STRESS-STRENGTH RELIABILITY WITH LOG-LINDLEY AND BETA DISTRIBUTION

A Dissertation Submitted to the Department of Statistics, Dibrugarh University for partial fulfillment of the Degree of Master of Science in Statistics

Submitted by

Namrata Bhattacharjee

Roll No.: 19192006

Registration No.: S1626048

Under the guidance of

Mr. Aniket Biswas

Assistant Professor

DEPARTMENT OF STATISTICS

Dibrugarh University

Dibrugarh-786004(Assam)

Mr. Aniket Biswas
Assistant Professor, Dept. of Statistics,
Dibrugarh University,
Assam, India

Email: aniket_stat@dibru.ac.in

Certificate

The undersigned has supervised the Dissertation on the topic entitled "Stress-strength reliability with log-Lindley and beta distribution" prepared by Namrata Bhattacharjee, a student of M.Sc. in Statistics, Dibrugarh University. The Dissertation as a whole or part has not been submitted to any other University for a Degree.

I found her analysis and findings quite satisfactory and I certify that it fulfills the requirement for awarding a Degree of Master of Science in Statistics of this University, provided the candidate has been considered duly in other examination for awarding the Degree.

I wish success in every sphere of life.

Date:

Place: Dibrugarh

(Mr. Aniket Biswas)

Acknowledgment

At the very out-set, I express my sincere gratitude and gratefulness to Mr. Aniket Biswas, Department of Statistics for his valuable guidance, inspiration, active supervision and constructive suggestion which enable me to complete this report.

I will be failing in my duty if I don't acknowledge the help, co-operation and encouragement given to me by my fellow mates.

My sincere thanks also goes to everyone who have helped and encouraged me either directly or indirectly in bringing out the dissertation in the present form.

Date:

Place:

(Namrata Bhattacharjee)

M. Sc. 4th semester

Department of Statistics,

Dibrugarh University

Contents

Sections **Page No.**

Title: Stress-strength reliability with log-Lindley and beta distribution

Abstract:

This study aims to estimate the stress-strength reliability, $R = P(X<Y)$ when X follows log-Lindley distribution with parameters (σ, π) and Y follows beta distribution with parameters (α, β). X and Y are assumed to be independent. We obtain value of R through numerical integration and Monte-Carlo integration method. Estimates of the parameters and R are obtained by maximum likelihood estimation method. We observe the biases and mean squared errors to validate the consistency and efficiency properties of the parameters and R by extensive simulation study with changes in parameter values and sample sizes. We see that estimates perform better with increasing sample sizes.

Keywords: Numerical integration, Monte-Carlo integration, simulation, bias, mean squared error.

1. Introduction

The Log–Lindley distribution is a continuous probability model. G´omez-D´eniz et al.(2014)[5] introduced the Log–Lindley distribution. A random variable Y is said to have a Log–Lindley distribution if its probability density function (pdf) is

$$f(X|\sigma,\pi) = \sigma\{\pi + \sigma(\pi-1)\log(x)\}\ ;\ 0 < x < 1,\ \sigma > 0,\ 0 \leq \pi \leq 1 \qquad (1)$$

This new distribution that depends on two parameters can be considered as an alternative to the classical beta distribution which supports on (0, 1) and refer the same as LL(σ, π). If X ∼ LL(σ, π) then distribution function of X is given by

$$F(X|\sigma,\pi) = \{1 + \sigma(\pi-1)\log(x)\}x^{\sigma}\ ;\ 0 < x < 1,\ \sigma > 0,\ 0 \leq \pi \leq 1 \qquad (2)$$

The beta distribution is a family of continuous probability distributions defined on the interval [0, 1] parameterized by two positive shape parameters, denoted by α and β, that appear as exponents of the random variable and control the shape of the distribution. The probability density function (pdf) of the beta distribution, Beta(α, β) for $0 \leq x \leq 1$, and shape parameters α, β >0

$$f(X|\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)} \qquad (3)$$

The beta function, B, is a normalization constant to ensure that the total probability is 1. The cumulative distribution function is

$$F(X|\alpha,\beta) = \frac{B(x;\ \alpha,\beta)}{B(\alpha,\beta)} = I_X(\alpha,\beta) \qquad (4)$$

where B(x; α, β) is the incomplete beta function and $I_x(\alpha, \beta)$ is the regularized incomplete beta function.

Stress-strength reliability is a measure to compare the lifetimes of two systems. Many authors took interest in 'stress-strength reliability' for research and study. Biswas & Chakraborty(2021) [3] addressed the issue of parameter estimation from a Bayesian perspective and study relative performance of different estimators through extensive simulation of experiments also significant emphasis is given to the estimation of stress–strength reliability unit-lindley distribution employing classical as well as Bayesian approach. Biswas et al.(2021)[4] Addressed Explicit structure of stress–strength reliability of log-lindley distribution and its inference under both classical and Bayesian set-ups. Results showed marked improvement with Bayesian approach over classical given reasonable prior

information. Kizilashan & Nadar(2018)[9] compared the reliability estimators of bivariate Kumaraswamy distribution by using the estimated risks through Monte Carlo simulations. Other than the study of reliability of distributions that support unit interval there are many studies on distributions such as normal, exponential, log-normal, Weibull, etc. Interested ones may follow up with Asgharzadeh et al.(2011)[1], Nkemnole et al.(2017)[11] and more. Much of the work of many authors presupposes that both random variables have distributions belonging to the same family and more significantly it assumes independence between them.

Wolfe & Hogg(1971)[12] stated that the values of P(X<Y) are more practical in many fields especially in the medical profession than the equivalent statements about $(\mu_1 - \mu_2)/\sigma$ (under the normal assumptions) and point out thatP(X < Y) can be estimated under many distributional assumptions (not only the normality) which makes more sense when using normal distributions are inappropriate. Deriving theoretical expressions for P(X < Y) and its modifications and extensions under various distributional assumptions were found to be challenging, but also estimation of these probabilities based on samples were efficient in deriving approximations to variances and confidence bounds. Halperin et al.(1987)[6] stated similar abstraction about suitability of P(X<Y). Bilikam(1985)[2] discussed intriguing thought in his paper, the strength is "necessarily conditioned on the stress because the physical realization of strength is found only when stress is applied". He studied simple models assuming time dependent parameter values and X and Y to be stochastically independent.

Stress strength reliability model is used in many applications of physics and engineering such as strength failure and the system collapse. Application of it in clinical trials for medical research purposes is in fast growing stage and application of relation between stress strength models and quality control are emerging with time. Stress strength models have roles in other different fields such as Bioinformatics, Environment, Behavioral/Psychology, Agriculture, etc.

The layout of this report is as follows: In Section 2 we have given the structure of stress-strength reliability, R for Log-Lindley and Beta distribution. We have figured out that the closed form expression of R is difficult if not impossible to find. In Section 3 we elaborate use of numerical integration method to our study. In Section 4, Monte-Carlo integration method and its application is elucidated. In Section 5 and 6, we obtain Maximum likelihood estimators (MLE) of the parameters of Log-Lindley and Beta distribution respectively. In Section 7, we obtain MLE of R taking into consideration the Invariance property of MLE. Section 8 was dedicated to understand the method of random sample generation. Simulation study and some results and observations are presented in Section 9. Finally, with brief discussion on the study we concluded in Section 10. Sample codes are provided in the Appendix.

