DOI: https://doi.org/10.34123/jurnalasks.v17i2.804

Jurnal Aplikasi Statistika & Komputasi Statistik
Vol. 17 No. 2, December 2025
e- ISSN: 2615-1367, p-ISSN:2086-4132

Available online at https://jurnal.stis.ac.id/

Adding MSNBURR-IIa Distribution to MultiBUGS

Eliana Putri Ramadani[1], Achmad Syahrul Choir[2*], Anindya Apriliyanti Pravitasari[3], Joynabel Paraguison[4]

[1]BPS-Statistics Kapuas Hulu Regency, Putussibau, Indonesia, [2]Politeknik Statistika STIS, Jakarta, Indonesia, [3]Department of Statistics, Faculty of Mathematics and Natural Sciences, Universitas Padjadjaran, Bandung, Indonesia, [4]Phillippine Statistics Authority, Phillippine
**Corresponding Author: E-mail address: madsyair@stis.ac.id*

ARTICLE INFO

Article history:
Received 12 September, 2024
Revised 13 February, 2025
Accepted 12 July, 2025
Published 31 December, 2025

Keywords:
Bayesian; MCMC; MSNBurr-IIa; MultiBUGS; Neo-normal

Abstract

Introduction/Main Objectives: The MSNBurr-IIa distribution is a neo-normal distribution designed to fit right-skewed data better. This article aims to integrate the MSNBurr-IIa distribution into MultiBUGS, thereby enabling Bayesian estimation of its parameters. **Background Problems:** Markov Chain Monte Carlo (MCMC) is a popular method for Bayesian computations, although its implementation is frequently challenging. MultiBUGS, a statistical tool that uses the BUGS language, is used to make this easier. **Novelty:** This paper details integrating the MSNBurr-IIa distribution into MultiBUGS, allowing for estimating its parameters. The module's effectiveness is demonstrated through its application on both simulated data and regional economic growth data of Indonesian districts/cities in 2021. **Research Methods:** The MSNBurr-IIa module was developed using five steps: requirement, design, development, testing, and implementation in simulation and real-world data. It was built with Blackbox Component Builder, an integrated development environment (IDE) for the Component Pascal programming language. **Finding/Results:** The findings confirm that MultiBUGS, with the MSNBurr-IIa module, successfully estimates the distribution's parameters across various datasets.

1. Introduction

The normal or Gaussian distribution is one of the most widely used statistical distributions. However, many real-world datasets exhibit skewness or fat tails, often resulting from outliers or other underlying factors. The symmetrical and mesokurtic nature of the normal distribution may not always align with the characteristics of real-world data, particularly those exhibiting skewness or non-normal kurtosis. Applying the normal distribution to non-normally distributed data, such as those with skewness or differing kurtosis, can lead to inaccurate inferences, as it fails to capture the underlying characteristics of the data adequately [1].

Modifying the normal distribution can overcome its limitations, yielding a more flexible model capable of accommodating diverse data characteristics [2]. Several methods exist for achieving this modification, including directly altering the probability density function (PDF) of the normal distribution [2], [3], applying Tukey's-*gh* transformation[4], [5], and employing the compounding method [6], [7]. These modifications can alter the distribution's symmetry and kurtosis, controlled by additional parameters introduced into the model [1]. For instance, in the skew-normal distribution, an

added skewness parameter allows for the control of skewness, with changes to this parameter directly altering the distribution's asymmetry [3].

Amemiya [8] proposed that the logistic distribution serves as an approximation of the standard normal distribution. Building upon this foundation, Iriawan [9] extended the concept to model skewed non-normal data by utilizing the Burr II distribution [10], which is a more flexible generalization of the logistic distribution. These distributions were termed 'neo-normal' by Iriawan [9].

Neo-normal distributions represent a class of distributions that either are inherently normal or can approximate normality under specific conditions [1], [9], [11]. For instance, the MSNBurr distribution can closely approximate normality when symmetric, the skew-normal distribution converges to the standard normal distribution when its skewness parameter is zero (indicating symmetry), and the exponential power distribution becomes identical to the normal distribution when its shape parameter results in a kurtosis of 3 (the kurtosis of a normal distribution, also known as mesokurtic).

The MSNBurr-IIa distribution extends the MSNBurr distribution to handle cases with opposite skewness characteristics [1]. Derived from the Burr-IIa distribution [12], the MSNBurr-IIa distribution is better suited for accommodating right-skewed data than left-skewed data. MSNBurr-IIa can estimate both symmetrical and skewed data distributions[1]. The parameters represent the shape, location, and scale parameters of the MSNBurr-IIa distribution α, μ, and σ, respectively. The shape parameter is related to variations in its skewness. Currently, the specific values of each parameter for the MSNBurr-IIa distribution are unknown. Consequently, estimation is required to determine these parameter values.

There are two primary methods for estimating parameters in statistics: the frequentist and Bayesian approaches [13]. The Bayesian approach is preferred in this study because it offers more flexible results than the frequentist method, particularly in handling complex models and incorporating uncertainty. This approach incorporates prior knowledge about the parameters, combining it with observed data to produce a probability distribution of the parameters after the data are observed, known as posterior distribution. While standard Bayesian calculations can become more complex and time-consuming when dealing with a larger number of parameters and data [14], advancements in user-friendly software, such as MultiBUGS, driven by scientific and technological progress, have made the Bayesian approach readily accessible.

Bayesian inference using Gibbs Sampling (BUGS) is a project developed to facilitate the implementation of Bayesian inference. The advantages of BUGS compared to other Bayesian software include ease of use and speed. It also has a model visualization feature called Doodle, which simplifies model understanding and analysis [15]. The latest and fourth version of the BUGS program is MultiBUGS. MultiBUGS was developed to address computational problems in previous software by employing an MCMC parallelization strategy. MultiBUGS can run on Linux and is fully implemenH within other statistical programs, such as R [14], [16].

Currently, only 29 theoretical distributions are available in MultiBUGS, although the program includes many specialized distributions, such as those for temporal, spatial, and reliability modeling. However, because MultiBUGS is open-source, users can modify the application as needed. Extensions to MultiBUGS can be implemented through the BlackBox Components Builder, an open-source Pascal-based Integrated Development Environment (IDE). BlackBox provides tools for creating module-based documents, assembling and running modules, and developing user interfaces, enabling users to customize MultiBUGS according to their specific requirements.

To address the absence of the MSNBurr-IIa distribution in MultiBUGS, we developed an MSNBurr-IIa distribution module. This addition aims to simplify the parameter estimation process for the MSNBurr-IIa distribution using the Bayesian approach, making it more accessible and user-friendly. The module underwent rigorous testing to ensure its functionality and accuracy. An example application involving real-world data with symmetry issues, for which the normal distribution is unsuitable, demonstrates the use of the module.

The addition of the MSNBurr-IIa distribution in MultiBUGS provides significant advantages for modeling heavy-tailed data, offering flexibility in handling both symmetric and asymmetric distributions. Compared to traditional distributions like the normal or logistic, MSNBurr-IIa allows better modeling of extreme values and heteroskedasticity, making it particularly useful for real-world data that exhibit high variability.

Beyond MultiBUGS, this distribution has also been integrated into other Bayesian software. Specifically, the R package neodistr extends support for Stan, while the R package neojags adds it to JAGS. These implementations enable Bayesian practitioners to leverage MSNBurr-IIa across multiple platforms, ensuring robust statistical modeling for diverse data structures.

2. Material and Methods

2.1. Bayesian Method

The Bayesian method is an inference method derived from Bayes' Theorem, where parameter values are unknown and treated as random variables [14],. These random variables are expressed using probability distributions: $f(\theta)$ represents the prior distribution and $f(\theta|\mathrm{x})$ represents the posterior distribution. The prior distribution reflects existing knowledge or beliefs about the parameters before analyzing the data (x) [8]. The posterior distribution is obtained by updating the prior distribution with observed data using Bayes' Theorem. The equation for the posterior distribution is

$$\boldsymbol{f(\theta|\mathbf{x})} = \frac{\boldsymbol{f(\mathbf{x}|\theta)f(\theta)}}{\boldsymbol{f(\mathbf{x})}} \propto \boldsymbol{f(\mathbf{x}|\theta)f(\theta)}. \quad (1)$$

2.2. MSNBurr-IIa Distribution

The probability density function of the MSNBurr-Ila distribution is given by

$$f(x|\mu,\sigma,\alpha) = \frac{\omega}{\sigma} exp\left(\frac{\omega}{\sigma}(x-\mu)\right)\left(1+\frac{exp\left(\frac{\omega}{\sigma}(x-\mu)\right)}{\alpha}\right)^{-(\alpha+1)} \quad (2)$$

where $-\infty < x < \infty$, $-\infty < \mu < \infty$, $\alpha > 0$, $\sigma > 0$, and

$$\omega = \frac{\left(1+\frac{1}{\alpha}\right)^{(\alpha+1)}}{\sqrt{2\pi}} \quad (3)$$

which α is a shape parameter, μ is location parameter, σ and is a scale parameter. If $\alpha > 1$ then data is left-skewed, if $\alpha = 1$ then data is symmetric, and if $0 < \alpha < 1$ then data is right-skewed. The mean, variance, mode, skewness, excess kurtosis, and quantile of the MSNBurr-Ila distribution can be computed using formulas (4)-(9), respectively.

