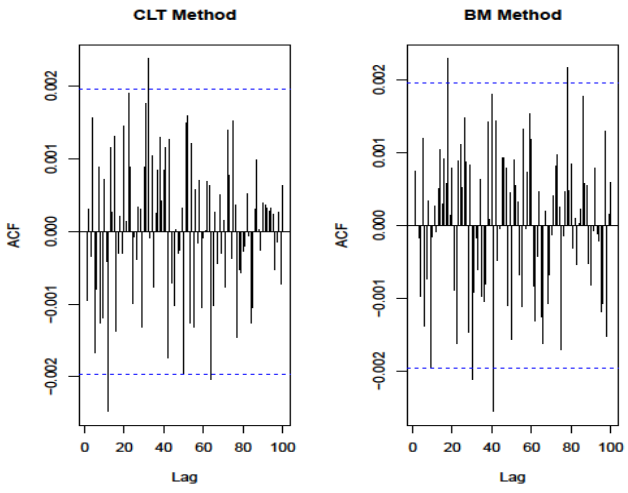

Fig. 1. ACF plots of generated samples

Table 3. Bootstrap confidence interval for μ and σ.

Method	Parameter	Statistics	Std. Error	95% CI	
				Lower	Upper
CLT	Mean	0.0009	0.00099	-0.0009	0.0029
	SD	1.0004	0.00074	0.999	1.0020
BM	Mean	-0.0003	0.00106	-0.0023	0.0018
	SD	1.0003	0.00072	0.999	1.0020
GM	Mean	-0.0017	0.00103	-0.0038	0.0003
	SD	1.0001	0.00071	0.999	1.0020
H_CDF	Mean	0.0000	0.00104	-0.0021	0.0020
	SD	1.0001	0.00069	0.999	1.0020
S_AR	Mean	0.0000	0.00100	-0.0021	0.0020
	SD	1.0011	0.00069	1.000	1.0030
K_GED	Mean	-0.0004	0.00099	-0.0024	0.0016
	SD	1.0020	0.00073	1.001	1.0040
R_CDF	Mean	0.0010	0.00103	-0.0010	0.0031
	SD	1.0656	0.00074	1.064	1.0670
B_shv	Mean	0.0012	0.00092	-0.0006	0.0029
	SD	0.9995	0.00070	0.9981	1.0009
B_roju	Mean	0.0009	0.00097	-0.0009	0.0029
	SD	0.9972	0.00069	0.9958	0.9985

Finally, from the above results, it can be concluded that all the methods can be used to generate normal random numbers as they satisfy normality and randomness properties. When generating normal random numbers is required subject to the constraints of statistical quality and required computing time then R_CDF is the best method (this is the fastest among all the methods considered in this paper).

V. A Bayesian Application

Suppose we have the observed data $\boldsymbol{X} = (X_1, X_2, \cdots, X_n)$ from some population with unknown d dimensional parameter $\Theta = (\Theta_1, \Theta_2, \cdots \Theta_d)$, and our aim is to know about the unknown parameter Θ based on this observed sample $\boldsymbol{X}$. In the frequentist approach, the unknown parameter Θ is considered as fixed quantity and can be directly estimated using the likelihood function. On the other hand, in the Bayesian approach, Θ is considered as a random variable which has a probability distribution $\pi(\Theta)$, known as prior distribution in Bayesian literatures. Using Bayes' theorem both the information from observed data and prior distribution are combined, resulting in the following posterior distribution $\pi(\Theta|\boldsymbol{X})$, which is the basis for any inference regarding Θ:

$$\pi(\Theta|\boldsymbol{X}) = \frac{\pi(\boldsymbol{x}|\Theta)\pi(\Theta)}{\int_{-\infty}^{\infty} \pi(\boldsymbol{x}|\Theta)\pi(\Theta)\, d\Theta}.$$

Suppose $X|\Theta \sim N(\Theta, 1)$ and $\Theta \sim$Cauchy (0, 1). To keep the problem simpler, we consider only one observation in our sample $(n = 1)$ and Θ is one dimensional. Then the posterior distribution is

$$\pi(\Theta|X) = \frac{\frac{1}{1+\Theta^2} e^{-0.5(\Theta - x)^2}}{\int_{-\infty}^{\infty} \frac{1}{1+\Theta^2} e^{-0.5\,(\Theta - x)^2} d\Theta}.$$