2. Stress-strength reliability

Reliability is then defined as the probability of not failing: P{X < Y). Assessment of "reliability" of a "component" in terms of random variables X representing "stress" experienced by the component and Y representing the "strength" of the component available to overcome the stress. A variable Y is said to be stochastically larger than a variable X if the cumulative distribution function (CDF) of Y is never greater than that of X, i.e. using the standard notation Fy(x) < Fx{x) for all x. An immediate consequence of this relation is that P{X < Y) > 1/2 since

$$P(X < Y) = \int_{-\infty}^{\infty} F_X(x) dF_Y(x) = R \geq \int_{-\infty}^{\infty} F_Y(x) dF_Y(x) = \frac{1}{2} \quad (5)$$

We equate P(X<Y) to 'R' in (5). In our study, X = (X_1, X_2, X_3,…, X_m) of size m follows LL(σ, π) and Y = (Y_1, Y_2, Y_3,…, Y_n) of size n follows Beta(α, β). The structure of R in our case, taking the CDF of X and PDF of Y given in (2) and (3) respectively and assuming independence between X and Y will be as

$$R = \int_0^1 (1 + \sigma(\pi - 1)\log(x)) x^{\sigma} x^{(\alpha-1)} (1-x)^{(\beta-1)} \frac{1}{B(\alpha,\beta)} dx \quad (6)$$

The closed form expression of R with finite number of standard operations is difficult to obtain if not impossible. We observe that R contains four unknown parameters σ, π, α and β. So, let's say R = f(x; σ, π, α, β) as a function of four parameters. The function f is continuous over (0,1). Due to unavailability of closed form expression of R we can obtain the value of R with 'Numerical Integration' method and 'Monte-Carlo Integration' method where the values of the four parameters and sample sizes will be provided before handedly.

3. Numerical Integration

Numerical integration is the approximate computation of an integral using numerical techniques. The basic problem considered by numerical integration is to compute an approximate solution to a definite integral. It is an approximation and will not yield an exact answer as with analytical integration. Error analysis is a very important aspect in numerical integration. However there is a way to approximate the integration by dividing the function into small intervals and approximating the area. One common method taught is a Rieman sum where rectangles are used to approximate a definite integral. While this is quite simple, it is usually the case that a large number of rectangles is needed to get acceptable accuracy. A similar method is better, the trapezoidal rule. This technique is a much more accurate way to approximate area beneath a curve. To construct the trapezoids, you mark the height of the function at the beginning and end of the width interval, then connect the two points. Another area approximating tool is Simpson's Rule. Geometrically, it creates tiny parabolas to wrap closer around the function we're approximating. The formula is similar to the Trapezoidal Rule but we can only use an even number of subintervals.

We have used coding in R programming software for finding the value of 'R' and the algorithm is as follows:

Algorithm 1:

Step 1: Construct R as a function with four unknown parameters σ, π, α and β as given in equation (6).
Step 2: Specify the values of the four parameters.

Step 3: Integrate the function, R from 0 to 1.

In our case of R , we failed to analytically integrate the function. Even if a closed integration formula exists, it might still not be the most efficient way of calculating the integral. So, we used numerical integration to find the value. To integrate R we used the function 'integrate'. The code for generating R by numerical integration method is provided in the Appendix.

4. Monte-Carlo integration

Monte Carlo integration is a technique for integration using random numbers. Monte Carlo randomly chooses points at which the integrand is evaluated. Monte Carlo integration employs a non-deterministic approach, each realization provides a different outcome. Consider the d-dimensional integral of a function f over the unit $[0, 1]^d$

$$\int f(x)dx = \int_{x_1=0}^{x_2=1} \dots \int_{x_d=0}^{x_d=1} f(x_1, \dots, x_d)dx_{1,\dots,}dx_d$$

The integral can be interpreted as the expectation E{f(X)} of the random variable f(X), where X is an R^d -valued random variable with a uniform distribution over $[0, 1]^d$, the components $X_1, \dots, X_d$ are independent and identically uniformly distributed over [0, 1]. The $X_1, \dots, X_d$ are random numbers. The Monte Carlo approximation of the integral is given by

$$S_n = \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

Monte Carlo estimation is nothing else than a sample mean, only, we substitute the population for a real-value arbitrary function. For this reason, Monte Carlo estimations and sample means share the same properties.

To generate value of R in our case, we state the algorithm,

Algorithm 2:

Step 1: Construct R as a function with four unknown parameters σ, π, α and β as given in equation (6).
Step 2: Specify the values of the four parameters.

Step 3: Generate random sample from uniform distribution of size 25,000.

Step 4: For each sample observation, find the value of R and store it in **value**.

Step 5: Obtain mean of **value.**

To generate random sample from uniform distribution we used 'runif' function. The code for generating R by Monte-Carlo integration method is given in the Appendix.

Monte Carlo estimation help us to generate sequences of random numbers which we can use to evaluate R for random values of x over the desired interval [0, 1] where the parameters are fixed ahead. Thus we can obtain R in a non-deterministic way.

5. Maximum likelihood estimation of Log-Lindley distribution

Let $X_1, \ldots, X_m$ be a random sample of size m from a LL(σ, π) and $x_1, \ldots, x_m$ the observed values. The PDF is given in (1). Assume that σ and π are both unknown. The likelihood function is

$$L(x;\sigma,\pi) = \sigma^m \prod_{i=1}^{m}\{\pi + \sigma(\pi-1)\log(x_i)\}\left(\left(\prod_{i=1}^{m} x_i\right)^{\sigma-1}\right.$$

And the log-likehood function is

$$\log L(\sigma,\pi) = n\log(\sigma) + (\sigma-1)\sum_{i=1}^{m}\log(x_i) + \sum_{i=1}^{m}\log\{\pi + \sigma(\pi-1)\log(x_i) \quad (7)$$

The ML estimates of σ and π are the pair of values $(\hat{\sigma},\hat{\pi})$ that maximizes (6) or minimizes the negative log likelihood function. Hence, they satisfy the following system of equations:

$$\frac{\partial}{\partial\sigma}\log L(\sigma,\pi) = \frac{m}{\sigma} + \sum_{i=1}^{m}\log(x_i) + \sum_{i=1}^{m}\frac{(\pi-1)\log(x_i)}{\pi + \sigma(\pi-1)\log(x_i)} = 0 \quad (8)$$

$$\frac{\partial}{\partial\pi}\log L(\sigma,\pi) = \sum_{i=1}^{m}\frac{1 + \sigma\log x_i}{\pi + \sigma(\pi-1)\log x_i} = 0 \quad (9)$$

However, the ML estimates of the parameters cannot be obtained by solving numerically the system of equations (8) and (9) because it has multiple real roots and it is not clear how to choose a root to get accurate estimates of the parameters. Therefore to maximize (7) we have used BFGS algorithm. The feasible starting point (σ_f, π_f), the initial guess for the coefficient are considered. Results can vary based on these values as the function can hit local minima. We note that for π = 0 and π = 1 we have,

$$E(X;\sigma,0) = \left(\frac{\sigma}{\sigma+1}\right)^2$$

$$E(X;\sigma,1) = \frac{\sigma}{\sigma+1}$$

Then using method of moments we obtain following estimates of σ for π = 0 and π = 1 respectively,

$$\hat{\sigma}_0 = \frac{\sqrt{\bar{x}}}{1-\sqrt{\bar{x}}}$$

$$\hat{\sigma}_1 = \frac{\bar{x}}{1-\bar{x}}$$

where $\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i$

Therefore we take feasible starting point,

$$\hat{\sigma}_f = \frac{\hat{\sigma}_0 + \hat{\sigma}_1}{2}$$

$$\pi_f = 0.5 \qquad \text{since } \pi \in [0,1]$$

To obtain the maximum likelihood estimate of both the parameters σ and π we have used 'constrOptim' function in R-software. The detailed code is attached in the Appendix.

6. Maximum likelihood estimation of Beta distribution

Let $Y_1, \ldots, Y_n$ be a random sample of size n from a Beta(α, β) and $y_1, \ldots, y_n$ the observed values. The pdf is given in (3). Assume that σ and π are both unknown. The likelihood function is

$$L(y;\ \alpha, \beta) = \frac{\Gamma(\alpha+\beta)^n}{\Gamma(\alpha)\Gamma(\beta)} \prod_{i=1}^{n} y_i^{\alpha-1}(1-y_i)^{\beta-1}$$

And the log likelihhod function, logL(α, β) is given by

$$n\log(\alpha+\beta) - n\log(\alpha) - n\log(\beta) + (\alpha-1)\sum_{i=1}^{n}\log(y_i) + (\beta-1)\sum_{i=1}^{n}\log(1-y_i) \quad (10)$$

The ML estimates of α and β are the pair of values $(\hat{\alpha}, \hat{\beta})$ that maximizes (9) or minimizes the negative log likelihood function. Hence, they satisfy the following system of equations:

$$\frac{\partial}{\partial\alpha}\log L(\alpha,\beta) = n\Psi(\alpha+\beta) - n\Psi(\alpha) + \sum_{i=1}^{n}\log(y_i) = 0 \quad (11)$$

$$\frac{\partial}{\partial\beta}\log L(\alpha,\beta) = n\Psi(\alpha+\beta) - n\Psi(\beta) + \sum_{i=1}^{n}\log(1-y_i) = 0 \quad (12)$$

where Ψ(a) is digamma function, $\Psi(a) = \{\text{derivative of } \Gamma(a)\}/\Gamma(a)$.