$$E(x) = \mu + \frac{\sigma}{\tilde{\omega}(\alpha)}\{\psi^0(1) + \ln(\alpha) - \psi^0(\alpha)\} \quad (4)$$

$$Var(x) = \frac{\sigma^2}{\tilde{\omega}(\alpha)^2}\{\psi^1(\alpha) + \psi^1(1)\} \quad (5)$$

$$Mode = \mu \quad (6)$$

$$Skew(x) = \frac{\psi^2(1) - \psi^2(\alpha)}{\{\psi^1(\alpha)+\psi^1(1)\}^{\frac{3}{2}}} \quad (7)$$

$$\mathrm{K}(x) = \frac{\left(\psi^3(\alpha) - \psi^3(1)\right)}{(\psi^1(\alpha)+\psi^1(1))^2} \quad (8)$$

$$q(u) = \mu + \frac{\sigma}{\omega}\left(ln(\alpha) + ln\left((1-u)^{-\frac{1}{\alpha}} - 1\right)\right) \quad (9)$$

where $0 < u < 1$, ω follows equation (3), and $\psi^0(.)$ is $\psi^1(.)$, $\psi^2(.)$,$\psi^3(.)$ are digamma, trigamma, tetragamma, and pentagamma functions, respectively [1].

2.3. Deviance Information Criterion (DIC)

The Deviance Information Criterion (DIC), a widely used Bayesian model selection criterion, assesses model validity and goodness of fit. It balances model complexity and goodness of fit to the observed data. In the BUGS program, DIC is automatically calculated and can be used to compare different models. The model with the smallest DIC value is preferred, as it indicates the best fit to the data. DIC is calculated as

$$DIC = \bar{D} + P_D \quad (10)$$

where $\boldsymbol{\bar{D}}$ is the posterior mean of the deviance and $\boldsymbol{P_D}$ is the adequate number of parameters. (See [17] for further details).

2.4. MSNBurr-IIa Distribution Module Development

The research began by analyzing the requirements for a functional MSNBurr-IIa distribution module within MultiBUGS. This analysis informed the module's design, which was then implemented using Pascal and the "Univariatetemp1.odc" template file. The following adjustments were made to the template to accommodate the MSNBurr-IIa distribution: (1) Module initiation adjustments, (2) Parameter adjustments in all procedures, (3) Equation changes in several procedures, including ClassifyPrior, DevianceUnivariate, DiffLogLikelihood, DiffLogPrior, LogLikelihood-Univariate, LogPrior, Cumulative, and Sample, (4) Addition of procedures for generating random values from the MSNBurr-IIa distribution in “Randnum.odc” file, (5) Updating "The 'Cumulative.odc' file with a new procedure to compute the cumulative distribution function of the MSNBurr-IIa distribution, (6) Updating The 'External.odc', 'Make.odc', and 'Linking.odc' files to incorporate 'GraphMSNBurr2a', allowing the MSNBurr-IIa distribution module to be accessed and utilized, and (7) Modification of the 'Strings.odc' file to include the parameters and default values for the MSNBurr-IIa distribution, enabling its graphical representation within the software.

The next step involved rigorous testing and validation of the developed module to ensure its functionality and accuracy for practical use. This testing process consisted of two main steps: First, the MSNBurr-IIa distribution module was tested to ensure proper compilation and the absence of coding errors. This was achieved by running the module through Doodle, MultiBUGS's graphical interface for model visualization. The absence of error messages indicated a successful compilation. Second, the probability values generated by the module were validated by comparing them with equivalent calculations performed in the R programming language, using identical input parameters. This aimed to ensure the accuracy of the probability calculations within MultiBUGS. The validation used the absolute difference in cumulative distribution function (CDF) and probability density function (PDF) values, following the approach used by Annis et al. [18] for adding distributions to the Stan program. More minor absolute differences indicate higher accuracy in the MultiBUGS.

After confirming the module's functionality and accuracy, the next step focused on its implementation for simulated and real-world data. Simulation data was used to assess the performance of the MSNBurr-IIa distribution in accurately modeling data with various characteristics, including right-skewed, symmetrical, and left-skewed data. Normal distribution simulations served as a benchmark for comparison. Simulation data is generated from the distribution of MSNBurr-IIa for three scenarios with varying degrees of skewness. Scenario 1 for right-skewed data is MSNBurr-IIa$(0,1,0.1)$, Scenario 2 for symmetrical data is MSNBurr-IIa $(0,1,1)$, and Scenario 3 for left-skewed data is MSNBurr-IIa$(0,1,3)$. Additionally, Scenario 4 is normal$(0,1)$. Each scenario comprised 1,000 simulated samples.

For real-world application, the module analyzed the economic growth rate of districts/cities in Indonesia in 2021. This data was used because it has a non-normal distribution, thus demonstrating the advantages of the MSNBurr-IIa distribution. This application involved several steps: First, the economic growth rate data was examined using the Shapiro-Wilk test for normality and the skewness coefficient to understand its characteristics and determine the degree of skewness. Second, the parameters of the MSNBurr-IIa distribution were estimated using the developed module in MultiBUGS. Convergence of the estimation process was assessed through trace plots, which visually display the sampled values of the parameters over time. Finally, the model was validated by comparing the characteristics of the fitted MSNBurr-IIa distribution (e.g., mean, variance, and skewness) with the observed characteristics of the economic growth rate data. This comparison, which included visual assessments of histograms, evaluated the model's goodness of fit and ability to represent the real-world data accurately.

3. Results and Discussion

3.1. Requirements

The necessary conditions in MultiBUGS can be identified by examining the functions and conditions in existing distribution modules, such as the normal and logistics distribution modules, which served as references for developing the MSNBurr-IIa distribution module. To create a new module in MultiBUGS, the following are required:

1. The MultiBUGS source code, including a template for new distribution modules and necessary modules (see Figure 3)
2. BlackBox component builder v1.7 from Oberon Microsystems (http://www.oberon.ch).
3. Mathematical formulas, including formulas for the log-likelihood, its derivatives, random variate generation, and the cumulative distribution function.

3.2. Module Design

The MSNBurr-IIa module in MultiBUGS is developed using the 'univariatetemp.odc' file template. To adhere to the MultiBUGS architecture, modifications are made not only to the 'Univariatetemp.odc' file but also involve adding or modifying code in several other files, including 'Randnum.odc', 'Cumulative.odc', 'External.odc', 'Make.odc', 'Linking.odc', 'Strings.odc'. The files modified within the architecture are highlighted in the diagram in Figure 1. Green indicates folders, while purple highlights indicate the files requiring modification.

For consistency, we change the parameter σ to τ, where $\tau = \frac{1}{\sigma}$. This change affects the probability density function (pdf). The pdf of the MSNBurr-IIa distribution after parameter changes is as follows.

$$f(x|\omega,\mu,\tau,\alpha) = \omega\tau \exp\big(\omega\tau(x-\mu)\big)\left(1+\frac{\exp(\omega\tau(x-\mu))}{\alpha}\right)^{-(\alpha+1)} \tag{11}$$

where ω follow equation (3). Changes in the PDF will cause changes to the other equations used, namely, as follows.

1. The natural logarithm of the probability density function
$$\ln\big(f(x)\big) = \ln(\omega) + \log(\tau) + \omega\tau(x-\mu) - (1-\alpha)\log\left(1+\frac{\exp(\omega\tau(x-\mu))}{\alpha}\right) \tag{12}$$
2. The natural logarithm of the unnormalized probability density function
$$\ln\big(p(x)\big) = \omega\tau(x-\mu) - (1-\alpha)\log\left(1+\frac{\exp(\omega\tau(x-\mu))}{\alpha}\right) \tag{13}$$
3. The natural logarithmic derivative of the probability density function for variable X
$$\frac{d}{dx}\log\big(f(x)\big) = -\frac{\alpha\omega\tau\big(\exp(\omega\tau(x-\mu))-1\big)}{\exp(\omega\tau(x-\mu))+\alpha} \tag{14}$$
4. Partial derivative of parameter α
$$\frac{\partial}{\partial\alpha}\log\big(f(x)\big) = -\frac{\exp(\omega\tau(x-\mu))(-1-\alpha)}{\left(\frac{\exp(\omega\tau(x-\mu))}{\alpha^2}+1\right)\alpha^2} - \log\left(\frac{\exp(\omega\tau(x-\mu))}{\alpha}+1\right) \tag{15}$$
5. Partial derivative of parameter μ
$$\frac{\partial}{\partial\mu}\log\big(f(x)\big) = \frac{\alpha\omega\tau\big(\exp(\omega\tau(x-\mu))-1\big)}{\exp(\omega\tau(x-\mu))+\alpha} \tag{16}$$
6. Partial derivative of parameter
$$\frac{\partial}{\partial\tau}\log\big(f(x)\big) = \frac{\omega(-1-\alpha)(x-\mu)\exp(\omega\tau(x-\mu))}{\alpha\left(\frac{\exp(\omega\tau(x-\mu))}{\alpha}+1\right)} + \frac{1}{\tau} + \omega(x-\mu) \tag{17}$$
7. Quantile
$$x = \mu + \frac{1}{\omega\tau}\left(\log(\alpha) + \log\left((1-u)^{-\frac{1}{\alpha}}-1\right)\right) \tag{18}$$
8. Cumulative distribution function
$$F(x) = 1 - \left(1-\frac{\exp(\omega\tau(x-\mu))}{\alpha}\right)^{-} \tag{19}$$

Making doodles the MSNBurr-IIa distribution in MultiBUGS requires a list of parameters used in the MSNBurr-IIa distribution: location parameter, rate parameters, shape parameters, and the default values of each parameter. along with the default values of each. The default values used are 0 for the location parameter, 1.0×10^{-3} for the rate parameter and 0.1 for the shape parameter. The diagram illustrating the relationships between modules in creating the MSNBurr-IIa distribution module is visualized in Figure 1.