Suppose we want to find the posterior mean $E(\Theta|X)$, known as Bayes' estimator of Θ, which can be defined as

$$E(\Theta|X) = \frac{\int_{-\infty}^{\infty} \frac{\Theta}{1+\Theta^2} e^{-0.5(\Theta - x)^2}}{\int_{-\infty}^{\infty} \frac{1}{1+\Theta^2} e^{-0.5\,(\Theta - x)^2} d\Theta}.$$

To the best of our knowledge, an analytical solution of the Bayes' estimator is impossible to find as there is no close-form expression of this Bayes' estimator. Using Monte Carlo integration, discussed in section 2, we can estimate $E(\Theta|X)$ by $\hat{I} = \sum_{i=1}^{k} \{\Theta_i (1 + \Theta_i^2)^{-1}\} / \sum_{i=1}^{k} (1 + \Theta_i^2)^{-1}\}$, where $\Theta_1, \Theta_2, \cdots \Theta_n$ are the random sample generated from $N(x, 1)$ using any method considered in this paper (preferably R_CDF). Considering $x = 5$ and $k = 10000$, we have $\hat{I} = 0.2078$. Computing time for calculating $\hat{I}$ under each method will be the time shown in the second column of Table 1 plus a constant h respectively, where h is the computing time required to execute average of the generated sample.

VI. Conclusion and Future Works

This paper compares performances of different existing methods available to generate normal random numbers, where performances are measured based on statistical quality of the generated samples and required computing time. From our simulation study, we have seen that R_CDF is the best method when both the conditions, statistical quality and computing time, are required to meet but reader can use any method considered in this paper when computing time is not any issue. As a future study one can compare performances of different available methods for generating multivariate normal random numbers along with a comparison study for generating other important univariate random variable (say, exponential variable).

References

1. Becher, H., P., Hall, S. R. Wilson 1993. Bootstrap hypothesis testing procedures. *Biometrics*. **49 (4)**, 1268-1272.
2. Bol'shev, L. N. 1959. On Transformations of Random Variables. *Theory of Probability and Its Applications.* **4 (2)**, 129-141.
3. Box, G. E. P. and M. E. Muller, 1958. A Note on the Generation of Random Normal Deviates. *Annals of Mathematical Statistics*. **29**, 610-611.
4. Davison, C., Anthony. and D. Hinkley, 1997. Bootstrap Methods and Their Application. Cambridge University Press, Cambridge.
5. Efron, B., R. J. Tibshirani 1993. An Introduction to the Bootstrap. *Chapman and Hall, New York.*
6. Hastings, C. 1955. Approximations for Digital Computers. *Princeton University Press, Princeton, N. J.*
7. Johnson, N., S., Kotz, and N. Balakrishnan, 1995. Continuous Univariate Distribution. *John Willey and Sons, New York.*
8. Kundu, D., Gupta, R. D. and Manglick, A. 2006. A Convenient Way of Generating Normal Random Variables

Using Generalized Exponential Distribution. *Journal of Modern Applied Statistical Methods*. **5**, 266–272.

9. G. M. Ljung, and , G. E. P. Box1978. On a Measure of Lack of Fit in Time Series Models. *Biometrika.* **65**, 297–303.

10. Marsaglia, G., T. A. Bray, 1964. A Convenient Method for Generating Normal Variables. *Society for Industrial and Applied Mathematics* Review. **6 (3)**, 260–264.

11. Press, W. H., S. A., Teukolsky, W. T., Vetterling and Flannery B. P. 1996. Numerical Recipes in Fortran 77: The Art of Scientific Computing. Fortran Numerical Recipes. 1 (Second ed.). Cambridge University Press. ISBN 978-0-521-43064-7.

12. Rao, K. R., N. K. Boiroju, and M. K. Reddy, 2011. Generation of Standard Normal Random Variables. *Indian Journal of Scientific Research.* **2 (4)**, 83-85.

13 Sawilowsky, S. S., G. C. Fahoome, 2003. Statistics via Monte Carlo Simulation with Fortran. *Journal of Modern Applied Statistical Methods*, Rochester Hills. ISBN 0-9740236-0-4.

14. Sigman, K. 2007. Simulating Normal (Gaussian) rvs with Applications to Simulating Brownian Motion and Geometric Brownian Motion in One and Two Dimensions. www.columbia.edu/~ks20/...Sigman/4703- 07-Notes- BM-GBM-I.pdf