Equation (11) and (12) are non-linear equations. To find the maximum likelihood estimates of (10) we use BFGS algorithm. We take the initial values of α and β to be the estimates of method of moments as follows:

$$\hat{\alpha}_0 = \bar{y}\left\{\frac{\bar{y}(1-\bar{y})}{V(\bar{y})} - 1\right\}$$

$$\hat{\beta}_0 = (1-\bar{y})\left\{\frac{\bar{y}(1-\bar{y})}{V(\bar{y})} - 1\right\}$$

where $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$

In our study, to obtain the maximum likelihood estimate of both the parameters α and β without any major computational work, we have used 'univariateML' package in R-software. The framed code is attached in the Appendix.

7. Maximum likelihood estimation of R

If $\hat{\theta}$ is an MLE of θ and if g is a function of θ, then $g(\hat{\theta})$ is an MLE of $g(\theta)$. This is the invariance property of MLE's. This property helps us in the understanding that when the maximum likelihood estimates of the four parameters σ, π, α and β maximizes the log likelihood function of their respective distribution then it naturally maximizes R also as R is the function of all the four parameters together, $\hat{R} = R(\hat{\sigma}, \hat{\pi}, \hat{\alpha}, \hat{\beta})$.

To get the maximum likelihood estimate of R, $\hat{R}$ for different parameter values, we have coded in R-software with the algorithm as follows:

Algorithm 3:

Step 1: Construct R as a function with four unknown parameters σ, π, α and β as given in equation (6).
Step 2: Obtain MLE for the parameters of log-Lindley distribution.

Step 3: Obtain MLE for the parameters of Beta distribution.

Step 4: Extract estimated parameters separately and store them.

Step 5: Execute R as the function of those estimated parameters obtained in Step 4.

The complete code structure is given in the Appendix.

8. Random sample generation

Generating sample of random observation from log-Lindley and Beta distribution is a crucial process in the simulation study (detailed in the next section).

The algorithm used for coding in R-software to generate random observations from log-Lindley distribution is as follows:

Algorithm 4:

Step 1: Generate m random observations from uniform distribution and store it in **u.**

Step 2: Construct the equation obtained from equating distribution function of log-Lindley distribution to **u.**

Step 3: For each observation generated in Step 1, obtain roots of the equation defined in Step 2 for lower and upper endpoints (interval) very close to 0 and 1 respectively, considering the desired accuracy (tolerance) required.

The roots of the equation are obtained using the function 'uniroot'. The algorithm executed in the code is attached in the Appendix.

A transformation method that is applied to generate random observations from Beta distribution can be stated as: If U follows Gamma(r, λ) distribution and V follows Gamma(s, λ) distribution are independent, then X = U/(U+V) follows Beta(r, s) distribution. The algorithm for generating random observations from Beta distribution is as follows:

Algorithm 5:

Step 1: Generate n random observations from Gamma(α, 1) and store it in **u.**

Step 2: Generate n random observations from Gamma(β, 1) and store it in **v.**

Step 3: For each set of **u** & **v** execute x = **u**/(**u**+**v**).

R-software comes with a function 'rbeta' to generate random observations without much computation. The code for executing in R-software is attached in the Appendix.

9. Simulation study

To observe the performance of bias and mean squared error of all the four estimated parameters of the distributions and also of stress-strength parameter *R*, we enacted an extensive simulation study for fixed parameter values and sample sizes in both numerical and Monte-Carlo integration method of evaluating R.

9.1. Simulation Setting

To establish the study, we generated N = 1000 random samples of different sizes of m and n for several values of σ, π, α and β. The following formulae are used in the calculation of bias and mean squared error (MSE) are:

$$\text{Bias}(\hat{\sigma}) = \text{mean}(\hat{\sigma} - \sigma)$$
$$\text{MSE}(\hat{\sigma}) = \text{mean}\{(\hat{\sigma} - \sigma)^2\}$$

Where, $\hat{\sigma}$ is the maximum likelihood estimate of the parameter σ. The formulae remain same for other parameters π, α, β nd also for R. The brief steps for the study is given below:

Algorithm 6:

Step 1: Construct R as a function with four unknown parameters σ, π, α and β as given in equation (6).
Step 2: Evaluate and extract R (as in Algorithm 1 or 2).

Step 3: Initialize bias and MSEs of all parameters to be zero.

Step 4: For N times, generate random sample from both the distributions and store them.

Step 5: Obtain MLEs of the distributions for each generated sample and extract the estimated parameter values.

Step 6: Obtain estimated R.

Step 7: Obtain the deviation between the estimated parameter value and the true value and store it as b.parameter (ex. b.sigma) for every parameter and R.

Step 8: Obtain the square of the deviation of the estimated parameter value and the true value and store it as m.parameter (ex. m.alpha) of every parameter and R.

Step 9: Obtain the mean of b.parameter and m.parameter for each parameter and R.

Step 10: Create a data frame to extract the output of the biases and MSEs..

The required code is attached in the Appendix.

9.2. Simulation results

While carrying out simulation study for various sample sizes and parameter values we have documented the output in two tables containing R values, Bias and MSEs of parameters and R. R obtained with numerical integration and the subsequent simulation is tabulated in Table 1. Table 2 hold output for R obtained from Monte-Carlo integration. The findings are as follows:

Findings of Table 1 (Simulation results using numerical integration) are quite satisfactory, given as follows:

- Biases and MSEs of parameters and R are decreasing with increasing sample sizes (m,n) which shows consistency and efficiency of the maximum likelihood estimators.
- For sample sizes (5,5) and (10,10) we observe greatest bias. Estimated values hugely deviate from true value for these sample sizes.
- For sample size (5,5) we observe greatest variance of the estimates.
- We observe, largest sample sizes (150,150), (100-150) and (100, 100) provide estimates very close to the true value given.
- Sample sizes (150,150), (100-150), (150-100) and (100, 100) provide lowest MSEs.

We obtain non-deterministic R values for Monte-Carlo simulation. Findings of Table 2 (Simulation results using Monte-Carlo integration are quite satisfactory, given as follows:

- Biases and MSEs of parameters and R are decreasing with increasing sample sizes (m,n) which shows consistency and efficiency of the maximum likelihood estimators.
- For sample sizes (5,5) we observe greatest bias.
- For sample size (5,5) we observe greatest MSEs of the estimates.
- We observe, largest sample sizes (150,150), (100-150) and (100, 100) provide estimates very close to the true value given.
- Sample sizes (150,150), (100-150) and (150-100) provide lowest MSEs.

Table 1: Simulation results using numerical integration.