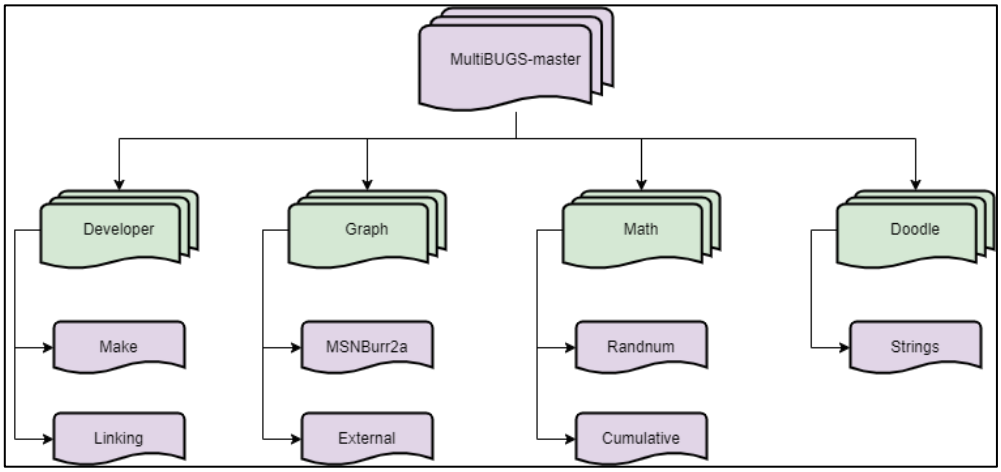

Figure 1. Diagram of the relationship between modules in the creation of the MSNBurr-IIa distribution module

3.3. Module Development

During the module creation stage, the module code will be written according to the design and equations prepared during the module design stage. The module is created using the "Univaritetemp1.odc" template in the MultiBUGS-master file. The code template is written in Pascal and displayed in black, red, and green colors. Lines of code in green and enclosed by (* ... *) are not executed by the program, as they are comments. Lines of code in black should not be changed except for definitions added. Lines of code in red can be modified according to the user's needs.

The MSNBurr-IIa distribution module-file will be written in a new file named "MSNBurr2a.odc". An example of the beginning code in the file "MSNBurr2a.odc" can be seen in Figure 2. This file is saved in the “Graph\mod” folder

The MSNBurr-IIa distribution module is initiated at the beginning of the file, first by naming the module "GraphMSNBurr2a". This name is chosen because the module will be stored in the Graph folder for the MSNBurr-IIa distribution. After naming the module, the folders used in the MSNBurr-IIa distribution module are specified: the Math folder, which stores mathematical function modules, and the Stores folder for storage in the IMPORT function. The TYPE function declares the data types of the nodes and parameters that will be used. Next, the Factory function is an abstract class to create nodes in the graphical model. A constant is also declared, specifically ε (eps), with a value of 1.0×10^{-10}. Finally, variables are declared, namely fact, version, and maintainer.

```
MODULE GraphMSNBurr2a;

   IMPORT
      Math, Stores := Stores64,
      GraphNodes, GraphRules, GraphStochastic, GraphUnivariate,
      MathCumulative, MathFunc, MathRandnum;

   TYPE
      Node = POINTER TO RECORD(GraphUnivariate.Node)
         mu, tau, alpha: GraphNodes.Node
   END;

   Factory = POINTER TO RECORD(GraphUnivariate.Factory)END;

   CONST
      eps = 1.0E-10;

   VAR
      fact-: GraphUnivariate.Factory;
      version-: INTEGER;
      maintainer-: ARRAY 40 OF CHAR;
```

Figure 2. Code of the beginning part of the file “MSNBurr2a.odc”

The *BoundsUnivariate* procedure establishes that the distribution is a continuous univariate distribution, with a domain of $(-\infty, \infty)$. The *CheckUnivariate* procedure verifies if the distribution is univariate and checks compliance with the MSNBurr-IIa distribution. These requirements dictate that the parameters τ and α must be positive. If $\tau < -\epsilon$ and $\alpha < -\epsilon$, an error will be triggered, indicating that the second and third arguments contain non-positive parameters (which is invalid for this distribution). The code implementing the *BoundsUnivariate* and *CheckUnivariate* procedures within the "MSNBurr2a.odc" file is shown in Figure 3.

```
PROCEDURE(node: Node)CheckUnivariate(): SET;
BEGIN
   IF node.tau.value < - eps THEN
      RETURN{GraphNodes.posative,GraphNodes.arg2
   END;
   IF node.alpha.value < -eps THEN
      RETURN{GraphNodes.posative,GraphNodes.arg3
   END;
   RETURN{}
END CheckUnivariate;
```

Figure 3. Code of boundsunivariate and checkunivariate procedure in "MSNBurr2a.odc"

The *ClassifyLikelihoodUnivariate* procedure classifies the likelihood of a distribution by examining the parent nodes of each parameter (α, μ, and τ). Other parts of the MultiBUGS system likely use this classification to determine how to handle the distribution during calculations and statistical inference. The parent nodes of parameters are categorized by the *ClassFunction*, which is found in the "Stochastic.odc" file (Figure 4).

```
PROCEDURE(node: Node)ClassifyLikelihoodUnivariate(parent: GraphStochastic.Node): INTEGER
VAR
      density, density0, density1, f0, f1: INTEGER;
BEGIN
   f0 := GraphStochastic.ClassFunction(node.mu, parent);
   f1 := GraphStochastic.ClassFunction(node.tau, parent);
   CASE f0 OF
   |GraphRules.const:
      density0 := GraphRules.unif
   |GraphRules.ident, GraphRules.prod, GraphRules.linear:
      density0 := GraphRules.logCon
   |GraphRules.other:
      density0 := GraphRules.general
   ELSE
      density0 := GraphRules.genDiff
   END;
   density1 := GraphRules.ClassifyShape(f1);
   IF density0 = GraphRules.unif THEN
      density := density1
   ELSIF density1 = GraphRules.unif THEN
      density := density0
   ELSIF (density0 # GraphRules.general)& (density1 # GraphRules.general)THEN
      density := GraphRules.genDiff
   ELSE
      density := GraphRules.general
   END;
   RETURNdensity
END ClassifyLikelihoodUnivariate;
```

Figure 4. Code of classifylikelihoodunivariate procedure in "MSNBurr2a.odc"

The *ClassifyPrior* procedure, as shown in Figure 5, returns a value indicating that the prior for the MSNBurr-IIa distribution is logCon (log-concave). This suggests that the parameters of this distribution are assumed to have priors belonging to the log-concave distribution family. The *Cumulative* procedure calculates the cumulative probability for the MSNBurr-IIa distribution, using Equation (19) for the cumulative distribution function.

```
PROCEDURE(node: Node)ClassifyPrior(): INTEGER;
   VAR
      class : INTEGER;
BEGIN
   class := GraphRules.logCon;
   RETURNclass
END ClassifyPrior;

PROCEDURE(node: Node)Cumulative(x: REAL): REAL;
VAR
   cumulative, mu, tau, alpha, omega: REAL;
BEGIN
   mu := node.mu.value;
   tau := node.tau.value;
   alpha := node.alpha.value;
   omega := Math.Power(1 + (1/alpha), (1 + alpha)) / Math.Sqrt(2.0 * Math.Pi());
   cumulative := 1 - Math.Power((1 + (Math.Exp(omega * (x-mu) * tau))/ alpha), -alpha)
   RETURNcumulative;
END Cumulative;
```

Figure 5. The code of classifyprior dan cumulative procedure in "MSNBurr2a.odc"

