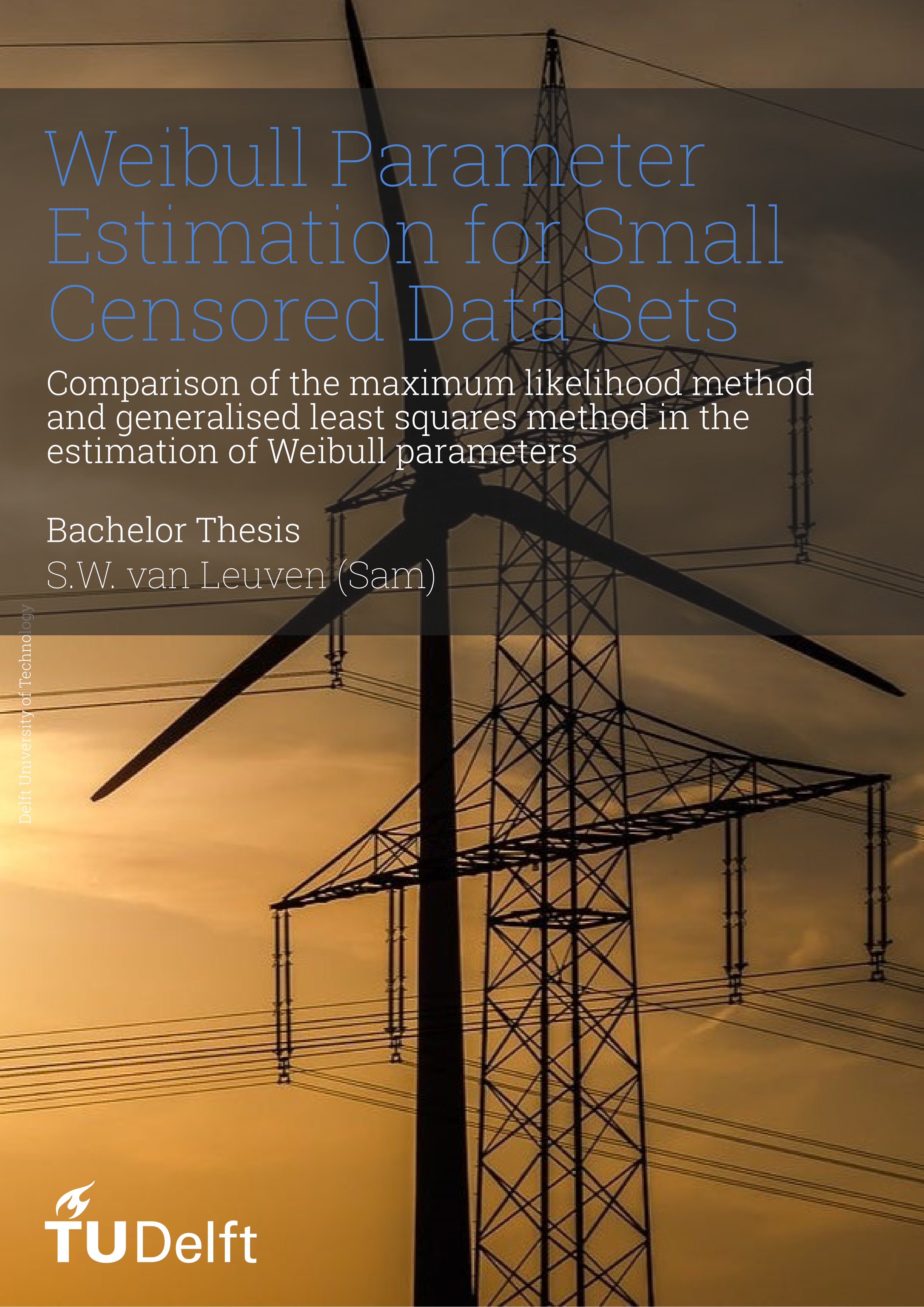


# Weibull Parameter Estimation for Small Censored Data Sets

## Comparison of the maximum likelihood method and generalised least squares method in the estimation of Weibull parameters

Bachelor Thesis

S.W. van Leuven (Sam)

Delft University of Technology

# Weibull Parameter Estimation for Small Censored Data Sets

## Comparison of the maximum likelihood method and generalised least squares method in the estimation of Weibull parameters

by

S.W. van Leuven (Sam)

to obtain the degree of Bachelor of Science
at the Delft University of Technology,
to be defended publicly on Wednesday July 24, 2024.

