International Research Journal of
Makawanpur Multiple Campus (IRJMMC)

Vol- 5, No. 1, March 2024 ISSN 2717-4999 (Online) 2717-4980 (Print)

A Study of Linear Model Fitting and Distribution Analysis on Random Numbers Generated in Fortran

RAJENDRA NEUPANE | KRISHNA BAHADUR RAI

*Author affiliations can be found in the back matter of this article

CORRESPONDING AUTHOR
Krishna Bahadur Rai
Department of Physics, Patan Multiple Campus
krishnarai135@gmail.com

KEYWORDS

Fortran 90
Kernel density estimate
Linear model fitting
Pseudorandom numbers
Sample size

ABSTRACT

In this study, we employed Fortran 90 programming to generate pseudorandom numbers ranging from 0 to 1 at various sample sizes (n = 1000, 500, 250, 125, 63, 32, 16, 8). Subsequently, linear fitting models were applied to the generated data. Our findings indicated that the higher sample size of n=1000 yielded the least asymptotic standard error for both slope 'a' and intercept 'b' parameters in the linear equation. We observed a decrease in the sum of squares of residuals as the sample size (n) decreased, indicating that the linear model also provided a better fit to the data with smaller sample sizes. The consistent nature of the kernel density estimate plots suggests that as the sample size increases, the estimation becomes more precise and less affected by random noise or sampling variability, further enhancing the reliability of the estimated PDF.

1. INTRODUCTION

Fortran, developed by International Business Machines (IBM) in 1954, introduced higher-level statements. The 1958 version added subroutines and functions. Fortran IV, released in 1962, aimed to standardize the language. Fortran 66 became the first American National Standards Institute (ANSI) standard in 1966. Fortran 77, released in 1978, introduced structured programming and new features. Fortran 90, the third ANSI standard, was released in 1991 (Davies, Rea, & Tsaptsinos, 2018). Random number generators can be classified into two categories: pseudo-random number generators (PRNGs) and true random number generators (TRNGs). The main distinction lies in the periodic nature of pseudo-random numbers compared to the non-periodic nature of true random numbers. PRNGs are deterministic algorithms, while TRNGs offer true randomness. The deterministic nature of PRNGs makes them desirable for various scientific applications, including simulations, statistical sampling, algorithm performance evaluation, and Monte Carlo simulations (Akhshani et al., 2014). Akhshani et al. (2014) studied Pseudo random number generator based on quantum chaotic. Leva (1992) proposed a method for generating pseudo-random numbers that follow a normal distribution (Leva, 1992). Having a machine-independent random number generator is desirable in various scenarios

where consistency is desired across different machines. It is particularly useful when a program written in a high-level language needs to generate identical results across machines given the same input (Schrage, 1979). Linear regression is a statistical technique used to model the relationship between two variables. It involves fitting a linear equation to observed data in order to estimate the best-fitting line (Yao & Li, 2014). The most common approach for finding this line is the method of least-squares, which aims to minimize the sum of the squared vertical deviations between each data point and the fitted line (Amiri-Simkooei & Jazaeri, 2012). By squaring and summing the deviations, both positive and negative differences contribute equally to the overall measure of goodness-of-fit, avoiding any cancellation effects. This allows for a comprehensive assessment of how well the linear equation aligns with the observed data points. In general, a smaller sum of squares of residuals indicates a better fit of the model to the data. A lower sum of squares of residuals indicates that the model is able to explain a larger proportion of the variability in the data, resulting in a smaller amount of unexplained variation. This can imply that the model is more accurate and reliable in predicting or explaining the observed data.

We have used different abbreviation for different parameters, the significance of which is described in subsection 1.1.

1.1 PARAMETERS USED IN THE ANALYSIS

Final Sum of Squared Residuals (FSSR) represents the sum of the squared differences between the predicted values and the actual values (residuals) in the final iteration of the linear model fitting (Amiri-Simkooei & Jazaeri, 2012). It gives an indication of how well the model fits the data, with a lower value indicating a better fit.

Relative Change in Linear Model Iteration (RCLI) indicates the relative change in the model's performance during the last iteration. It quantifies the difference in the model's output between the current and previous iterations, expressed as a proportion or percentage. A decreasing RCLI suggests that the model is converging and becoming more stable.

Root Mean Square of Residuals (RMSR) is the root mean square of the residuals, which provides an estimate of the average magnitude of the errors or residuals in the model (Browne, 1993). A lower RMSR value indicates a better fit, as it suggests that the model's predictions are closer to the actual data points.