(m, n)	X	Y	R	Bias					MSE				
	(σ, π)	(α, β)		$\hat{\sigma}$	$\hat{\pi}$	$\hat{\alpha}$	$\hat{\beta}$	$\hat{R}$	$\hat{\sigma}$	$\hat{\pi}$	$\hat{\alpha}$	$\hat{\beta}$	$\hat{R}$
(5, 5)	(2.5, 0.2)	(2.5, 5.5)	0.208257	0.029545	0.00214	0.155595	0.340993	0.000086	0.131808	0.029181	0.291235	1.608266	0.001903
	(2.5, 0.5)	(5.5, 2.5)	0.582037	0.815382	-0.17806	7.435473	3.129124	-0.00783	3.630902	0.171101	315.5141	61.61881	0.030436
	(2.5, 0.8)	(0.5, 0.8)	0.245386	1.601288	-0.39847	0.551395	1.074976	-0.01136	8.347346	0.320557	2.932636	8.069487	0.023468
	(1.0, 0.5)	(0.8, 0.5)	0.712323	0.371625	-0.16914	1.301763	0.593717	0.008626	0.827232	0.171120	13.96191	5.067120	0.026283
	(5.0, 0.8)	(1.0, 1.0)	0.194444	3.199298	-0.38018	1.211392	1.201879	-0.00429	42.60761	0.310281	9.696043	13.25543	0.017905
(10, 10)	(2.5, 0.2)	(2.5, 5.5)	0.208257	0.088138	0.023817	1.109621	2.453601	0.001141	0.683038	0.097016	10.01209	39.81314	0.008246
	(2.5, 0.5)	(5.5, 2.5)	0.582037	0.423552	-0.10837	2.271377	0.934311	-0.00283	1.303287	0.148856	30.22113	5.388092	0.015552
	(2.5, 0.8)	(0.5, 0.8)	0.245386	0.898099	-0.29318	0.147498	0.343336	-0.00735	2.831566	0.228923	0.162069	0.858823	0.011306
	(1.0, 0.5)	(0.8, 0.5)	0.712323	0.151274	-0.11476	0.327036	0.168935	0.006432	0.201043	0.151552	0.835740	0.172590	0.011407
	(5.0, 0.8)	(1.0, 1.0)	0.194444	1.950419	-0.29379	0.394589	0.410144	-0.01199	13.64706	0.228212	1.091424	0.989007	0.008856
(25, 25)	(2.5, 0.2)	(2.5, 5.5)	0.208257	0.016061	0.014657	0.339494	0.848948	-0.00191	0.264563	0.060148	0.823588	4.652401	0.003464
	(2.5, 0.5)	(5.5, 2.5)	0.582037	0.079638	-0.01571	0.874267	0.368452	0.003005	0.518402	0.106081	5.134647	0.910196	0.006914
	(2.5, 0.8)	(0.5, 0.8)	0.245386	0.436064	-0.15187	0.054554	0.087564	0.001351	1.036901	0.126782	0.025385	0.076210	0.004181
	(1.0, 0.5)	(0.8, 0.5)	0.712323	0.053213	-0.01988	0.104481	0.047438	0.003381	0.088673	0.100254	0.099489	0.024353	0.004979
	(5.0, 0.8)	(1.0, 1.0)	0.194444	0.958747	-0.16579	0.100163	0.109169	-0.00406	4.151532	0.127993	0.107629	0.116626	0.003376
(50, 50)	(2.5, 0.2)	(2.5, 5.5)	0.208257	0.011457	0.006061	0.144369	0.327852	0.000096	0.129219	0.033723	0.279507	1.491501	0.001946
	(2.5, 0.5)	(5.5, 2.5)	0.582037	0.020892	0.008729	0.367839	0.158792	-0.00058	0.288968	0.070132	1.653897	0.299904	0.002975
	(2.5, 0.8)	(0.5, 0.8)	0.245386	0.254485	-0.09929	0.026132	0.049936	-0.00015	0.502407	0.078052	0.010003	0.034321	0.002191
	(1.0, 0.5)	(0.8, 0.5)	0.712323	0.016008	-0.00104	0.041141	0.018041	0.002361	0.043432	0.064909	0.033175	0.009193	0.002195
	(5.0, 0.8)	(1.0, 1.0)	0.194444	0.488428	-0.08884	0.056798	0.067759	-0.00387	2.072606	0.071596	0.049294	0.048538	0.001804
(100, 100)	(2.5, 0.2)	(2.5, 5.5)	0.208257	0.024575	-0.00439	0.059417	0.142652	-0.00148	0.068001	0.016267	0.120987	0.649828	0.000877
	(2.5, 0.5)	(5.5, 2.5)	0.582037	-0.00632	0.011761	0.193151	0.074069	0.000784	0.148769	0.042030	0.730351	0.127870	0.001631
	(2.5, 0.8)	(0.5, 0.8)	0.245386	0.108012	-0.04201	0.010975	0.022948	-0.00005	0.268017	0.047339	0.003994	0.014229	0.001074
	(1.0, 0.5)	(0.8, 0.5)	0.712323	0.004643	-0.00398	0.020780	0.010431	0.002336	0.023002	0.040971	0.014311	0.004222	0.001255
	(5.0, 0.8)	(1.0, 1.0)	0.194444	0.286025	-0.05702	0.015569	0.019261	0.000029	1.173359	0.052893	0.018873	0.019881	0.000889
(150, 150)	(2.5, 0.2)	(2.5, 5.5)	0.208257	0.004829	0.000487	0.049343	0.112871	0.000148	0.042601	0.010991	0.082934	0.431621	0.000649
	(2.5, 0.5)	(5.5, 2.5)	0.582037	0.002209	0.004952	0.109258	0.058008	-0.00102	0.109219	0.028901	0.448400	0.087843	0.001075
	(2.5, 0.8)	(0.5, 0.8)	0.245386	0.062145	-0.02378	0.008726	0.012184	0.001388	0.199485	0.038205	0.002703	0.008499	0.000723
	(1.0, 0.5)	(0.8, 0.5)	0.712323	-0.00201	0.006832	0.018348	0.006960	0.001972	0.016768	0.028273	0.009251	0.002639	0.000846
	(5.0, 0.8)	(1.0, 1.0)	0.194444	0.150865	-0.02728	0.018554	0.013422	0.000958	0.788240	0.036706	0.013177	0.013316	0.000654
(15, 10)	(2.5, 0.2)	(2.5, 5.5)	0.208257	0.100285	0.011678	0.872604	2.224150	-0.00758	0.460177	0.072764	4.166038	26.04871	0.007266
	(2.5, 0.5)	(5.5, 2.5)	0.582037	0.291129	-0.07682	2.046659	0.844041	0.001208	0.871722	0.130931	22.67434	3.953921	0.011349
	(2.5, 0.8)	(0.5, 0.8)	0.245386	0.691914	-0.21815	0.141664	0.317729	-0.01012	1.937322	0.177938	0.133870	0.654663	0.010201
	(1.0, 0.5)	(0.8, 0.5)	0.712323	0.105789	-0.06168	0.318598	0.152447	0.003369	0.139771	0.130276	0.605791	0.155070	0.011243
	(5.0, 0.8)	(1.0, 1.0)	0.194444	1.335912	-0.24608	0.370345	0.353365	-0.00191	7.159201	0.192028	0.979288	0.794541	0.008685

Table 1: (continued).