| | |
|---|---|
| Student number: | 5167310 |
| Thesis committee: | L. Meester, TU Delft (supervisor) |
| | R. Lophuaä, TU Delft |
| | R. Ross, TU Delft |
| Project Duration: | April, 2024 - July, 2024 |
| Faculty: | Faculty of Electrical Engineering, Mathematics and Computer Sciences, Delft |


| | |
|---|---|
| Cover: | K.Reichert, from: https://medium.com/@LineVision/our-20th-century-grid-wont-support-a-21st-century-clean-energy-economy-22bbf09df3ed |
| Style: | TU Delft Report Style, with modifications by Daan Zwaneveld |

Voor Eja

# Abstract

The Weibull distribution is one of the most widely used distributions in reliability analysis. The ability to accurately estimate the parameters of Weibull distributed data can be very useful, and particularly important when dealing with small data sets and high degrees of censoring. This project aims to compare the performance of the maximum likelihood (ML) method and a generalised least squares (GLS) method in estimating the parameters of small, censored Weibull distributed data. The study involves a simulation of type II censored Weibull data to estimate the log-Weibull parameters and predict the quantiles of the next failure. Simulations were performed across a broad range of sample sizes and number of observed failures. We evaluated the efficiency and accuracy of both estimation methods.

The simulation results demonstrated that the unbiased versions of the ML estimators consistently outperform the GLS estimators in terms of efficiency. The root mean squared error (RMSE) of the ML estimator were lower, indicating a higher accuracy in parameter estimation. Additionally, the efficiency of the ML estimation method in predicting the future failure quantiles was higher across all studied cases compared to the GLS estimation method. These findings suggest that the ML estimation method is more suitable for small, censored Weibull datasets, offering significant advantages in accuracy and efficiency.

# Lay summary

The Weibull distribution is often used in engineering to predict the lifespan of different products, such as high-voltage electricity cables and medical devices. Estimating the parameters of an underlying distribution of failure data can be very important, especially when dealing with small or incomplete (censored) data sets. Our research compares two methods to estimate those parameters: the maximum likelihood (ML) method and the generalised least squares (GLS) method. Through simulations we found that the ML method consistently provides more accurate and reliable estimates than the GLS, also in predicting when next failures will occur. These results highlight the importance of choosing the right estimation method to ensure better product durability and safety. Understanding these methods can significantly impact fields like engineering and medicine, where reliable lifespan predictions are crucial.

# Contents

# List of Figures

# 1 Introduction

The Weibull distribution is probably the most widely used distributions in reliability analysis. This distribution, named after Wallodi Weibull [15], is used in a variety of real-world applications. For instance, to describe the time until breakdown of high-voltage cables and insulators [12], and to model safe dosage levels of medicine in medical research [3]. A key aspect of Weibull analysis is the estimation of the parameters of Weibull distributed data. In this report two popular parameter estimation methods are evaluated: the maximum likelihood (ML) method and the linear regression (LR) method.

To obtain accurate estimates of parameters one needs to acquire ample data. However, in reality, this is not always feasible. Experiments may be very expensive, or the occurrence of failure events can be highly undesirable, thus restricting the amount of data. Another limiting factor is that exact breakdown of failure times may not be known. Such data is said to be censored. There are many reasons for data to be censored. This could, for instance, be due to experimental restrictions, similar to the ones named before, or due to other irrelevant breakdowns during an experiment.

When observing equipment lifetimes one may wish to analyse the data well before all components have failed, a common occurrence for highly reliable equipment. Unexpected early failures may indicate that a whole batch is inferior and that possibly drastic measures may have to be taken. In this situation one may have only observed a small number of failures and for the remainder of the components one only knows that their lifetime exceeds a certain value. This type of censoring is called type II censoring, the type considered in this report, but there are many other censoring schemes.

The ability to accurately estimate the parameters of Weibull distributed data can be very useful, and particularly important when dealing with small data sets and high degrees of censoring. As the choice of estimation method can impact the accuracy of the estimation, this projects aims to compare the performance of the maximum likelihood method and the linear regression by generalised least squares method in estimating the parameters of small, censored Weibull distributed data.

The report begins with a theoretical overview of the Weibull distribution in Chapter 2. In the following two chapters the maximum likelihood method (Chapter 3) and the linear regression method (Chapter 4) for estimating Weibull parameters are discussed. The theory on linear regression will consists of regression by ordinary, weighted, and generalised least squares. Chapter 5 then explains the simulation methods used in this study. In Chapter 6, the performance of different estimators is compared, and the results are discussed. Finally, Chapter 7 concludes the report, summarising the findings and their implications.