The DevianceUnivariate procedure (Figure 6) defines the distribution's deviance, which is calculated as negative two times the logarithm of the probability density function, as described in Equation (12). This procedure is also commonly used in calculating the Deviance Information Criterion (DIC).

```
PROCEDURE(node: Node)DevianceUnivariate(): REAL;
VAR
   logDensity, logTau, logOmega, logExp, mu, tau, alpha, x, omega: REAL;
BEGIN
   x := node.value;
   mu := node.mu.value;
   tau := node.tau.value;
   alpha := node.alpha.value;
   omega := Math.Power(1 + (1/alpha), (1 + alpha)) / Math.Sqrt(2.0 * Math.Pi());
   logTau := MathFunc.Ln(tau);
   logOmega := MathFunc.Ln(omega);
   logDensity := logOmega + logTau + (omega*(x-mu)*tau)- (1 + alpha) * Math.Ln(1.0+
(Math.Exp(omega*(x-mu)*tau))/ alpha);
RETURN-2.0 * logDensity;
END DevianceUnivariate;
```

Figure 6. The code of devianceunivariate procedure in"MSNBurr2a.odc"

The *DiffLogLikelihood* procedure calculates the derivative of the log-likelihood function concerning each parameter. These derivatives are defined in Equations (15), (16), and (17). The *DiffLogPrior* procedure calculates the derivative of the log probability density function for the MSNBurr-IIa distribution, based on the random variable X. In the context of the MSNBurr-IIa distribution, this derivative refers to the log of the probability density function (log-pdf) concerning the variable X, as defined in Equation (14). The *DiffLogLikelihood* and *DiffLogPrior* procedures are shown in Figures 7 and 8, respectively.

```
PROCEDURE(node: Node) DiffLogLikelihood(x: GraphStochastic.Node): REAL;
VAR
    mu, tau, alpha, val, omega, differential, diffTau, diffMu, exp, diffAlpha: REAL;
BEGIN
    val := node.value;
    mu := node.mu.value;
    tau := node.tau.value;
    alpha := node.alpha.value;
    omega := Math.Power(1 + (1/alpha), (1 + alpha)) / Math.Sqrt(2.0 * Math.Pi());
    exp := Math.Exp(omega*(val-mu)*tau);
    IF (GraphStochastic.hint1 IN x.props) OR (GraphNodes.dataIN node.tau.props) OR (GraphNodes.data
IN node.alpha.props) THEN
        diffMu := node.mu.Diff(x);
        differential := diffMu * ((alpha * omega * tau * exp - 1) / (exp + alpha));
    ELSIF (GraphStochastic.hint1 IN x.props) OR (GraphNodes.dataIN node.mu.props) OR
(GraphNodes.dataIN node.alpha.props) THEN
        diffTau := node.tau.Diff(x);
        differential := diffTau* (((-alpha-1)* omega * (val- mu) * exp) / (alpha * (exp / alpha + 1)) + (1/tau) +
omega * (val- mu));
    ELSIF (GraphStochastic.hint1 IN x.props) OR (GraphNodes.dataIN node.mu.props) OR
(GraphNodes.dataIN node.tau.props) THEN
        diffAlpha := node.alpha.Diff(x);
        differential := diffAlpha * (- (exp * (-1-alpha))/ ((exp / alpha + 1) * alpha * alpha) - Math.Ln(exp /
alpha + 1));
    ELSE
        diffMu := node.mu.Diff(x);
        diffTau := node.tau.Diff(x);
        diffAlpha := node.alpha.Diff(x);
        differential := (diffMu* ((alpha * omega * tau * exp - 1) / (exp + alpha)) + (diffTau* (((-alpha-1)*
omega * (val- mu) * exp) / (alpha * (exp / alpha + 1)) + (1/tau) + omega * (val- mu))) + (diffAlpha * (- (exp *
(-1-alpha))/ ((exp / alpha + 1) * alpha * alpha) - Math.Ln(exp / alpha + 1))));
    END;
    RETURN differential;
    END DiffLogLikelihood;
```

Figure 7. The code of the diffloglikelihood procedure in "MSNBurr2a.odc"

```
PROCEDURE(node: Node) DiffLogPrior(): REAL;
VAR
    differential, exp, mu, tau, alpha, omega, x: REAL;
BEGIN
    x := node.value;
    mu := node.mu.value;
    tau := node.tau.value;
    alpha := node.alpha.value;
    omega := Math.Power(1 + (1/alpha), (1 + alpha)) / Math.Sqrt(2.0 * Math.Pi());
    exp := Math.Exp(omega * (x - mu) * tau);
    differential := (- (alpha * omega * tau *(exp - 1)) / (exp + alpha));
RETURN differential;
END DiffLogPrior;
```

Figure 8. The code of the difflogprior procedure in "MSNBurr2a.odc"

The *ExternalizeUnivariate* procedure, shown in Figure 9, functions to externalize or write the nodes of each parameter—namely α, μ, and τ,—to be stored in Stores. Meanwhile, the *InternalizeUnivariate* procedure is used to internalize or read the nodes of each stored parameter. The *InitUnivariate* procedure initializes the nodes for the univariate distribution, with the initial value of each node being NIL.

```
PROCEDURE(node: Node) ExternalizeUnivariate (VAR wr: Stores.Writer)
BEGIN
   GraphNodes.Externalize(node.mu, wr);
   GraphNodes.Externalize(node.tau, wr);
   GraphNodes.Externalize(node.alpha, wr);
END ExternalizeUnivariate;

PROCEDURE(node: Node) InitUnivariate;
BEGIN
   node.mu := NIL;
   node.tau := NIL;
   node.alpha := NIL;
END InitUnivariate;

PROCEDURE(node: Node) InternalizeUnivariate(VAR rd: Stores.Reader)
BEGIN
   node.mu := GraphNodes.Internalize(rd);
   node.tau := GraphNodes.Internalize(rd);
   node.alpha := GraphNodes.Internalize(rd);
END InternalizeUnivariate;
```

Figure 9. The code of externalizeunivariate, initunivariate, and internalizeunivariate procedures in "MSNBurr2a.odc"

The *Install* procedure in Figure 10 sets up the MSNBurr-IIa distribution, ensuring that the "MSNBurr2a.odc" file is automatically integrated when the "External.odc" file is compiled. The *Location** procedure retrieves the value of the location parameter, which is μ in the MSNBurr-IIa distribution.

```
PROCEDURE(node: Node) Install (OUT install: ARRAY OF CHAR)
BEGIN
   install := "GraphMSNBurr2a.Install"
END Install;

PROCEDURE(node: Node) Location (): REAL;
   VAR
      mu: REAL;
BEGIN
   mu := node.mu.value;
RETURN mu;
END Location;
```

Figure 10. The code of install and location procedure in "MSNBurr2a.odc"

The *LogLikelihoodUnivariate* procedure in Figure 11 is used to compute the log-likelihood function of the probability density function, as specified in equation (33). This procedure ensures that the log-likelihood is correctly defined for univariate distributions, essential for accurate Bayesian computation.

```
PROCEDURE(node: Node) LogLikelihoodUnivariate(): REAL;
   VAR
      logDensity, logTau, logOmega,mu, tau,  alpha, x, omega: REAL;
BEGIN
   x := node.value;
   mu := node.mu.value;
   tau := node.tau.value;
   alpha := node.alpha.value;
   omega := Math.Power(1 + (1/alpha), (1 + alpha)) / Math.Sqrt(2.0 * Math.Pi());
   logTau := MathFunc.Ln(tau);
   logOmega := MathFunc.Ln(omega);
   logDensity := logOmega + logTau + omega*(x-mu)*tau - (1 + alpha) * Math.Ln(1.0+
(Math.Exp(omega*(x-mu)*tau) / alpha));
RETURN logDensity;
END LogLikelihoodUnivariate;
```

Figure 11. The code of loglikelihoodunivariate procedure in "MSNBurr2a.odc"

The *LogPrior* procedure, as in Figure 12, defines the equation for the log-likelihood function of the probability density function, which is dependent on the variable X, as specified in equation (31). This procedure is crucial for incorporating prior distributions into the model, allowing for a

comprehensive Bayesian analysis by accounting for the influence of prior knowledge on the current data.

```
PROCEDURE(node: Node)LogPrior(): REAL;
    VAR
        x, mu, tau, alpha, omega, logPrior: REAL;
BEGIN
    x := node.value;
    mu := node.mu.value;
    tau := node.tau.value;
    alpha := node.alpha.value;
    omega := Math.Power(1 + (1/alpha), (1 + alpha)) / Math.Sqrt(2.0 * Math.Pi());
    logPrior := omega*(x-mu)*tau - (1 + alpha) * Math.Ln(1.0 + (Math.Exp(omega*(x-mu)*tau)/alpha))
RETURN logPrior;
END LogPrior;
```

Figure 12. Source code prosedur logprior in “MSNBurr2a.odc”