(m, n)	X	Y	R	Bias					MSE				
	(σ, π)	(α, β)		$\hat{\sigma}$	$\hat{\pi}$	$\hat{\alpha}$	$\hat{\beta}$	$\hat{R}$	$\hat{\sigma}$	$\hat{\pi}$	$\hat{\alpha}$	$\hat{\beta}$	$\hat{R}$
(25, 10)	(2.5, 0.2)	(2.5, 5.5)	0.208257	0.031912	0.010471	1.027376	2.442287	-0.00310	0.256251	0.058472	5.491823	29.34877	0.005167
	(2.5, 0.5)	(5.5, 2.5)	0.582037	0.131771	-0.03302	2.109632	0.905253	0.000347	0.505731	0.098797	23.94643	4.267361	0.008814
	(2.5, 0.8)	(0.5, 0.8)	0.245386	0.470309	-0.18657	0.153185	0.331597	-0.00499	0.977385	0.134726	0.142718	0.681255	0.010366
	(1.0, 0.5)	(0.8, 0.5)	0.712323	0.036548	-0.00582	0.335383	0.168113	0.002464	0.074686	0.098254	0.859335	0.196603	0.009609
	(5.0, 0.8)	(1.0, 1.0)	0.194444	0.919667	-0.16041	0.369919	0.387307	-0.00780	4.200799	0.125380	0.683140	0.920156	0.007652
(10, 15)	(2.5, 0.2)	(2.5, 5.5)	0.208257	0.108558	0.02312	0.550485	1.321935	-0.00334	0.656057	0.093946	1.892886	10.63752	0.008132
	(2.5, 0.5)	(5.5, 2.5)	0.582037	0.364897	-0.09721	1.291381	0.554321	0.000548	1.278111	0.143780	11.51962	2.176616	0.012378
	(2.5, 0.8)	(0.5, 0.8)	0.245386	0.946286	-0.28258	0.082830	0.197514	-0.01021	2.963588	0.220917	0.054922	0.240385	0.008407
	(1.0, 0.5)	(0.8, 0.5)	0.712323	0.134430	-0.09256	0.179743	0.083361	0.009368	0.197426	0.141835	0.211890	0.052599	0.008404
	(5.0, 0.8)	(1.0, 1.0)	0.194444	1.875464	-0.27449	0.213781	0.214634	-.0.00429	13.33743	0.219675	0.317593	0.308933	0.007289
(10, 25)	(2.5, 0.2)	(2.5, 5.5)	0.208257	0.110289	0.014919	0.291165	0.667438	-0.00138	0.572826	0.086349	0.770191	4.073478	0.007141
	(2.5, 0.5)	(5.5, 2.5)	0.582037	0.380745	-0.09166	0.850794	0.353988	-0.00112	1.432139	0.141221	4.777875	0.914937	0.012258
	(2.5, 0.8)	(0.5, 0.8)	0.245386	0.806538	-0.28398	0.058733	0.118996	0.001355	2.793671	0.230967	0.024612	0.095397	0.005962
	(1.0, 0.5)	(0.8, 0.5)	0.712323	0.170993	-0.11999	0.097783	0.045964	0.006831	0.218860	0.143991	0.085523	0.023969	0.006786
	(5.0, 0.8)	(1.0, 1.0)	0.194444	1.771473	-0.29811	0.121600	0.125991	-0.00027	11.56848	0.231946	0.124004	0.128786	0.005236
(25 ,50)	(2.5, 0.2)	(2.5, 5.5)	0.208257	0.036262	0.010927	0.164079	0.392155	-0.00187	0.269525	0.059173	0.286573	1.542792	0.003134
	(2.5, 0.5)	(5.5, 2.5)	0.582037	0.134098	-0.03468	0.392208	0.157711	0.002272	0.486505	0.099014	1.661568	0.294352	0.005031
	(2.5, 0.8)	(0.5, 0.8)	0.245386	0.383396	-0.14538	0.025354	0.047193	0.001778	0.898642	0.121741	0.009667	0.034853	0.002874
	(1.0, 0.5)	(0.8, 0.5)	0.712323	0.040669	-0.02206	0.050963	0.023805	0.004615	0.074731	0.099756	0.032506	0.009107	0.003135
	(5.0, 0.8)	(1.0, 1.0)	0.194444	0.946003	-0.17374	0.052254	0.055951	-0.00123	4.099781	0.135251	0.046451	0.050182	0.002305
(50, 25)	(2.5, 0.2)	(2.5, 5.5)	0.208257	0.030015	0.008291	0.293533	0.696221	-0.00279	0.129235	0.032879	0.785995	4.361865	0.002394
	(2.5, 0.5)	(5.5, 2.5)	0.582037	0.043115	0.000018	0.786336	0.302921	0.003045	0.276691	0.070008	4.586204	0.823109	0.003742
	(2.5, 0.8)	(0.5, 0.8)	0.245386	0.236301	-0.08441	0.052070	0.101196	-0.00202	0.512083	0.075606	0.029738	0.098921	0.003809
	(1.0, 0.5)	(0.8, 0.5)	0.712323	0.021708	-0.00454	0.108505	0.058826	0.002022	0.021708	0.065830	0.094711	0.029625	0.003837
	(5.0, 0.8)	(1.0, 1.0)	0.194444	0.498350	-0.10049	0.130694	0.124434	-0.00024	2.068763	0.077105	0.137592	0.135552	0.003234
(100, 50)	(2.5, 0.2)	(2.5, 5.5)	0.208257	0.018854	-0.00291	0.172070	0.406172	-0.00091	0.060532	0.015514	0.317159	1.791987	0.001300
	(2.5, 0.5)	(5.5, 2.5)	0.582037	-0.00827	0.010762	0.304721	0.128561	0.000303	0.148047	0.040304	1.620871	0.289207	0.002228
	(2.5, 0.8)	(0.5, 0.8)	0.245386	0.155834	-0.06036	0.019994	0.044486	-0.00106	0.306242	0.051111	0.009227	0.033478	0.001963
	(1.0, 0.5)	(0.8, 0.5)	0.712323	0.002511	0.010017	0.050127	0.019303	0.002910	0.024788	0.039470	0.034263	0.008648	0.001960
	(5.0, 0.8)	(1.0, 1.0)	0.194444	0.228568	-0.04682	0.057514	0.060978	-0.00126	1.168521	0.049613	0.045943	0.047467	0.001588
(50, 100)	(2.5, 0.2)	(2.5, 5.5)	0.208257	0.016003	0.011676	0.064661	0.158484	-0.00213	0.134899	0.036872	0.131177	0.666178	0.001580
	(2.5, 0.5)	(5.5, 2.5)	0.582037	0.046536	-0.00954	0.148746	0.068409	0.001471	0.283693	0.065737	0.699783	0.131536	0.002646
	(2.5, 0.8)	(0.5, 0.8)	0.245386	0.221706	-0.09270	0.010121	0.023254	0.001215	0.496333	0.078023	0.003991	0.014453	0.001341
	(1.0, 0.5)	(0.8, 0.5)	0.712323	0.014756	0.000723	0.023362	0.012129	0.00151	0.046061	0.072986	0.014081	0.004036	0.001646
	(5.0, 0.8)	(1.0, 1.0)	0.194444	0.512497	-0.10085	0.032809	0.031061	0.000029	2.058304	0.078547	0.020132	0.019137	0.001138

Table 1: (continued).

(m, n)	X	Y	R	Bias					MSE				
	(σ, π)	(α, β)		$\hat{\sigma}$	$\hat{\pi}$	$\hat{\alpha}$	$\hat{\beta}$	$\hat{R}$	$\hat{\sigma}$	$\hat{\pi}$	$\hat{\alpha}$	$\hat{\beta}$	$\hat{R}$
(150, 100)	(2.5, 0.2)	(2.5, 5.5)	0.208257	0.011642	-0.00131	0.078024	0.206914	-0.00147	0.045793	0.011593	0.127875	0.749157	0.000768
	(2.5, 0.5)	(5.5, 2.5)	0.582037	0.011145	-0.00362	0.104127	0.044323	0.001527	0.106136	0.029387	0.630966	0.118109	0.001214
	(2.5, 0.8)	(0.5, 0.8)	0.245386	0.089809	-0.03373	0.010079	0.020350	-0.00021	0.204085	0.039053	0.004534	0.014002	0.001062
	(1.0, 0.5)	(0.8, 0.5)	0.712323	-0.00529	0.012016	0.020092	0.010267	0.000359	0.016731	0.029364	0.014003	0.004025	0.001039
	(5.0, 0.8)	(1.0, 1.0)	0.194444	0.146276	-0.03051	0.027118	0.020922	0.001478	0.806369	0.037418	0.019924	0.018329	0.000781
(100, 150)	(2.5, 0.2)	(2.5, 5.5)	0.208257	0.015716	-0.00481	0.054454	0.125010	0.000037	0.067428	0.016969	0.081871	0.445676	0.000918
	(2.5, 0.5)	(5.5, 2.5)	0.582037	0.012379	0.013253	0.134374	0.053998	-0.00204	0.171257	0.045265	0.440803	0.080903	0.001457
	(2.5, 0.8)	(0.5, 0.8)	0.245386	0.109678	-0.04496	0.007721	0.015388	0.000666	0.278611	0.049376	0.002704	0.008407	0.000836
	(1.0, 0.5)	(0.8, 0.5)	0.712323	-0.00372	0.007027	0.016871	0.006216	0.002797	0.023587	0.039989	0.009148	0.002635	0.001009
	(5.0, 0.8)	(1.0, 1.0)	0.194444	0.344498	-0.06671	0.019912	0.023365	-0.00139	1.222075	0.052211	0.012488	0.012939	0.000695

Table 2: Simulation results using Monte-Carlo integration.