# 2

# Weibull distribution and order statistics

In this report we will use the 2-parameter Weibull distribution (hereafter: Weibull distribution), with probability density function

$$f(t;\alpha,\beta) = \begin{cases} \frac{\beta}{\alpha}\left(\frac{t}{\alpha}\right)^{\beta-1} e^{-(t/\alpha)^\beta} & , t \geq 0 \\ 0 & , t < 0, \end{cases} \tag{2.1}$$

and cumulative distribution function

$$F(t;\alpha,\beta) = \begin{cases} 1 - e^{-(t/\alpha)^\beta} & , t \geq 0 \\ 0 & , t < 0. \end{cases} \tag{2.2}$$

The two parameters that characterise this distribution are the scale parameter $\alpha > 0$ and the shape parameter $\beta > 0$. For $\beta$ near $3.6$ this distribution is very similar to that of the Gaussian distribution. Also $\beta = 1$ is a special case, where the Weibull distribution coincides with the exponential distribution.

For the estimation of Weibull-parameters we need to introduce the notion of order statistics. Consider a random sample $T_1, T_2, \ldots, T_n$ of the Weibull distribution. If these random variables are arranged in order of magnitude such that

$$T_{1:n} \leq T_{2:n} \leq \ldots \leq T_{n:n},$$

then $T_{i:n}$ is called the i-th *order statistic* of a sample of size $n$ [2]. Similarly arranging the actual realisations $t_1, t_2, \ldots, t_n$ we write

$$t_{1:n} \leq t_{2:n} \leq \ldots \leq t_{n:n}.$$

If we are working with type II censored data only the first $r < n$ failure times are known. The order statistics of such a censored sample are $T_{1:n} \leq T_{2:n} \leq \ldots \leq T_{r:n}$.

# 3

# Maximum likelihood estimator

In this chapter the maximum likelihood estimators for the Weibull scale and shape parameters are constructed. First the estimators for the case where the data set contains only uncensored data (Section 3.1) are determined. After which, the case of partially censored data is elaborated upon (Section 3.2). The information in this chapter is based on the work of Rob Ross [13].

## 3.1. ML-estimation for uncensored data

Consider a random sample of $n$ random variables $T_1, T_2, \ldots, T_n$ that are independent identically distributed with the probability density function of the Weibull distribution $f(\,\cdot\,; \alpha, \beta)$, for $\alpha, \beta > 0$. Now let $t_1, t_2, \ldots, t_n$ be the realisations of the i.i.d. random variables. To obtain the maximum likelihood estimates, denoted $a$ and $b$ for $\alpha$ and $\beta$ respectively, the likelihood function is minimised. The likelihood function is defined as

$$L(\alpha, \beta) = \prod_{i=1}^{n} f(t_i; \alpha, \beta) \tag{3.1}$$

By taking the logarithm of the likelihood function, the log-likelihood function is computed, which is used for convenience as this operation turns the product into a sum

$$\begin{aligned}
\log L(\alpha, \beta) &= \sum_{i=1}^{n} \log f(t_i; \alpha, \beta) \\
&= \sum_{i=1}^{n} \left[ \log\left(\frac{\beta}{\alpha}\right) + (\beta - 1)\log\left(\frac{t_i}{\alpha}\right) + \left(-\left(\frac{t_i}{\alpha}\right)^{\beta}\right)\right] \\
&= n\log\beta - n\beta\log\alpha + (\beta - 1)\sum_{i=1}^{n}\log\left(\frac{t_i}{\alpha}\right) - \sum_{i=1}^{n}\left(\frac{t_i}{\alpha}\right)^{\beta}.
\end{aligned} \tag{3.2}$$

Since the logarithm is a monotone increasing function, the log-likelihood function has the same maxima and minima as the likelihood function. Therefore, the maximum of the likelihood function can be determined by maximising $\log L(\alpha, \beta)$. To compute the maximum of the log-likelihood function, the roots of the partial derivatives are determined

$$\frac{\partial \log L(a, b)}{\partial a} = -\frac{nb}{a} + \frac{b}{a^{b+1}}\sum_{i=1}^{n} t_i^{b} = 0, \tag{3.3}$$

$$\frac{\partial \log L(a, b)}{\partial b} = \frac{n}{b} - n\log a + \sum_{i=1}^{n}\log t_i - \sum_{i=1}^{n}\left(\frac{t_i}{a}\right)^{b}\log\left(\frac{t_i}{a}\right) = 0. \tag{3.4}$$