The *ParentsUnivariate* procedure (Figure 13) adds the parents of each parameter node to a list. The *SetUnivariate* procedure is used to insert the nodes of each parameter into an array.

```
PROCEDURE(node: Node)ParentsUnivariate(all: BOOLEAN): GraphNodes.List;
    VAR
        list: GraphNodes.List;
BEGIN
    list := NIL;
    node.mu.AddParent(list);
    node.tau.AddParent(list);
    node.alpha.AddParent(list);
    RETURN list;
END ParentsUnivariate;

PROCEDURE(node: Node)SetUnivariate(IN args: GraphNodes.Args; OUT res: SET);
BEGIN
    res := {};
    WITH args: GraphStochastic.Args DO
        ASSERT(args.scalars[0] # NIL, 21);
        node.mu := args.scalars[0];
        ASSERT(args.scalars[1] # NIL, 21);
        node.tau := args.scalars[1];
        ASSERT(args.scalars[2] # NIL, 21);
        node.alpha := args.scalars[2];
    END;
END SetUnivariate;
```

Figure 13. The code of parentsunivariate and setunivariate procedure in “MSNBurr2a.odc”

The *Sample* procedure shown in Figure 14 defines the sampling method to be used in the estimation process. If the distribution has no constraints or domain, sampling will use the MSNBurr2a function from the "Randnum.odc" file in the “Math\mod” folder. If the distribution has constraints or a domain, sampling will use the MSNBurr2aLB, MSNBurr2aRb, or MSNBurr2aIB functions.

```
PROCEDURE(node: Node) Sample (OUT res: SET);
    VAR
        mu, tau, omega, alpha, x, lower, upper: REAL;
        bounds: SET;
BEGIN
    mu := node.mu.value;
    tau := node.tau.value;
    alpha := node.alpha.value;
    bounds := node.props * {GraphStochastic.leftImposed, GraphStochastic.rightImposed};
    IF bounds = {} THEN
        x := MathRandnum.MSNBurr2a(mu, tau, alpha);
    ELSE
        node.Bounds(lower,upper);
        IF bounds = {GraphStochastic.leftImposed} THEN
            x := MathRandnum.MSNBurr2aLB(mu,tau, alpha, lower);
        ELSIF bounds = {GraphStochastic.rightImposed} THEN
            x := MathRandnum.MSNBurr2aRB(mu,tau, alpha, upper);
        ELSE
            x := MathRandnum.MSNBurr2aIB(mu,tau, alpha, lower, upper);
        END;
    END;
    node.value := x;
    res := {};
END Sample;
```

Figure 14. The code of the sample procedure in "MSNBurr2a.odc"

The New procedure (Figure 15) is responsible for creating and initializing a new instance of a Node object, which represents a key element in the probabilistic graphical model used by MultiBUGS. It allocates memory for the Node, calls its Init method to configure default properties, and returns the fully initialized object. This process is essential for defining new distributions, as each corresponds to a unique node in the model's computational graph. The Signature procedure, on the other hand, assigns a specific identifier ("sssCT") to the distribution or object, where "sss" indicates that the MSNBurr-IIa distribution has three parameters.

```
PROCEDURE(f: Factory) New (): GraphUnivariate.Node;
    VAR
        node: Node;
BEGIN
    NEW(node);
    node.Init;
    RETURN node;
END New;

PROCEDURE(f: Factory) Signature (OUT signature: ARRAY OF CHAR)
BEGIN
    signature := "sssCT";
END Signature;
```

Figure 15. The code of the new and signature procedure in "MSNBurr2a.odc"

The code in Figure 16 defines procedures for initializing and managing a module named GraphMSNBurr2a. The *Init* procedure creates a new instance of Factory, sets it to fact, and calls the Maintainer procedure to assign a version number and maintainer name. The *Install* procedure sets the fact object as the factory for GraphNodes, and the entire module begins by calling *Init* to initialize its components.

```
PROCEDUREInstall*;
   BEGIN
      GraphNodes.SetFactory(fact);
   END Install;

   PROCEDUREMaintainer;
   BEGIN
      version:= 500;
      maintainer := "Eliana P. Ramadani";
   END Maintainer;

   PROCEDUREInit;
      VAR
      f: Factory;
   BEGIN
      Maintainer;
      NEW(f);
      fact := f
   END Init;

   BEGIN
      Init
   END GraphMSNBurr2a.
```

Figure 16. The code of procedure install and signature in "MSNBurr2a.odc"

In addition to creating the MSNBurr-IIa distribution module file, it is necessary to add and modify code in several other files to support the MSNBurr-IIa distribution module. Code additions to the "Randnum.odc" file (Figure 17) involve implementing four procedures for generating values distributed according to the MSNBurr-IIa distribution, which will then be used for sampling during estimation. The algorithm used in value generation employs the inverse transform method with the following steps:

1. Generate u from a Uniform(0,1) distribution.
2. Let $v = F(a) + [F(b) - F(a)]u$
3. The generated value is $x = F^{-1}(v)$,

where $F(x)$ is cumulative distribution function in equation (19) , $F(a)$ is the cumulative probability of the function when x equals the lower bound of the distribution, $F(b)$ is the cumulative probability when x equals the upper bound, and $x = F^{-1}(v)$ represents the inverse function as defined in equation (18).

```
   PROCEDUREMSNBurr2a*(mu, tau, alpha: REAL): REAL;
      VAR
         u, omega: REAL;
   BEGIN
      u := generator.Rand();
      omega := Math.Power((1 + 1/alpha), (1+alpha))/ Math.Sqrt(2 * Math.Pi());
      RETURNmu + (1 /(omega*tau))* (Math.Ln(alpha) + Math.Ln(Math.Power( 1 - u, (-1/alpha))- 1))
   END MSNBurr2a;
```

Figure 17. The code of generate MSNBurr2a distribution in "Randnum.odc"

When a distribution is known to have a lower bound, an upper bound, or both, it will have a specific generating function for these conditions. The conditional distribution resulting from restricting the domain of certain other probability distributions is called a truncated distribution. The function for the truncated distribution is as follows:

$$f^*(x) = \begin{cases} \frac{f(x)}{F(b)-F(a)}, & a \le x \le b \\ 0, & others \end{cases} \tag{20}$$

To calculate $F(a)$ and $F(b)$, we need to compute the value of the CDF of MSNBurr-IIa. The code for calculating it is written in the file Cumulative.odc in the "Math\Mod" folder, as shown in Figure 18.

```
PROCEDUREMSNBurr* (mu, tau, alpha, x: REAL): REAL;
   VAR
        omega: REAL;
BEGIN
   omega := Math.Power(1 + (1/alpha), (1 + alpha)) / Math.Sqrt(2.0 * Math.Pi());
   RETURN Math.Power(1+(Math.Exp(-omega*(x-mu)*tau)/alpha),alpha);
END MSNBurr;
```

Figure 18. The code of CDF MSNBurr2a distribution in "Cumulative.odc"

Therefore, the additions to the "Randnum.odc" file for truncated distributions are as follows: for truncation on the left side, truncation on the right side, and truncation on both sides (Figure 19-21).

1. The distribution is truncated on the left side

```
PROCEDURE MSNBurr2aLB* (mu, tau, alpha, lower: REAL): REAL;
    VAR
       u, pLower, omega: REAL;
BEGIN
    pLower := MathCumulative.MSNBurr2a(alpha,  mu, tau, lower);
    u := pLower + (1 - pLower) * generator.Rand();
    omega := Math.Power((1 + 1/alpha), (1+alpha)) / Math.Sqrt(2 * Math.Pi());
    RETURN mu + (1/(omega*tau)) * (Math.Ln(alpha) + Math.Ln(Math.Power( 1 - u, (-1/alpha)) - 1));
END MSNBurr2aLB;
```

Figure 19. The code of MSNBurr2aLB procedure in "Randnum.odc"

2. The distribution is truncated on the right side

```
PROCEDURE MSNBurr2aRB* (mu, tau, alpha, upper: REAL): REAL;
    VAR
       u, pUpper, omega: REAL;
BEGIN
    pUpper:= MathCumulative.MSNBurr2a(alpha,  mu, tau, upper);
    u := pUpper * generator.Rand();
    omega := Math.Power((1 + 1/alpha), (1+alpha)) / Math.Sqrt(2 * Math.Pi());
    RETURN mu + (1 / (omega*tau)) * (Math.Ln(alpha) + Math.Ln(Math.Power( 1 - u, (-1/alpha)) - 1));
END MSNBurr2aRB;
```

Figure 20. The code of MSNBurr2aRB procedure in "Randnum.odc"

3. The distribution is truncated on both sides, the left and right

```
PROCEDUREMSNBurr2aIB*(mu, tau, alpha, lower, upper: REAL): REAL;
    VAR
       u, pLower, pUpper, omega: REAL;
BEGIN
    pLower := MathCumulative.MSNBurr2a(alpha,mu, tau, lower);
    pUpper := MathCumulative.MSNBurr2a(alpha,mu, tau, upper);
    u := pLower + (pUpper - pLower) * generator.Rand();
    omega := Math.Power((1 + 1/alpha), (1+alpha))/ Math.Sqrt(2 * Math.Pi());
    RETURNmu + (1/(omega*tau))* (Math.Ln(alpha) + Math.Ln(Math.Power( 1 - u, (-1/alpha))- 1))
END MSNBurr2aIB;
```

Figure 21. The code of MSNBurr2aIB procedure in "Randnum.odc"