(m, n)	X	Y	R	Bias					MSE				
	(σ, π)	(α, β)		$\hat{\sigma}$	$\hat{\pi}$	$\hat{\alpha}$	$\hat{\beta}$	$\hat{R}$	$\hat{\sigma}$	$\hat{\pi}$	$\hat{\alpha}$	$\hat{\beta}$	$\hat{R}$
(5, 5)	(2.5, 0.2)	(2.5, 5.5)	0.207962	0.411113	-0.01122	3.496208	8.059887	-0.00581	1.801242	0.101341	113.4395	493.6784	0.018322
	(2.5, 0.5)	(5.5, 2.5)	0.578162	0.798675	-0.17618	6.986735	2.872253	-0.00918	3.881169	0.171614	302.1423	39.35341	0.030058
	(2.5, 0.8)	(0.5, 0.8)	0.247446	1.515862	-0.39105	0.677647	1.328643	-0.01887	8.523403	0.312065	17.71708	18.17913	0.022951
	(1.0, 0.5)	(0.8, 0.5)	0.726049	0.327646	-0.19344	1.442231	0.554272	0.011151	0.589825	0.175528	23.22687	1.887292	0.129911
	(5.0, 0.8)	(1.0, 1.0)	0.197259	3.471341	-0.40494	1.165309	1.211260	-0.01662	41.77231	0.322977	9.699013	11.67444	0.018389
(10, 10)	(2.5, 0.2)	(2.5, 5.5)	0.208266	0.118534	0.018526	1.012071	2.448597	-0.00362	0.682123	0.089797	4.509002	26.14572	0.009040
	(2.5, 0.5)	(5.5, 2.5)	0.579812	0.332884	-0.09237	2.478500	1.029729	0.004504	1.182121	0.146241	30.02546	5.140935	0.013510
	(2.5, 0.8)	(0.5, 0.8)	0.244926	0.833196	-0.26989	0.157774	0.316481	-0.00128	2.750783	0.220434	0.154252	0.698537	0.011247
	(1.0, 0.5)	(0.8, 0.5)	0.710521	0.144133	-0.10212	0.344295	0.158183	0.010037	0.219009	0.146319	0.756168	0.141195	0.011980
	(5.0, 0.8)	(1.0, 1.0)	0.196074	1.755397	-0.28698	0.375817	0.366797	-0.00592	11.73736	0.232276	1.110142	0.991879	0.008837
(25, 25)	(2.5, 0.2)	(2.5, 5.5)	0.208026	0.032083	0.014128	0.317131	0.705764	-0.00042	0.252334	0.057606	0.816279	4.262043	0.003389
	(2.5, 0.5)	(5.5, 2.5)	0.582988	0.140181	-0.02719	0.662386	0.293137	-0.00332	0.483497	0.096748	4.175351	0.815896	0.005681
	(2.5, 0.8)	(0.5, 0.8)	0.246592	0.466719	-0.17386	0.046887	0.082985	0.001782	1.026726	0.130976	0.023452	0.083856	0.004634
	(1.0, 0.5)	(0.8, 0.5)	0.705864	0.050419	-0.02773	0.085369	0.054716	0.006439	0.087801	0.100094	0.076503	0.024991	0.004994
	(5.0, 0.8)	(1.0, 1.0)	0.194425	0.843856	-0.14511	0.125006	0.118300	-0.00152	4.247545	0.128025	0.129261	0.122714	0.003676
(50, 50)	(2.5, 0.2)	(2.5, 5.5)	0.207872	0.017266	0.011751	0.123562	0.297792	-0.00101	0.137819	0.033124	0.273879	1.568441	0.001842
	(2.5, 0.5)	(5.5, 2.5)	0.582787	0.068393	-0.01312	0.350582	0.154906	-0.00076	0.287218	0.066974	1.651713	0.309901	0.002858
	(2.5, 0.8)	(0.5, 0.8)	0.247384	0.285931	-0.10474	0.024953	0.041608	-0.00162	0.533421	0.081499	0.010607	0.033385	0.002107
	(1.0, 0.5)	(0.8, 0.5)	0.712615	0.016603	-0.00709	0.033520	0.016431	0.001354	0.041523	0.067275	0.028709	0.009259	0.002456
	(5.0, 0.8)	(1.0, 1.0)	0.196536	0.468336	-0.09084	0.048247	0.049386	-0.00245	1.886976	0.076475	0.041670	0.046058	0.001797
(100, 100)	(2.5, 0.2)	(2.5, 5.5)	0.210171	0.020946	-0.00343	0.084235	0.175488	-0.00204	0.072158	0.016459	0.136695	0.717866	0.000987
	(2.5, 0.5)	(5.5, 2.5)	0.576843	-0.00431	0.010264	0.162972	0.071364	0.00555	0.162534	0.044406	0.702143	0.134302	0.001544
	(2.5, 0.8)	(0.5, 0.8)	0.243669	0.141415	-0.05415	0.011832	0.022447	0.001884	0.297633	0.048634	0.004146	0.014000	0.001062
	(1.0, 0.5)	(0.8,0.5)	0.717167	0.003423	0.00509	0.020667	0.012154	-0.00369	0.025637	0.042254	0.012705	0.004210	0.001453
	(5.0, 0.8)	(1.0, 1.0)	0.194191	0.210366	-0.03947	0.028158	0.028798	-0.00002	1.131689	0.047141	0.019831	0.020381	0.000988
(150, 150)	(2.5, 0.2)	(2.5, 5.5)	0.207887	0.003921	0.00080	0.047572	0.124571	0.000055	0.042473	0.010388	0.082593	0.456684	0.000684
	(2.5, 0.5)	(5.5, 2.5)	0.579911	0.018646	-0.00397	0.133948	0.062542	0.002194	0.101824	0.027883	0.470931	0.087823	0.001101
	(2.5, 0.8)	(0.5, 0.8)	0.245856	0.065538	-0.02679	0.010385	0.017628	0.000203	0.202291	0.037947	0.002889	0.008854	0.000745
	(1.0, 0.5)	(0.8, 0.5)	0.705828	-0.00165	0.01268	0.012020	0.004609	0.006617	0.017935	0.029181	0.008458	0.002711	0.001061
	(5.0, 0.8)	(1.0, 1.0)	0.191908	0.126655	-0.02091	0.019049	0.017583	0.002529	0.802992	0.035982	0.012561	0.013304	0.000641
(15, 10)	(2.5, 0.2)	(2.5, 5.5)	0.208513	0.074366	0.016139	0.947069	2.321799	-0.00520	0.414356	0.081027	5.246853	32.51808	0.006956
	(2.5, 0.5)	(5.5, 2.5)	0.576659	0.226699	-0.06804	2.246245	0.993720	0.004922	0.773751	0.125291	25.05196	4.740335	0.011103
	(2.5, 0.8)	(0.5, 0.8)	0.246500	0.661514	-0.22617	0.152755	0.328057	-0.00407	1.728951	0.176766	0.143962	0.760190	0.010976
	(1.0, 0.5)	(0.8, 0.5)	0.716926	0.103454	-0.08652	0.319881	0.166624	0.006264	0.135538	0.130904	0.577706	0.168956	0.013405
	(5.0, 0.8)	(1.0, 1.0)	0.193469	1.293562	-0.21471	0.425067	0.382571	-0.00244	6.637686	0.171450	0.854148	0.769636	0.008415

Table 2: (continued).