Rewriting and combining these equations yields the following expressions for the maximum likelihood estimates

$$a^b = \frac{1}{n}\sum_{i=1}^{n} t_i^b, \tag{3.5}$$

$$\frac{1}{b} = \frac{\sum_{i=1}^{n} t_i^b \log t_i}{\sum_{i=1}^{n} t_i^b} - \frac{1}{n}\sum_{i=1}^{n} \log t_i. \tag{3.6}$$

These estimators are based on the actual observations. Since this research aims to study the performance of the estimator, we will not use the estimates for a specific set of observations, but the maximum likelihood estimators for the random sample $T_1, T_2, \ldots, T_n$. These estimators, denoted $\hat{\alpha}$ for $\alpha$ and $\hat{\beta}$ for $\beta$, follow directly from the expressions above

$$\hat{\alpha}^{\hat{\beta}} = \frac{1}{n}\sum_{i=1}^{n} T_i^{\hat{\beta}}, \tag{3.7}$$

$$\frac{1}{\hat{\beta}} = \frac{\sum_{i=1}^{n} T_i^{\hat{\beta}} \log T_i}{\sum_{i=1}^{n} T_i^{\hat{\beta}}} - \frac{1}{n}\sum_{i=1}^{n} \log T_i. \tag{3.8}$$

It should be noted that the actual computation of the maximum likelihood estimators, given a set of observations, requires the use of a numerical method such as the Newton-Raphson method, because the implicit expression for the shape parameter estimator $\hat{\beta}$ Equation (3.8) cannot be solved analytically. By substituting the result for $\hat{\beta}$ in Equation (3.7), the maximum likelihood estimator $\hat{\alpha}$ can be obtained.

## 3.2. ML-estimation for censored data

Below we derive the maximum likelihood estimators for the type II censored case. Again consider a random sample of $n$ Weibull distributed random variables. From this we obtain a censored sample $T_{1:n} \leq T_{2:n} \leq \ldots \leq T_{r:n}$. Now let $t_{1:n}, t_{2:n}, \ldots, t_{r:n}$ be the uncensored realisations. For the censored part, the exact time of failure is unknown. But because of the type of censoring a lower bound for the failure time is known, namely $t_{r:n}$. To adapt the likelihood function $L$ for the censored data, we use the cumulative distribution function $F$ where the likelihood for a censored component of the experiment is given by

$$\mathbb{P}(T_{i:n} \geq t_{r:n}) = 1 - F(t_{r:n}; \alpha, \beta).$$

This yields for the log-likelihood function

$$\begin{aligned} \log L(\alpha,\beta) &= \sum_{j=1}^{n-r} \log(1 - F(t_{r:n};\alpha,\beta)) + \sum_{i=1}^{r} \log f(t_{i:n};\alpha,\beta) \\ &= \sum_{j=1}^{n-r} \log(e^{-(t_{r:n}/\alpha)^\beta}) + r\log\beta - r\log\alpha + (\beta-1)\sum_{i=1}^{r}\log\left(\frac{t_{i:n}}{\alpha}\right) - \sum_{i=1}^{r}\left(\frac{t_{i:n}}{\alpha}\right)^\beta \\ &= -(n-r)\left(\frac{t_{r:n}}{\alpha}\right)^\beta + r\log\beta - r\log\alpha + (\beta-1)\sum_{i=1}^{r}\log\left(\frac{t_{i:n}}{\alpha}\right) - \sum_{i=1}^{r}\left(\frac{t_{i:n}}{\alpha}\right)^\beta. \end{aligned} \tag{3.9}$$

Similar to the uncensored case, the maximum of the log-likelihood function is determined by finding the roots of the partial derivatives, which is attained by solving

$$\frac{\partial \log L(a,b)}{\partial a} = -\frac{rb}{a} + \frac{b}{a^{b+1}}\left(\sum_{i=1}^{r} t_{i:n}^b + (n-r)t_{r:n}^b\right), \tag{3.10}$$

$$\frac{\partial \log L(a,b)}{\partial b} = \frac{r}{b} - r\log a + \sum_{i=1}^{r}\log t_{i:n} - \sum_{i=1}^{r}\left(\frac{t_{i:n}}{a}\right)^b \log\left(\frac{t_{i:n}}{a}\right) - (n-r)\left(\frac{t_{r:n}}{a}\right)^b \log\left(\frac{t_{r:n}}{a}\right) = 0. \tag{3.11}$$