Code additions to the "External.odc" file (Figure 22), located in the "Bugs\Mod" folder, are made to integrate the MSNBurr-IIa module so it can be accessed alongside other distributions. These additions are made within the *LOAD* procedure. The first addition includes the MSNBurr-IIa univariate distribution from the previously created module file. The second addition loads the function for the MSNBurr-IIa cumulative distribution function (CDF). The third addition loads the function for the MSNBurr-IIa probability density function (PDF). Finally, additional code is added to load the available deviance function.

```
    Density("dmsnburr2a", "GraphMSNBurr2a.Install");
    Function("pdf.msnburr2a", "GraphDensity.DensityUVInstall(dmsnburr2a)");
    Function("cdf.msnburr2a", "GraphDensity.CumulativeInstall(dmsnburr2a)");
    Function("dev.msnburr2a","GraphDensity.DevianceUVInstall(dmsnburr2a)"
```

Figure 22. Adding MSNBurr-Iia code in "External.odc"

The code additions to "Linking.odc", as shown in Figure 23, are intended to ensure that the MSNBurr-IIa distribution module can be executed as part of a dynamic-link library. The compilation process depends on the operating system used, either Linux or Windows. In addition to modifying

"Linking.odc", code is also added to "Make.odc" (Figure 24) to ensure that the MSNBurr-IIa module can be compiled and linked correctly along with other programs. The modifications to both files involve including "GraphMSNBurr2a" (the name of the module) in the same section as the other distribution files. Both files are in “Developer” folder.

```
GraphRules GraphNodes GraphGrammar GraphLogical GraphStochastic GraphLimits
GraphScalar GraphLinkfunc GraphVector GraphParamtrans GraphWeight
GraphUnivariate GraphConjugateUV GraphVD GraphCensoring Truncation
GraphMultivariate GraphConjugateMV GraphChain GraphGMRF GraphFunctional
GraphBlock GraphMessages GraphResources GraphMSNBurr2a
```

Figure 23. Adding graphMSnburr2a in “Linking.odc”

```
GraphBern GraphBinomial GraphCat GraphFounder GraphGeometric
GraphHypergeometric GraphMendelian GraphNegbin GraphPoisson GraphRecessive
GraphZipf GraphMSNBurr2a
```

Figure 24. Adding graphMSnburr2a in “Make.odc”

Code additions to the "Strings.odc" file, shown in Figure 25, are made to include a doodle for the MSNBurr-IIa distribution, facilitating easier visual modeling within MultiBUGS. This addition involves listing the parameters used in the MSNBurr-IIa module, as defined during the module design phase. Each parameter is associated with index 17, which corresponds to the assigned sequence number of the MSNBurr-IIa distribution within the system.

```
densities[17]    dmsnburr2a
param0[17]       location
param1[17]       rate
param2[17]       shape
default0[17]     0
default1[17]     1.0E-3
default2[17]     1
```

Figure 25. Adding code for MSNburr-Iia module in “Strings.odc”

To integrate the MSNBurr-IIa distribution module, MultiBUGS needs to be recompiled. This is done by opening the "Make.odc" file and clicking the exclamation mark icon (typically used to initiate the build process). The MSNBurr-IIa distribution module will then be compiled along with other MultiBUGS components, successfully integrating it into the system. The complete procedure for compiling MultiBUGS source code is documented at: https://github.com/MultiBUGS/MultiBUGS.

After the module has been created and the necessary changes have been made to other related files, MultiBUGS should be recompiled using the "Make.odc" file. Once this is done, the MSNBurr-IIa distribution module will be successfully integrated into MultiBUGS and can be accessed at: https://github.com/elianaputrir/MultiBUGS-MSNBurr2a.

3.4. Module testing

The module testing phase consisted of five steps: testing of doodle, validation of probability values, validation of the simulation data density plot, validation of simulation data parameter estimation results, and comparison of DIC values. This testing ensured that the program ran correctly and produced the expected results without any detected errors.

Testing of doodle was conducted by running the MSNBurr-IIa distribution with the following prior distributions for each parameter: normal$(0;\ 1.0 \times 10^{-6})$ for parameter μ, gamma (1000; 1000) for parameter τ, and uniform(0; 10) for parameters α. Figure 26 visualizes the MSNBurr-IIa distribution model. Based on this figure, the added MSNBurr-IIa distribution module functioned correctly and ran without any errors or warnings detected by MultiBUGS.

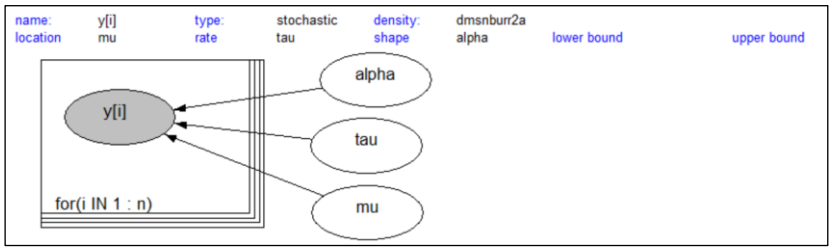

Figure 26. Visualization of trial model with doodle

Validation of the probability values was performed by comparing the probability density function (PDF) and cumulative distribution function (CDF) calculations of the MSNBurr-IIa distribution obtained from MultiBUGS [14] and neodistr package [19] in R that was published in CRAN. Ten sample points (x) were generated from the MSNBurr-IIa distribution in MultiBUGS using the parameters $\mu = 0, \sigma = 1$, and $\alpha = 0.1$. The PDF and CDF values were then calculated for these sample points using both MultiBUGS and the corresponding functions in the neodistr package (Table 1).

Table 1. Calculation of the probability value (PDF) and cumulative opportunity value (CDF) of the MSNBurr-IIa distribution using MultiBUGS and R functions

X	PDF			CDF		
	MultiBUGS	R	Absolute difference	MultiBUGS	R	Absolute difference
(1)	(2)	(3)	(4)	(5)	(6)	(7)
-4.750	0.03467	0.03467	0.00000	0.3191	0.3191	0.00000
-3.137	0.03750	0.03750	0.00000	0.3774	0.3774	0.00000
-10.65	0.02084	0.02085	0.00001	0.1545	0.1545	0.00000
-4.855	0.03446	0.03446	0.00000	0.3155	0.3154	0.00010
16.340	0.01020	0.01020	0.00000	0.9314	0.9313	0.00010
-8.602	0.02574	0.02574	0.00000	0.2022	0.2022	0.00000
-4.278	0.03359	0.03558	0.00001	0.3357	0.3357	0.00000
- 4.917	0.03433	0.03433	0.00000	0.3133	0.3133	0.00000
-1.452	0.03936	0.03936	0.00000	0.4423	0.4423	0.00000
11.670	0.01857	0.01857	0.00000	0.8655	0.8656	0,00010
Total			0.00002	Total		0.0003

This close agreement between the two methods indicates that the PDF and CDF values calculation in MultiBUGS is accurate. However, it's worth noting that this analysis was limited to 10 sample points. Further validation with a larger sample size could provide additional confidence in the accuracy of the MultiBUGS implementation.

3.5. The Implementation of simulation and real-world data

The simulated data were generated using MultiBUGS with a sample size of 1,000 across four distinct scenarios. The data of four scenarios are visualized using a density plot, as shown in Figure 27. This plot displays the distribution of the simulated data for each scenario. The x-axis represents the simulation data, and the y-axis represents the density. The shape of each curve reflects the probability density function of the MSNBurr-IIa distribution under the specific parameters of those scenarios.

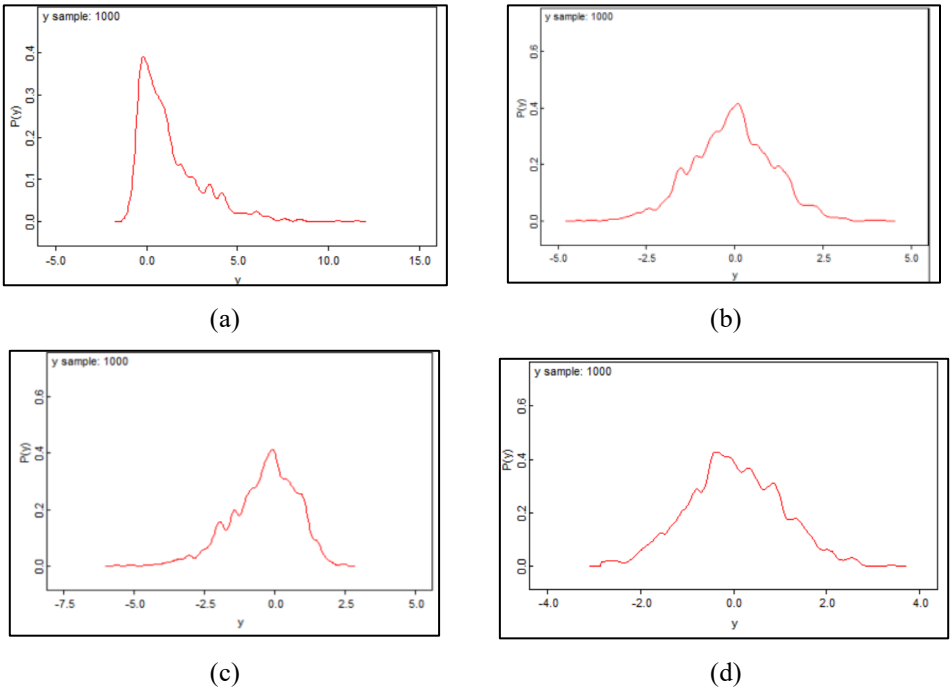

Figure 27. Density plot simulation data (a) scenario 1; (b) scenario 2; (c) scenario 3; (d) scenario 4