(m, n)	X	Y	R	Bias					MSE				
	(σ, π)	(α, β)		$\hat{\sigma}$	$\hat{\pi}$	$\hat{\alpha}$	$\hat{\beta}$	$\hat{R}$	$\hat{\sigma}$	$\hat{\pi}$	$\hat{\alpha}$	$\hat{\beta}$	$\hat{R}$
(25, 10)	(2.5, 0.2)	(2.5, 5.5)	0.208721	0.044927	0.017913	0.977934	2.264829	-0.00395	0.263558	0.057201	5.223055	26.33746	0.005524
	(2.5, 0.5)	(5.5, 2.5)	0.579132	0.094162	-0.01967	2.415365	0.983171	0.00808	0.513319	0.102049	31.68673	5.328729	0.008781
	(2.5, 0.8)	(0.5, 0.8)	0.243840	0.423827	-0.14631	0.161045	0.316012	-0.00282	1.020983	0.122976	0.158427	0.695086	0.009047
	(1.0, 0.5)	(0.8, 0.5)	0.714411	0.049482	-0.05269	0.319344	0.169374	0.005447	0.075527	0.097612	0.585297	0.211199	0.010451
	(5.0, 0.8)	(1.0, 1.0)	0.195632	0.745134	-0.14754	0.351543	0.332043	-0.00002	3.571187	0.122031	0.645975	0.680936	0.007660
(10, 15)	(2.5, 0.2)	(2.5, 5.5)	0.209026	0.098540	0.02403	0.599095	1.449910	-0.00351	0.640130	0.090175	2.076321	11.28231	0.007891
	(2.5, 0.5)	(5.5, 2.5)	0.581021	0.422013	-0.10356	1.357140	0.615461	-0.00648	1.370424	0.146988	11.32183	2.256161	0.013776
	(2.5, 0.8)	(0.5, 0.8)	0.244791	0.858122	-0.27644	0.097929	0.207086	-0.00282	2.723582	0.223538	0.059714	0.279424	0.008611
	(1.0, 0.5)	(0.8, 0.5)	0.726817	0.141828	-0.10101	0.180534	0.098865	-0.01099	0.189111	0.150878	0.233536	0.065544	0.009445
	(5.0, 0.8)	(1.0, 1.0)	0.193205	1.745311	-0.25634	0.206355	0.209985	-0.00374	11.75725	0.211221	0.266253	0.285319	0.007185
(10, 25)	(2.5, 0.2)	(2.5, 5.5)	0.209369	0.138324	0.025867	0.308762	0.706661	-0.00581	0.761456	0.098998	0.829999	4.335818	0.006847
	(2.5, 0.5)	(5.5, 2.5)	0.578365	0.382549	-0.12905	0.868204	0.353088	0.011908	1.248438	0.147738	5.267443	0.900094	0.012712
	(2.5, 0.8)	(0.5, 0.8)	0.245003	0.959686	-0.28368	0.054377	0.089302	0.001179	3.429666	0.221418	0.027110	0.083894	0.006174
	(1.0, 0.5)	(0.8, 0.5)	0.724079	0.143769	-0.08374	0.107296	0.048298	-0.00712	0.190414	0.147854	0.090711	0.022757	0.008504
	(5.0, 0.8)	(1.0, 1.0)	0.196283	1.829193	-0.28341	0.127624	0.124801	-0.00339	12.26152	0.225797	0.113005	0.125352	0.005186
(25, 50)	(2.5, 0.2)	(2.5, 5.5)	0.206547	0.009054	0.017679	0.146898	0.354093	0.002026	0.250972	0.063448	0.288560	1.547066	0.003122
	(2.5, 0.5)	(5.5, 2.5)	0.579286	0.116727	-0.04222	0.276813	0.129478	0.007993	0.528931	0.106486	1.541201	0.298341	0.005431
	(2.5, 0.8)	(0.5, 0.8)	0.248331	0.461417	-0.15499	0.028396	0.048669	-0.00321	1.070791	0.124584	0.010237	0.033295	0.002858
	(1.0, 0.5)	(0.8, 0.5)	0.709828	0.049035	-0.03421	0.046020	0.025401	0.006508	0.085451	0.105578	0.033458	0.009228	0.003792
	(5.0, 0.8)	(1.0, 1.0)	0.197451	0.892497	-0.15453	0.050431	0.052590	-0.00417	4.052606	0.121906	0.043125	0.044793	0.002481
(50, 25)	(2.5, 0.2)	(2.5, 5.5)	0.210632	0.022127	0.004876	0.307973	0.723748	-0.00408	0.124785	0.034556	0.786939	4.426586	0.002275
	(2.5, 0.5)	(5.5, 2.5)	0.582341	0.053089	-0.01419	0.636795	0.280714	0.001112	0.271673	0.069565	4.292317	0.814649	0.003966
	(2.5, 0.8)	(0.5, 0.8)	0.243743	0.245271	-0.09261	0.053418	0.094171	0.001641	0.486057	0.077033	0.023667	0.083142	0.003653
	(1.0, 0.5)	(0.8, 0.5)	0.711855	0.017721	0.000673	0.101253	0.054078	0.001124	0.047118	0.069225	0.085630	0.025674	0.004201
	(5.0, 0.8)	(1.0, 1.0)	0.196415	0.496731	-0.08528	0.124499	0.129421	-0.00730	2.175283	0.075058	0.123603	0.117603	0.003031
(100, 50)	(2.5, 0.2)	(2.5, 5.5)	0.210011	-0.00414	0.007766	0.139404	0.332330	-0.00141	0.069003	0.017041	0.300367	1.626574	0.001264
	(2.5, 0.5)	(5.5, 2.5)	0.581498	0.010097	-0.00221	0.356358	0.149078	0.00338	0.147047	0.039697	1.513093	0.304588	0.002007
	(2.5, 0.8)	(0.5, 0.8)	0.244432	0.107721	-0.04831	0.024206	0.045945	0.00226	0.281571	0.048596	0.010331	0.035795	0.001981
	(1.0, 0.5)	(0.8, 0.5)	0.702878	0.000643	0.002565	0.046881	0.024134	0.012877	0.023081	0.038733	0.031009	0.008854	0.003671
	(5.0, 0.8)	(1.0, 1.0)	0.194201	0.213932	-0.04165	0.057367	0.052393	0.00049	1.158258	0.051177	0.047722	0.045089	0.001696
(50, 100)	(2.5, 0.2)	(2.5, 5.5)	0.208596	0.030950	0.001439	0.077285	0.190037	-0.00301	0.127453	0.034441	0.135884	0.733471	0.001551
	(2.5, 0.5)	(5.5, 2.5)	0.583783	0.028561	0.002098	0.193045	0.075996	-0.00119	0.257674	0.069389	0.758088	0.128101	0.002887
	(2.5, 0.8)	(0.5, 0.8)	0.245583	0.253602	-0.09962	0.012137	0.029932	-0.00139	0.495781	0.078152	0.004237	0.014768	0.001415
	(1.0, 0.5)	(0.8, 0.5)	0.722373	0.014239	-0.00348	0.024091	0.012463	-0.00745	0.044673	0.065541	0.014248	0.004381	0.001954
	(5.0, 0.8)	(1.0, 1.0)	0.194702	0.449582	-0.08283	0.033963	0.032666	-0.00056	1.986678	0.069366	0.019292	0.019207	0.001241

Table 2: (continued).

(m, n)	X	Y	R	Bias					MSE				
	(σ, π)	(α, β)		$\hat{\sigma}$	$\hat{\pi}$	$\hat{\alpha}$	$\hat{\beta}$	$\hat{R}$	$\hat{\sigma}$	$\hat{\pi}$	$\hat{\alpha}$	$\hat{\beta}$	$\hat{R}$
(150, 100)	(2.5, 0.2)	(2.5, 5.5)	0.208265	0.019721	0.00142	0.077300	0.173331	-0.00183	0.044313	0.011747	0.125423	0.675796	0.000747
	(2.5, 0.5)	(5.5, 2.5)	0.585533	-0.01583	0.015218	0.189875	0.079889	-0.00412	0.098242	0.028199	0.707084	0.129897	0.001211
	(2.5, 0.8)	(0.5, 0.8)	0.243312	0.083308	-0.02746	0.009802	0.017609	0.002404	0.209196	0.036367	0.003822	0.013989	0.001095
	(1.0, 0.5)	(0.8, 0.5)	0.723141	-0.00494	0.016849	0.016329	0.009425	-0.01156	0.016883	0.027435	0.014113	0.004195	0.001379
	(5.0, 0.8)	(1.0, 1.0)	0.196121	0.187475	-0.03333	0.016826	0.022855	-0.00294	0.844036	0.038168	0.019470	0.020331	0.000840
(100, 150)	(2.5, 0.2)	(2.5, 5.5)	0.208448	0.002001	0.003858	0.049352	0.096774	0.000784	0.060386	0.017070	0.076548	0.417269	0.000839
	(2.5, 0.5)	(5.5, 2.5)	0.586331	0.006686	0.013848	0.113506	0.048239	-0.00634	0.159696	0.044092	0.467262	0.081357	0.001559
	(2.5, 0.8)	(0.5, 0.8)	0.243094	0.129898	-0.04774	0.006945	0.017088	0.001315	0.286447	0.050689	0.002696	0.009020	0.000786
	(1.0, 0.5)	(0.8, 0.5)	0.698669	0.000046	0.00472	0.015588	0.008292	0.016188	0.024726	0.043503	0.008465	0.002721	0.001831
	(5.0, 0.8)	(1.0, 1.0)	0.194647	0.256815	-0.05137	0.017881	0.018285	-0.00021	1.082450	0.048901	0.012319	0.012536	0.000678

9. Discussion

In this study, we try to validate the consistency and efficiency of the maximum likelihood estimates of four parameters and R, the stress-strength model in our case. We computed R through two methods numerical integration and Monte-Carlo integration methods. We observe that with increasing sample sizes, biases and MSE's decrease. Biases imply how much distant the average estimate value is from the true value of the parameter. MSE's imply how widely spread the estimates are from one data sample to another. Both the integration methods for R gives more or less the same precision for each sample size and set of parameter values. Application of such studies in real life data to deduce inferences on that could be a scope of further studies. Such studies are useful in drawing inference about the stress-strength model to hold how reliable the model is, which has many applications in different fields.