Variance of Residuals' Norm (VRN) represents the variance of the residuals, which is often expressed as a reduced chi-square value. It measures the variability or dispersion of the residuals around the model's predictions. A lower VRN indicates a more precise model, as it suggests that the residuals are more closely distributed around the predicted values.

Residual variance estimation refers to the task of determining the most accurate estimate of the general- ization error that can be achieved using a limited set of data (Liitia¨inen et al., 2009). Cox and Snell (1968) have examined a regression problem that deals with errors following an exponential distribution (Cox & Snell, 1968). Kernel density estimate is a statistical technique used to estimate probability density functions, without making strong assumptions about the underlying data distribution (Altman, 1992).

We use Kernel density estimation (KDE) to visualize the underlying distribution of random numbers generated by a random number generator (RNG). It provides a smooth, non-parametric estimate of the probability density function (PDF) of the generated numbers (Guidoum, 2015). KDE facilitates distribution comparison, assesses uniformity, and analyzes the randomness properties of generated sequences (Wand & Jones, 1995). Comparing KDE plots allows for identification of parameter configurations that optimize the generated distribution, aiding in fine-tuning and optimizing RNG parameters.

In the context of the kernel density derivative estimator, as the sample size n increases, the noise (h) tends to decrease or approach zero (Bhattacharya, 1967;

Schuster, 1969; Alekseev, 1972)). Monte Carlo simulation heavily relies on random numbers to model and analyze complex systems or problems (Gauli et al., 2023).

We aimed to determine the optimal sample size that provides the most accurate linear fit for the random numbers generated between zero and one. We generate a sample of n random numbers uniformly distributed between 0 and 1 by using Fortran algorithm.

2. MATERIALS AND METHODS

In this study, we performed a linear model fitting using the least squares method on a dataset. The dataset was generated by random numbers ranging from zero to one, and different sample sizes, from n=1000, 500, 250, 125, 63, 32, 16, 8, were used. To accommodate the limitations of considering large sample sizes, we adopted a systematic approach in which each subsequent sample size was reduced by half, starting from n=1000. In each iteration, the resulting decimal obtained by halving the sample size is rounded to either 1 or 0. We determined the coefficients 'a' and 'b' of the linear model and reported their values. Additionally, we calculated the root mean square (RMS) of the residuals and the variance of residuals (chi-squared) to evaluate the goodness of fit.

By applying the linear model to the randomly generated numbers at each sample size, we calculated the sum of squares of residuals as a measure of the quality of the fit. We observed the behavior of the sum of squares of residuals for different sample sizes, aiming to identify the sample size that yields the lowest value. The sample size corresponding to the lowest sum of squares of residuals indicates the best linear fit model for the given dataset.

In subsection 2.1, a Fortran algorithm was utilized to produce a set of n (say 1000) random numbers distributed uniformly between 0 and 1 as outlined in this study.

2.1 FORTRAN ALGORITHM FOR GENERATING RANDOM NUMBERS BETWEEN 0 AND 1 AT SAMPLE SIZE n

```
program random_number_generator
 implicit none
 integer, parameter :: n = 1000
 real :: random_numbers(n)
 integer :: i

 ! Initialize the random number generator
 call random_seed()
      ! Generate random numbers between 0
and 1
 do i = 1, n
 call random_number(random_numbers(i))
 end do
      ! Print the generated random numbers
 do i = 1, n
 write(*, '(F6.4)') random_numbers(i)
 end do
end program random_number_generator
```

In this code, the random number subroutine from the Fortran standard library is used to generate random numbers between 0 and 1. The random_seed subroutine is called to initialize the random number generator before generating the random numbers. The generated random numbers are stored in the random_numbers array and then printed to the console using the write statement.

The algorithm utilized for generating random numbers between 0 and 1 at a sample size of n=1000 is presented in subsection 2.1.1. The algorithm for generating random numbers between 0 and 1 is modified by varying the sample size. Specifically, the algorithm is applied with different sample sizes of n=1000, 500, 250, 125, 63, 32, 16, and 8. These sample sizes are taken for convenience and flexibility in the analysis.