Figure 27 illustrates the density plots of four scenarios: three scenarios for the MSNBurr-IIa distribution with varying shape parameters (α), and one scenario as the normal distribution. Figures 27(a) through 27(d) display simulated data for right-skewed (scenario 1, $\alpha = 0.1$), symmetric (scenario 2, $\alpha = 1$), left skewed (scenario 3, $\alpha = 3$), and normal distribution, respectively. This is consistent with the theoretical characteristics of the MSNBurr-IIa distribution, that when $0 < \alpha < 1$ is a right-skewed distribution, $\alpha = 1$ is a symmetrical distribution, and $\alpha > 1$ is a left-skewed distribution. Consequently, the implemented module effectively captures and visualizes the distribution's flexible behavior

Parameter estimation value validation was conducted using four specifically designed simulated data scenarios. These scenarios encompassed various parameter combinations to assess the performance of the estimation procedure under diverse conditions. A simplified "trial model" was initially used for parameter estimation. This model focused on the core components of the MSNBurr-IIa distribution to ensure efficient convergence and accurate estimation of the key parameters. The following Markov Chain Monte Carlo (MCMC) settings were employed for parameter estimation:

Table 2. Estimated value of simulation data parameters

Scenario	Parameter	Simulation Parameter Value	Estimation MSNBurr-IIa parameter	Credible Interval	
				2.5%	97.5%
1	μ	0.0	-0.0397	-0.0976	0.0253
	τ	1.0	1.0050	0.9634	1.0490
	α	0.1	0.1054	0.0102	0.1185
2	μ	0.0	0.0154	-0.1075	0.1100
	τ	1.0	0.9738	0.9359	1.0130
	α	1.0	1.1730	0.9560	1.5210
3	μ	0.0	0.0207	-0.0839	0.1176
	τ	1.0	0.9711	0.9333	1.0090
	α	3.0	4.4510	2.5680	7.8240
4	μ	0.0	0.0598	-0.0463	0.1628
	τ	1.0	1.0450	0.9004	1.0880
	α	-	1.1530	0.8593	1.5330

1. Burn-in: 1,000 iterations. This initial period was discarded to remove samples that may not represent the target distribution.
2. Thinning: 1. Every sample from the chain was kept.
3. Refresh: 100. The chain was refreshed every 100 iterations to improve mixing.
4. Iterations: 10,000. A total of 10,000 iterations were run, resulting in 9,000 samples for analysis after the burn-in period. Table 2 Estimated value of simulation data parameters

The average time needed to estimate the four scenarios was 120.3 seconds. Table 2 shows the values of the three previously determined parameters, all falling within the 95% credible interval. This indicates that the developed MSNBurr-IIa distribution module can be used for parameter estimation and produces results as expected.

The Deviance Information Criterion (DIC) is used as an indicator for Bayesian model evaluation, such as in MultiBUGS. The model with the lowest DIC is generally preferred, as it suggests a better fit to the data. The MSNBurr-IIa distribution model's DIC will be compared to that of the normal distribution model when estimating parameters for the four simulated data scenarios.

Table 3. DIC value of MSNBurr-IIa distribution and normal distribution

Scenario	DIC value		Δ DIC
	MSNBurr-IIa	Normal	
1	3,461.0	4,202.0	741
2	3,128.0	3,140.0	12
3	3,088.0	3,169.0	81
4	2,882.0	2,858.0	-24

Table 3 shows that the Deviance Information Criterion (DIC) of the MSNBurr-IIa distribution was lower than that of the normal distribution in scenarios 1, 2, and 3. However, in scenario 4, the DIC of the MSNBurr-IIa distribution was higher than that of the normal distribution. This indicates that the MSNBurr-IIa distribution generally provided a better fit to the non-normal simulated data, with the greatest improvement in DIC observed for the right-skewed data in scenario 1.

The normal distribution outperformed the MSNBurr-IIa distribution in scenario 4 because the MSNBurr-IIa distribution tends to have heavier tails than the normal distribution, even when the overall shape is symmetric. This characteristic may lead to a slightly worse fit when the true data-generating process has lighter tails [1]. The testing process confirmed the consistency between the data characteristics and the estimation results. MultiBUGS had detected no errors in any of the testing steps. Therefore, it can be concluded that the MSNBurr-IIa distribution module created and added to MultiBUGS functions correctly and produces accurate results.

The MSNBurr-IIa distribution module and its source code have been compiled into a single MultiBUGS program available on GitHub https://github.com/elianaputrir/MultiBUGS-MSNBurr2a/blob/main/MultiBUGS2.zip. Users must install Microsoft MPI before running the compiled MultiBUGS program from the zip file.

The histogram of the economic growth rate data for districts/cities in 2021 reveals a right-skewed distribution (Figure 28). This indicates that most districts/cities have lower economic growth rates, with a few outliers exhibiting much higher rates. This observation is supported by the skewness test, which yielded a skewness value of 18.8663, significantly greater than 0, confirming the right-skewed nature of the distribution. Furthermore, the Anderson-Darling normality test confirms that the data is not normally distributed (p-value < 0.05). This non-normality suggests that standard statistical methods that assume normality may not be appropriate for analyzing this data, and alternative methods may need to be considered.

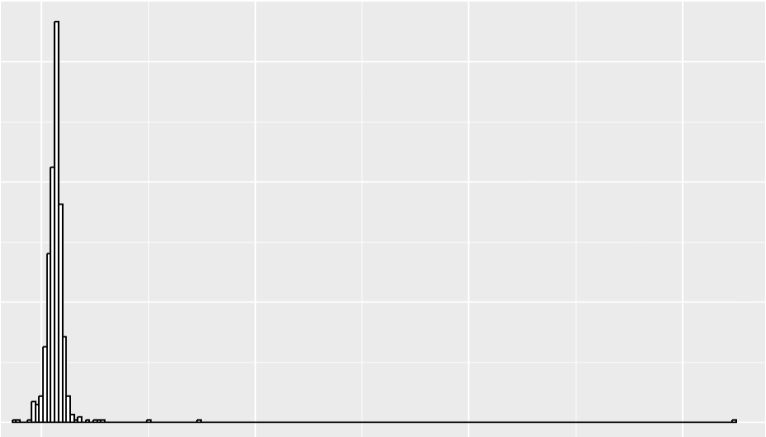

Figure 28. Histogram of district/city economic growth rate data for 2021

Parameter estimation was performed using 1,000 burn-in, 1 thin, 100 refresh, 10,000 iterations, and 9,000 samples. The prior for each parameter is normal (0; 1,0E-6) for μ, gamma (1000; 1000) for τ, and uniform (0,1; 10) for α.

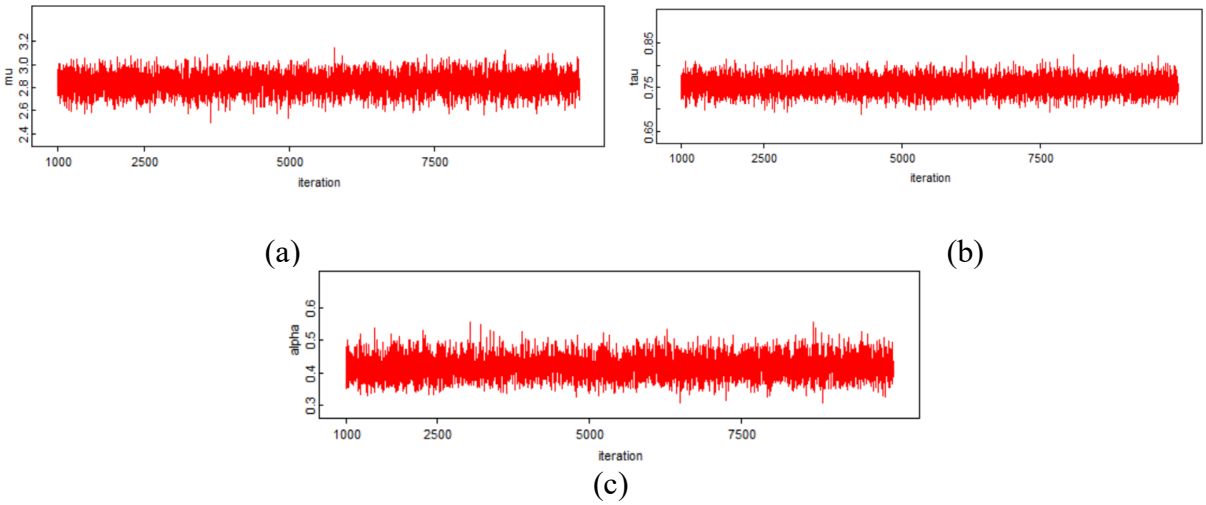

Figure 29. Trace plot of the estimated parameter of economic growth rate data for 2021: (a) $\boldsymbol{\mu}$; (b) $\boldsymbol{\tau}$; (c) $\boldsymbol{\alpha}$

Based on the trace plot of the 9.000 generated samples that were shown in Figure 29, the samples produced in the MCMC output are like fat caterpillars. It is shown that the parameter estimation process has reached a convergent condition.