References

1. Asgharzadeh, A., Valiollahi, R., & Raqab, M. Z. (2011). Stress-strength reliability of Weibull distribution based on progressively censored samples. *SORT-Statistics and Operations Research Transactions*, 103-124.

2. Bilikam, J. E. (1985). Some stochastic stress-strength processes. *IEEE transactions on reliability*, *34*(3), 269-274.

3. Biswas, A., & Chakraborty, S. (2021). Stress–Strength Reliability for the Unit-Lindley Distribution with an Application. *Calcutta Statistical Association Bulletin*, *73*(1), 7-23.

4. Biswas, A., Chakraborty, S., & Mukherjee, M. (2021). On estimation of stress–strength reliability with log-Lindley distribution. *Journal of Statistical Computation and Simulation*, *91*(1), 128-150.

5. Gómez-Déniz, E., Sordo, M. A., & Calderín-Ojeda, E. (2014). The Log–Lindley distribution as an alternative to the beta regression model with applications in insurance. *Insurance: Mathematics and Economics*, *54*, 49-57.

6. Halperin, M., Hamdy, M. I., & Thall, P. F. (1989). Distribution-free confidence intervals for a parameter of Wilcoxon-Mann-Whitney type for ordered categories and progressive censoring. *Biometrics*, 509-521.

7. Jodrá, P., & Jiménez-Gamero, M. D. (2016). A note on the Log-Lindley distribution. *Insurance: Mathematics and Economics*, *71*, 189-194.

8. Khan, A. H., & Jan, T. R. (2015). Estimation of stress-strength reliability model using finite mixture of two parameter Lindley distributions. *Journal of Statistics Applications and Probability*, *4*(1), 147-159.

9. Kızılaslan, F., & Nadar, M. (2018). Estimation of reliability in a multicomponent stress–strength model based on a bivariate Kumaraswamy distribution. *Statistical Papers*, *59*(1), 307-340.

10. Kotz, S., & Pensky, M. (2003). *The stress-strength model and its generalizations: theory and applications*. World Scientific.

11. Nkemnole, E. B., & Samiyu, M. A. (2017). Inference on Stress-Strength Reliability for Log-Normal Distribution based on Lower Record Values. *AMSE JOURNALS-AMSE IIETA*, *22*(1), 77-97.

12. Wolfe, D. A., & Hogg, R. V. (1971). On constructing statistics and reporting data. *The American Statistician*, *25*(4), 27-30.

Appendix

Code1: Obtaining R by numerical integration method.

```
rel_LB=function(sigma,pi,alpha,beta)
{
 f=function(x)
 {
  (1+sigma*(pi-1)*log(x))*x^(sigma)*x^(alpha-1)*(1-x)^(beta-1)*(1/beta(alpha, beta))
 }
 R_num=integrate(f,lower=0,upper=1)
 return(R_num$value)
}
```

Code2: Obtaining R by Monte-Carlo integration method.

```
rel_LB=function(sigma,pi,alpha,beta)
{
 f=function(x)
 {
  (1+sigma*(pi-1)*log(x))*x^(sigma)*x^(alpha-1)*(1-x)^(beta-1)*(1/beta(alpha, beta))
 }
 size.mc=25000
 sample.unif=runif(size.mc,0,1)
 value=f(sample.unif)
 R_mc=mean(value)
 return(R_mc)
}
```

Code3: Random sample generation from log-Lindley distribution.

```
sample_L=function(m,sigma,pi)
{
 sampl=0
 for(i in 1:m)
 {
  u=runif(1)
  nleq=function(x)
  {
   (1+sigma*(pi-1)*log(x))*x^sigma-u
```

```
  }
    sampl[i]=uniroot(nleq,c(0.00000000000001,0.9999999999999),tol=0.00001)$root
  }
  return(sampl)
}
```

Code4: MLE for log-Lindley distribution.

```
mle.LLD=function(sample)
{
  ## Initialization: Starting values
  pi.f=0.5
  x.bar=mean(sample)
  sig.0= sqrt(x.bar)/(1-sqrt(x.bar))
  sig.1=x.bar/(1-x.bar)
  sigma.f=(sig.0+sig.1)/2
  start=c(pi.f,sigma.f)
  ## function for negative log likelihood
  y=sample
  neg.LL=function(theta)
  {
    -(length(y)*log(theta[2])+(theta[2]-1)*sum(log(y))+sum(log(theta[1]+theta[2]*(theta[1]-
1)*log(y))))
  }
  ##constrain matrix and vector for constrOptim
  constr.mat=matrix(c(1,0,0,1,-1,0),nrow=3,byrow=T)
  constr.vec=c(0,0,-1)
  ## neg.ll minimization using constrOptim function
  object=constrOptim(theta=start,f=neg.LL,NULL,ui=constr.mat,ci=constr.vec)
  pi.mle=object$par[1]
  sigma.mle=object$par[2]
  return(c(sigma.mle,pi.mle))
}
```

Code5: Random sample generation from Beta distribution.

```
sample_B=function(n,alpha,beta)
{
  return(rbeta(n,alpha,beta)) }
```

Code6: MLE of Beta distribution.

```
Library(univariateML)
mle_B=function(sample)
{
  mlbeta(sample)
}
```

Code7: Main simulation.

```
# m: Sample size of L; n: Sample size of B
details=function(m,sigma,pi,n,alpha,beta)
{
  N=1000 #Simulation size
  r=rel_LB(sigma,pi,alpha,beta)
  b.sigma=0
  m.sigma=0
  b.pi=0
  m.pi=0
  b.alpha=0
  m.alpha=0
  b.beta=0
  m.beta=0
  b.r=0
  m.r=0
  for (i in 1:N)
  {
    x=sample_L(m,sigma,pi)
    y=sample_B(n,alpha,beta)
    object.1=mle_L(x)
    est.sigma=object.1[1]
    est.pi=object.1[2]
    object.2=mle_B(y)
    est.alpha=object.2[1]
    est.beta=object.2[2]
    est.r=rel_LB(est.sigma,est.pi,est.alpha,est.beta)
    b.sigma[i]=est.sigma-sigma
    m.sigma[i]=(est.sigma-sigma)^2
    b.pi[i]=est.pi-pi
    m.pi[i]=(est.pi-pi)^2
```

```
    b.alpha[i]=est.alpha-alpha
    m.alpha[i]=(est.alpha-alpha)^2
    b.beta[i]=est.beta-beta
    m.beta[i]=(est.beta-beta)^2
    b.r[i]=est.r-r
    m.r[i]=(est.r-r)^2
  }
  head=c("sigma","pi","alpha","beta","Rel")
  bias=c(mean(b.sigma),mean(b.pi),mean(b.alpha),mean(b.beta),mean(b.r))
  mse=c(mean(m.sigma),mean(m.pi),mean(m.alpha),mean(m.beta),mean(m.r))
  d=data.frame(head,bias,mse)
  return(list("r"=r,"d"=d))
}
# Taking outputs
out=details(100,2.5,0.4,100,1.2,3.5)
out
```