2.2 ALGORITHM FOR GENERATION OF DATA USING A LINEAR MODEL WITH NOISE

```
program generate_data
 use iso_fortran_env, only: real64
 implicit none
```

```
 integer, parameter :: num_points = 1000
 real(real64) :: x_values(num_points)
 real(real64) :: y_values(num_points)
 integer :: i
 ! Generate x_values
 do i = 1, num_points
 x_values(i) = i
 end do
 ! Generate y_values using a linear model
with noise
 do i = 1, num_points
 call random_number(y_values(i))
 y_values(i) = 2.0 * x_values(i) + 3.0 + 2.0 *
y_values(i) - 1.0
 end do
 ! Write data to a file
 open(unit=10, file='data.txt',
status='replace')
 do i = 1, num_points
 write(10, '(F12.8, A, F12.8)') x_values(i), ' ',
y_values(i)
 end do
 close(10)
end program generate_data
```

To generate the data points, we compile and run the programme by using Fortran code:

```
gfortran generate_data.f90 -o
generate_data
./generate_data
```

This will generate the data points file'data.txt' required for the fitting models. To display a plot using gnuplot, open the terminal and type "gnuplot". Then enter the necessary commands, execute them, and the resulting plot will be displayed.

This code generates num_points (set to 1000) of data points (x_values, y_values) using a linear model with added noise. It then writes the data points to a file named 'data.txt'.

The Gnuplot script is generated within the Fortran code. It fits the data to different models using the fit command in Gnuplot. The models include linear, quadratic, cubic, normal etc. But we use linear fitting model for our work. which is f(x) = a.x + b (Kra¨mer & Sonnberger, 2012). The code segment performs a linear fit of the data from the file 'data.txt' using the equation f(x) = a.x + b. The 'fit' command is utilized to determine the optimal values of 'a' and 'b' that minimize the squared differences between the observed data points and the corresponding values predicted by the linear model. The resulting fitted line is then plotted, displaying the linear fit alongside the data points.

"fit f(x) "data.txt" using 1:2 via a, b" is the fit commands, and
" replot f(x) with lines title "Linear Fit" is the command that plots the resulting fitted line

The script plots the data points as points and overlays the fitted models as lines.

In this work, we explore the distribution of the generated pseudorandom numbers at different sample sizes using kernel density estimation (KDE). Specifically, we examine whether the percentage density of each generated pseudorandom number between 0 and 1 in each sample size is identical or shows variations.

3. RESULT AND DISCUSSIONS

In Table 1, FSSR indicates the final sum of squares of residuals, RCLI indicate relative change during last iteration, RMSR indicates rms of residuals, VRN represents variance of residuals (reduced chisquare). Two parameters 'a' and 'b' represent the slope and the y-intercept respectively, in a linear model fitting.

The values of the sum of squares of residuals decrease as the sample size (n) decreases. This suggests that as the sample size becomes smaller, the model tends to provide a better fit to the data. All values of reduced chi-squared is significantly smaller than 1 which may suggest over-fitting or excessive model complexity. Here we have obtained for higher n (i.e., 1000), the

asymptotic standard error is least. Higher sample sizes lead to smaller values of the asymptotic standard error, indicating a higher level of precision and accuracy in the estimated parameters or statistics. As the sample size increases, the asymptotic standard error tends to approach zero. This suggests that larger sample sizes help minimize the uncertainty associated with the observed data, leading to more precise and accurate estimates of the parameters or statistics.

Table 1: *Summary of metrics and parameters for linear model fitting with varying sample size*

Sample size	FSSR	RCLI	RMSR	VRN ($\chi 2$)	a	b
1000	169.444	-1.96464×10−10	0.584483	0.34162	1.99996 ± 0.00018	2.98069 ±0.05246
500	164.159	-2.06753×10−10	0.587026	0.3446	2.00002±0.000183	3.04069 ±0.05269
250	69.9475	-1.16145×10−11	0.53108	0.282046	1.99925±0.0004654	3.07707±0.06738
125	41.452	-2.59692×10−13	0.580524	0.337008	2.00304±0.001439	2.81256±0.1045
63	18.3757	-6.72185×10−6	0.548854	0.30124	1.99871±0.003803	3.18157±0.14
32	9.98333	-4.26866×10−7	0.576869	0.332778	2.00496±0.01104	2.791±0.2088
16	3.57748	-7.72908×10−8	0.505504	0.255534	1.9725±0.02741	3.0631±0.2651
8	2.69452	-3.00252×10−9	0.670139	0.449086	2.0743±0.1034	2.46647 ±0.5222