Table 4. Summary of parameter estimates for the rate of economic growth data in 2021

Parameter	Estimation	Confidence Interval	
		2.5%	97.5%
μ	2.8360	2.6720	2.9940
τ	0.7541	0.7193	0.7908
α	0.4182	0.3567	0.4861

The parameters of the MSNBurr-IIa distribution for the economic growth rate data in 2021 were estimated using Bayesian inference with MCMC sampling in MultiBUGS. The estimation process took 49.938 seconds. Convergence of the MCMC chains was assessed using standard diagnostic tools to ensure reliable parameter estimates. Using the parameters estimated in Table 4, the mean, variance, skewness, and kurtosis of the MSNBurr-IIa distribution were derived and presented in Table 5. These characteristics provide a comprehensive summary of the distribution's shape and properties, which can be used to further analyze and interpret the economic growth rate data.

Table 5 shows that the estimated skewness of the fitted MSNBurr-IIa distribution is 1.0810, indicating that the distribution is right-skewed. This is consistent with the value of alpha in Table 5,

which is less than 1. This finding aligns with the right-skewed shape observed in the histogram of the economic growth rate data. The agreement between the estimated skewness and the visual representation of the data suggests that the MSNBurr-IIa distribution effectively captures the characteristics of the economic growth rate data for regencies in Indonesia.

Table 5. Characteristics of the MSNBurr-IIa distribution of economic growth data for 2021

Characteristics	Characteristic Value
Mode	3.4050
Variance	2.7789
Mode	2.8240
Skewnes	1.0810
Kurtosis	2.8590

4. Conclusion

This study successfully integrated the MSNBurr-IIa distribution into MultiBUGS by creating and compiling a dedicated distribution module. Rigorous testing confirmed the module's functionality and accuracy. The testing process included: ensuring correct computation of the CDF and PDF, confirming the module's ability to create doodle without errors, comparing the outputs with the expected characteristics of the MSNBurr-IIa distribution, assessing the model's fit against the normal distribution. The results consistently demonstrated the module's correct implementation and the accuracy of its calculations.

The MSNBurr-IIa distribution was then applied to analyze the economic growth rate data for regencies in Indonesia in 2021. The analysis revealed that the MSNBurr-IIa distribution effectively captured the characteristics of the data, particularly its right-skewness, as evidenced by the consistency between the estimated skewness, the histogram, and the skewness indicator. This successful application highlights the potential of the MSNBurr-IIa distribution for modeling and analyzing skewed data in various fields.

Future research can explore several directions to further enhance the applicability and utility of the MSNBurr-IIa distribution. One potential area is extending its integration into other Bayesian modeling platforms such as pyMC for broader accessibility. Additionally, further studies can investigate the performance of the MSNBurr-IIa distribution in different real-world datasets, particularly in financial risk modeling, survival analysis, and climate data analysis. Moreover, conducting simulations across various sample size scenarios—ranging from small-scale samples to large-scale datasets—would provide deeper insights into the model's robustness and efficiency.

Ethics approval

The study was conducted in accordance with the ethical guidelines, and informed consent was obtained from all individual participants included in the study.

Acknowledgments

In this study, we would like to express our gratitude to Politeknik Statistika STIS, and Universitas Padjadjaran for the support and resources they have provided. We would like to thank BPS-Statistics Indonesia for the publication of data.

Competing interests

All the authors declare that there are no conflicts of interest.

Funding

The research received no external funding.

Underlying data

Derived data supporting the findings of this study are available from the corresponding author on request.

Credit Authorship

Eliana Putri Ramadani: Conceptualization, Methodology, Data Collection, Data Analysis, Writing–Original Draft, Software Development. **Achmad Syahrul Choir:** Conceptualization, Methodology, Manuscript Review, Research Advisor, Software Development. **Anindya Apriliyanti Pravitasari:** Editing, Writing–Review. **Joynabel Paraguison:** Editing, Writing–Review.

References

[1] A. S. Choir, "The new neo-normal distributions and their properties", Unpublished Dissertation, Institut Teknologi Sepuluh Nopember, Surabaya, 2020.
[2] G. E. P. Box and G. C. Tiao, *Bayesian inference in statistical analysis*. Reading, Massachusetts: Addison Wesley, 1973.
[3] A. Azzalini, "A class of distributions which includes the normal ones", *Scand. J. Stat.*, vol. 12, no. 2, pp. 171–178, 1985.
[4] M. Jorge and Iglewicz Boris, "Some properties of the Tukey g and h family of distributions", *Commun. Stat. - Theory Methods*, vol. 13, no. 3, pp. 353–369, 1984, doi: 10.1080/03610928408828687.
[5] J. A. Jiménez, V. Arunachalam, and G. M. Serna, "A generalization of tukey's g-h family of distributions", *J. Stat. Theory Appl.*, vol. 14, no. 1, pp. 28–44, 2017, doi: 10.2991/jsta.2015.14.1.3.
[6] N. Eugene, C. Lee, and F. Famoye, "Beta normal distribution and its applications", *Commun. Stat. - Theory Methods*, vol. 31, pp. 497–512, 2002.
[7] M. A. Aljarrah, C. Lee, and F. Famoye, "On generating T-X family of distributions using quantile functions", *J. Stat. Distrib. Appl.*, vol. 1, no. 1, p. 24, Dec. 2014, doi: 10.1186/2195-5832-1-2.
[8] T. Amemiya, "Qualitative response models: a survey", *J. Econ. Lit.*, vol. 19, no. 4, pp. 1483–1536, 1981.
[9] N. Iriawan, *Computationally intensive approaches to inference in neo-normal linear models*. Perth, Australia: Disertasi Doktor, Curtin University of Technology, 2000.
[10] Irving. W. Burr, "Cumulative frequency functions", *Ann. Math. Stat.*, vol. 13, pp. 215–232, 1942, doi: 10.1214/aoms/1177731607.
[11] A. Choir, N. Iriawan, B. Ulama, and M. Dokhi, "MSEPBurr distribution: properties and parameter estimation", *Pak. J. Stat. Oper. Res.*, vol. 15, no. 1, pp. 179–193, 2019.
[12] A. Ragab, J. R. Green, and M. C. K. Tweedie, "The burr type II distribution: properties, order statistics", *Microelectron. Reliab.*, vol. 31, pp. 1181–1191, 1991, doi: 10.1016/0026-2714(91)90309-U.
[13] W. M. Bolstad, *Introduction to bayesian statistics*. New York: John Wiley and Sons, 2007.
[14] R. J. B. Goudie, R. M. Turner, D. D. Angelis, and A. Thomas, "MultiBUGS: a parallel implementation of the BUGS modeling framework for faster bayesian inference", *J. Stat. Softw.*, vol. 95, pp. 1–20, Oct. 2020, doi: 10.18637/jss.v095.i07.
[15] M. K. Cowles, "Review of WinBUGS 1.4", *Am. Stat.*, vol. 58, no. 4, pp. 330–336, Nov. 2004, doi: 10.1198/000313004X8515.
[16] E. Štrumbelj *et al.*, "Past, present and future of software for bayesian inference", *Stat. Sci.*, vol. 39, no. 1, pp. 46–61, Feb. 2024, doi: 10.1214/23-STS907.
[17] D. Lunn, C. Jackson, N. Best, A. Thomas, and D. Spiegelhalter, *The BUGS book: a practical introduction to bayesian analysis*. Boca Raton, FL: CRC Press, 2013.

[18] J. Annis, B. J. Miller, and T. J. Palmeri, “Bayesian inference with Stan: a tutorial on adding custom distributions”, *Behav. Res. Methods*, vol. 49, no. 3, pp. 863–886, Jun. 2017, doi: 10.3758/s13428-016-0746-9.

[19] A. S. Choir, A. Faoziah, and N. Iriawan, *Neodistr: neo-normal distribution*. [Online]. Available: doi:10.32614/CRAN.package.neodistr, 2024

[20] O. Abril-Pla, V. Andreani, C. Carroll, L. Dong, C. J. Fonnesbeck, M. Kochurov, R. Kumar, J. Lao, C. C. Luhmann, O. A. Martin, M. Osthege, R. Vieira, T. Wiecki, and R. Zinkov, “PyMC: a modern and comprehensive probabilistic programming framework in Python,” *PeerJ Computer Science*, vol. 9, p. e1516, 2023, doi: 10.7717/peerj-cs.1516.