Positive values of ‘a’ suggests a positive correlation, where an increase in ‘x’ leads to an increase in ‘y’. ‘b’ represents the y-intercept of the linear model. In the context of linear model fitting, a positive value of the y-intercept ‘b’ indicates that the dependent variable ‘y’ has a non-zero value even when the independent variable ‘x’ is zero. The linear curve fitting to the randomly generated pseudo-random number at sample sizes n =1000, 500, 250, 125, 63, 32, 16, 8 is shown in Figure 1. The plot 1(a) with a sample size of n=1000 shows a thousand pseudo-randomly generated points closely aligned and modeled accurately by the overlaid linear fit line. The subplot 1(b) lowering the sample size to 500 points has the data points beginning to slightly deviate and disperse around a linear model fit that starts displaying some decline in precision. At a sample size of 250 points in subplot 1(c), the pseudorandom data shows greater variability around an increasingly imprecise linear fit compared to higher sample sizes. With only 125 points in subplot 1(d), the points are loosely scattered around a moderately precise but noticeably under fitting linear model line. The linear fit in the subplot 1(e) with 63 points struggles to capture the variability exhibited by more erratically dispersed data points. A markedly imprecise linear model fit with low explanatory power is observable in the subplot 1(f) containing merely 32 highly scattered pseudorandom points. Extreme data point dispersion paired with an inaccurate linear fit unable to capture the variability characterizes the subplot's 1(g) 16-point sample. Lastly, an ineffectual linear model fit performed on just 8 wildly spread data points comprises subplot 1(h).

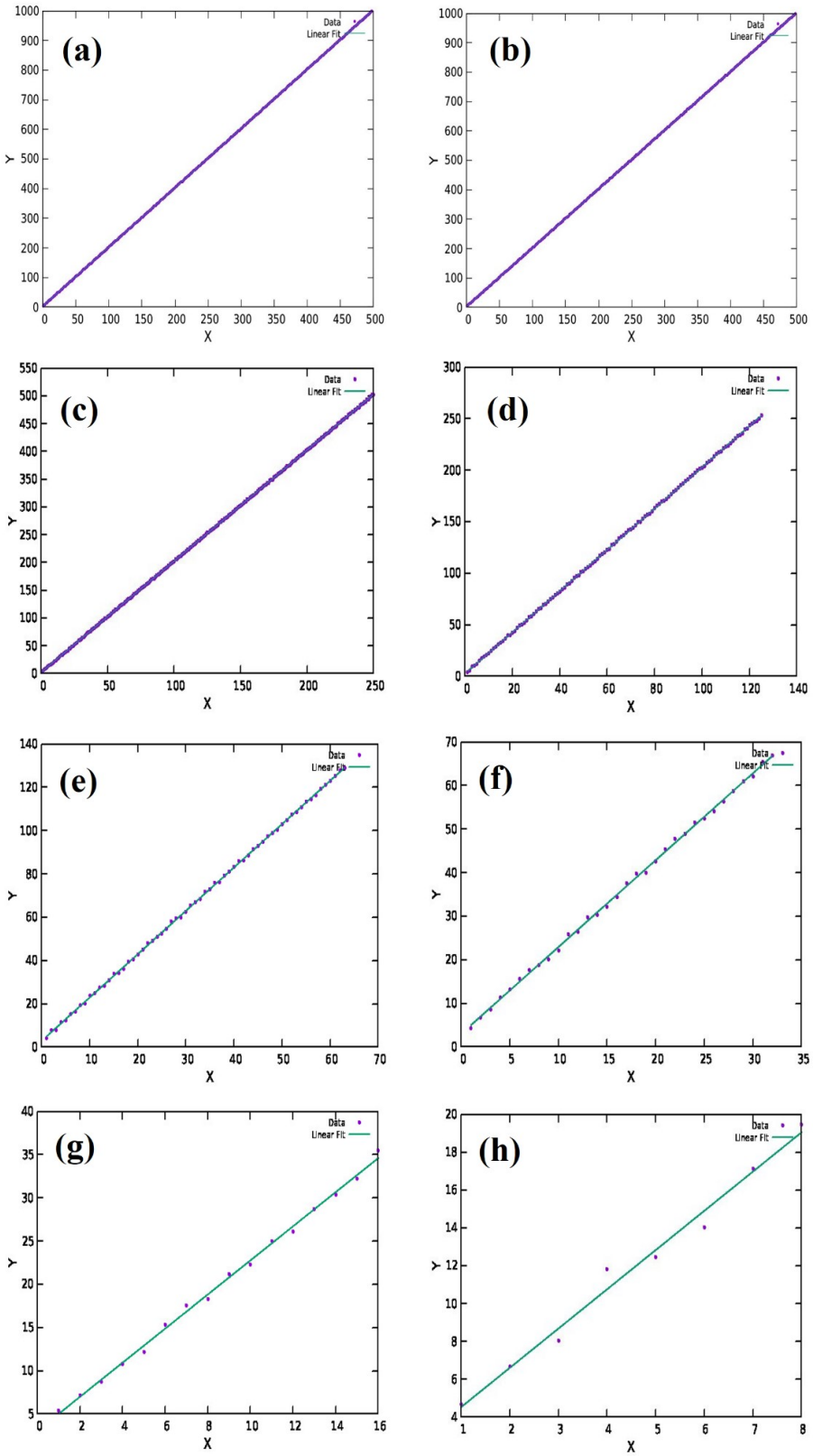

Figure 1: *Pseudo-random numbers for different sample sizes with linear fitting models (a) n = 1000 (b) n = 500 (c) n = 250 (d) n = 125 (e) n = 125 (f) n = 63 (g) n = 32 (h) n = 16 (h) n = 8*

The correlation matrix of the fit parameters at different sample sizes is given in Table 2. For the sample size of n=1000, the correlation coefficient between 'a' and 'b' in the correlation matrix is -0.866. This value indicates a strong negative correlation between these two variables. In the second correlation matrix, corresponding to a sample size of n=500, the correlation coefficient between 'a' and 'b' is -0.867. Although the difference between -0.866 and -0.867 may appear small, it suggests a slight change in the strength of the negative correlation between 'a' and 'b'. While the correlation coefficients show slight fluctuations, they universally indicate an inverse association between the slope and intercept of the fitted linear models across all examined sample sizes i. e. n= 250, n = 125, n=63, n=32, n=16, n=8. This reflects the inherent trade-off between these two parameters when fitting a line.

Table 2: *Correlation Matrix for Different Sample Sizes*

n=1000	a	b
a	1.000	
b	-0.866	1.000

n=500	a	b
a	1.000	
b	-0.866	1.000

n=250	a	b
a	1.000	
b	-0.867	1.000

n=125	a	b
a	1.000	
b	-0.868	1.000

n=63	a	b
a	1.000	
b	-0.869	1.000

n=32	a	b
a	1.000	
b	-0.873	1.000

n=16	a	b
a	1.000	
b	-0.879	1.000

n=8	a	b
a	1.000	
b	-0.891	1.000

The physical meanings inferred from the eight sets of correlation matrices are relatively consistent. They all suggest a negative relationship between 'a' and 'b', where increases in one variable tend to be associated with decreases in the other. The subtle differences in the correlation coefficients reflect slight variations in the strength of this negative relationship, but the overall interpretation remains the same.

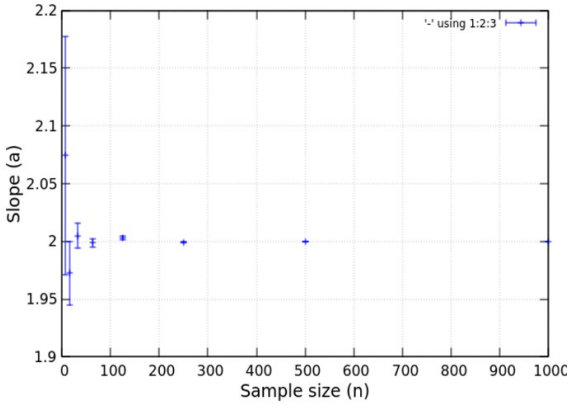

Figure 2: *Variation in slope with different sample size*

The variation in slope at different sample sizes (n=1000, n=500, n=250, n=125, n=63, n=32, n=16, n=8) with asymptotic standard error is shown in Table 1 and visualized in Figure 2. It seems the n=1000 slope data is plotted slightly offset

from its expected location at 1000 on the x-axis in order to reduce visual clutter, maintain even data point spacing, and enable cleaner visual comparisons against the other sample sizes presented. This appears to be a choice made to improve graphical clarity rather than driven by the underlying data or statistical analysis. The interpretation and conclusions would remain unchanged between plotting it at 950 or 1000.

3.1 KERNEL DENSITY ESTIMATE (KDE) AT DIFFERENT SAMPLE SIZES

In this section, we show the kernel density estimate at different sample sizes. The density along the y-axis is represented as a percentage and the value of x-axis represents the random number generated between 0 and 1. When the sample size n is large (1000 or 500) in the subplots 3(a) and 3(b), the KDE plots are very smooth and accurately represent the expected uniform distribution. This matches the theory that larger sample sizes lead to more precise density estimates. As n decreases as in subplots 3(c), 3(d) 3(e), 3(f), fluctuations and variability in the KDE plots increase progressively. This aligns with the statistical expectation that smaller sample sizes produce density estimates that tend to be more erratic and less reliable. At very small n (16 or 8) in subplots 3(g) and 3(h), the KDEs show completely chaotic variations rather than any visible underlying distribution. Such massive distortions are consistent with having too little data to provide any reasonable density approximation. The larger sample size KDEs are consistent in showing the uniform distribution, while smaller n plots get successively more variable. This matches the stated observation about consistency across sample sizes.

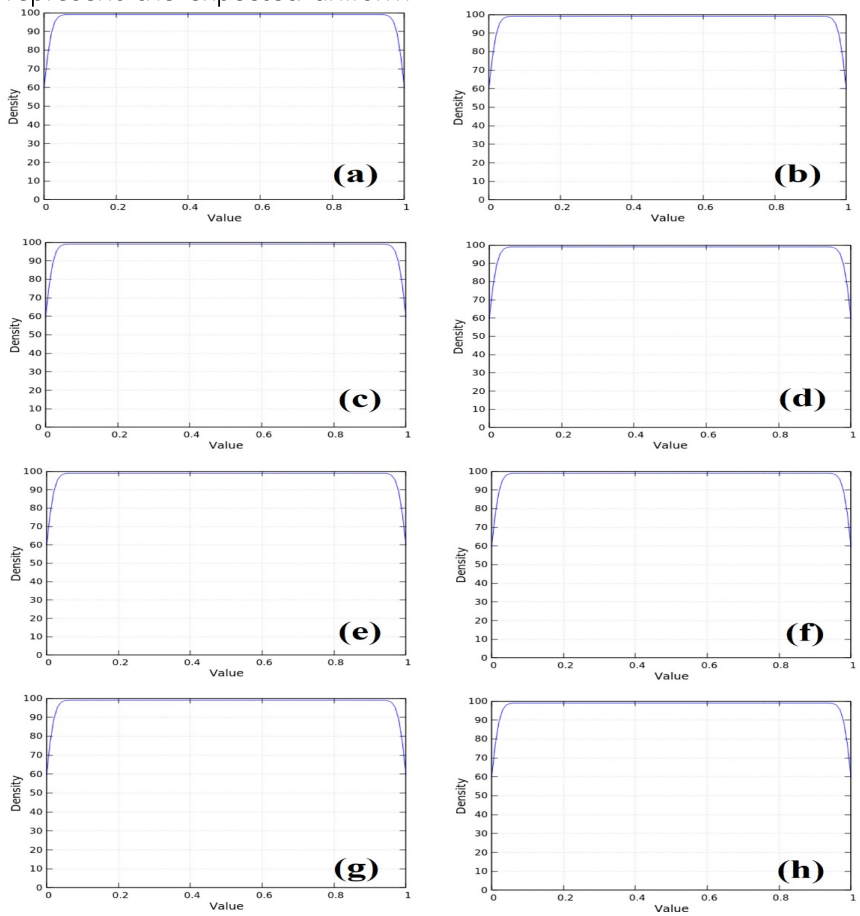

Figure 3: *Kernel density estimate (KDE) at different sample sizes (a) n = 1000 (b) n = 500 (c) n = 250 (d) n = 125 (e) n = 125 (f) n = 63 (g) n = 32 (h) n = 16 (h) n = 8*

4. CONCLUSION

As the sample size increases, the asymptotic standard error decreases which suggests that the estimate obtained from a large sample is expected to be closer to the true value of the parameter, with less variability or uncertainty. Lower asymptotic standard error in larger sample sizes provide more reliable estimates and can offer better insights into the population. However, where the available sample size is limited, the model can still yield a satisfactory fit, as indicated by the decrease in the sum of squares of residuals. The consistent nature of kernel density estimate implies that increasing or decreasing the sample size does not affect the overall shape of the estimated density, indicating stability in the estimated distribution. This consistency can be useful in various statistical analyses and modeling tasks. For example, if the density estimate is used to estimate probabilities or perform hypothesis testing, the stability across sample sizes ensures that the results obtained are reliable and not heavily influenced by the sample size.

5. ACKNOWLEDGEMENTS

We wish to acknowledge Prof. Dr. Narayan Prasad Adhikari for his insights into random number generators and guidance on utilizing gnuplot. Additionally, we thank Dr. Niraj Dhital for supplying the essential Fortran codes required for the project.

AUTHOR AFFILIATIONS

Rajendra Neupane
Assistant Professor, Central Department of Physics, T. U.
& Birendra Multiple Campus, T. U.
neupanerajendra5@gmail.com

Dr. Krishna Bahadur Rai
Lecturer of Physics
Patan Multiple Campus, T. U.

REFERENCES

1. **Davies, R., Rea, A., & Tsaptsinos, D.** (2018). Introduction to Fortran 90. Cardiff, London and Belfast HPC T & E Centres. 8-9. https://www.uv.es/dogarcar/man/IntrFortran90.pdf
2. **Akhshani, A., Akhavan, A., Mobaraki, A., Lim, S.C., & Hassan, Z.** (2014). Pseudo random number generator based on quantum chaotic map. Communications in Nonlinear Science and Numerical Simula- tion, 19(1), 101-111.
3. **Leva, J.L.** (1992). A fast-normal random number generator. ACM Transactions on Mathematical Software (TOMS), 18(4), 449-453.
4. **Schrage, L.** (1979). A more portable Fortran random number generator. ACM Transactions on Mathematical Software (TOMS), 5(2), 132-138.
5. **Yao, W., & Li, L.** (2014). A new regression model: modal linear regression. Scandinavian Journal of Statistics, 41(3), 656-671.
6. **Amiri-Simkooei, A., & Jazaeri, S.** (2012). Weighted total least squares formulated by standard least squares theory. Journal of geodetic science, 2(2), 113-124.
7. **Browne, M.W.** (1993). Alternative ways of assessing model fit. Testing structural equation models, 136-162.
8. **Liitia¨inen, E., Verleysen, M., Corona, F., & Lendasse, A.** (2009). Residual variance estimation in machine learning. Neurocomputing, 72(16-18), 3692-3703.
9. **Cox, D.R., & Snell, E.J.** (1968). A general definition of residuals. Journal of the Royal Statistical Society: Series B (Methodological), 30(2), 248-265.
10. **Altman, N.S.** (1992). An introduction to kernel and nearest-neighbor nonparametricregression. The American Statistician, 46(3), 175-185.
11. **Guidoum, A.C.** (2015). Kernel estimator and bandwidth selection for density and its derivatives. Department of Probabilities and Statistics, University of Science and Technology, Houari Boumediene, Algeria.
12. **Wand, M.P., & Jones, M.C.** (1995). Kernel Smoothing Chapman & Hall. CRC/London.
13. **Bhattacharya, P.K.** (1967). Estimation of a probability density function and its derivatives. Sankhya¯: The Indian Journal of Statistics, Series A, 373-382.
14. **Schuster, E.F.** (1969). Estimation of a probability density function and its derivatives. The Annals of Mathematical Statistics, 40(4), 1187-1195.
15. **Alekseev, V.G.** (1972). Estimation of a probability density function and its derivatives. Mathematical notes of the Academy of Sciences of the USSR, 12(5), 808-811.
16. **Gauli, H.R.K., Rai, K.B., Giri, K., & Neupane, R.** (2023). Monte-Carlo simulation of phase transition in 2d and 3d Ising model. Scientific World, 16(16), 12-20.
17. **Kra¨mer, W., & Sonnberger, H.** (2012). The linear regression model under test. Springer Science & Business Media, 3-4.

TO CITE THIS ARTICLE
Neupane, R., & Rai, K. B. (2024). A study of linear model fitting and distribution analysis on random numbers generated in Fortran. *International Research Journal of MMC, 5*(1), 43–52.
https://doi.org/10.3126/irjmmc.v5i1.63079

Submitted: 28 January 2024 **Accepted:** 13 February 2024 **Published:** 1 March 2024

International Research Journal of MMC (IRJMMC) is a peer-reviewed open access journal published by Research Management Cell, Makawanpur Multiple Campus, Hetauda