By rewriting and combining these two equations, the (implicit) expressions for the two maximum likelihood estimates given a set of censored observations are found

$$a^b = \frac{1}{r}\left(\sum_{i=1}^{r} t_{i:n}^b + (n-r)t_{r:n}^b\right), \tag{3.12}$$

$$\frac{1}{b} = \frac{\sum_{i=1}^{r} t_{i:n}^b \log t_{i:n} + (n-r)t_{r:n}^b \log t_{r:n}}{\sum_{i=1}^{r} t_{i:n}^b + (n-r)t_{r:n}^b} - \frac{1}{r}\sum_{i=1}^{r} \log t_{i:n}. \tag{3.13}$$

The ML-estimators, denoted $\hat{\alpha}$ for $\alpha$ and $\hat{\beta}$ for $\beta$, follow directly from the expressions above, which are computed using the uncensored random variables $T_{1:n}, T_{2:n}, \ldots, T_{r:n}$, as follows

$$\hat{\alpha}^{\hat{\beta}} = \frac{1}{r}\left(\sum_{i=1}^{r} T_{i:n}^{\hat{\beta}} + \sum_{j=1}^{n-r} T_{r:n}^{\hat{\beta}}\right), \tag{3.14}$$

$$\frac{1}{\hat{\beta}} = \frac{\sum_{i=1}^{r} T_{i:n}^{\hat{\beta}} \log T_{i:n} + (n-r)T_{r:n}^{\hat{\beta}} \log T_{r:n}}{\sum_{i=1}^{r} T_{i:n}^{\hat{\beta}} + (n-r)T_{r:n}^{\hat{\beta}}} - \frac{1}{r}\sum_{i=1}^{r} \log T_{i:n}. \tag{3.15}$$

Just like in the uncensored case, the latter of these two equations (eq. 3.15) has to be solved numerically.

# 4

# Linear regression

Another method to estimate the parameters of the Weibull distribution is the method of linear regression. In this chapter three different methods of linear regression to estimate Weibull parameters will be discussed: regression by ordinary least squares (4.3), by weighted least squares (4.4.2), and by generalised least squares (4.4.1). Linear regression assumes a linear relation, therefore we will construct a location-scale family from the Weibull distribution function (4.1) of which a linear model will follow (4.2). The information in this chapter is partly based on the work of Balakrishnan and Cohen [2].

## 4.1. Location-scale family

First we will show that a location-scale family can be constructed from any distribution function. Let $Z$ be a random variable with cumulative distribution function $F$ and define $X = \mu + \sigma Z$, with $\mu \in \mathbb{R}$ and $\sigma > 0$. Then the cumulative distribution function of $X$ is given by

$$F(x;\mu,\sigma) \doteq \mathbb{P}(X \leq x) = \mathbb{P}(\mu + \sigma Z \leq x) = \mathbb{P}\left(Z \leq \frac{x-\mu}{\sigma}\right) = F\left(\frac{x-\mu}{\sigma}\right)$$

This defines a location-scale family that we can use with Weibull distributed data. Let $T_1, T_2, \ldots, T_n$ be independent and identically distributed Weibull random variables with scale and shape parameter $\alpha$ and $\beta$, respectively. Next define $X_i = \log T_i$ for all $i \in \{1, 2, \ldots, n\}$. The distribution function of $X_i$ for all $i \in \{1, 2, \ldots, n\}$ is

$$\begin{aligned}
\mathbb{P}(X_i \leq x) &= \mathbb{P}(\log T_i \leq x) \\
&= \mathbb{P}(T_i \leq e^x) \\
&= 1 - \exp\left(-\left(\frac{e^x}{\alpha}\right)^\beta\right) \\
&= 1 - \exp\left(-\exp(x - \log\alpha)\beta)\right) \\
&= 1 - \exp\left(-\exp\left(\frac{x - \log\alpha}{1/\beta}\right)\right) \\
&= G\left(\frac{x - \log\alpha}{1/\beta}\right),
\end{aligned}$$

Here $G(x) = 1 - \exp(-e^{-x})$, which is the (reversed) standard Gumbel distribution. It follows $X_1, \ldots, X_n$ are from a location-scale family with distribution function $G(\,\cdot\,; \log\alpha, 1/\beta)$. By defining $\mu \doteq \log\alpha$ and $\sigma = 1/\beta$ we can write $X_i = \mu + \sigma Z_i$ with $Z_1, Z_2, \ldots, Z_n$ independent and identically distributed with distribution function $G$.

### 4.1.1. Pivots

An important result for a location-scale family with parameters $\mu$ and $\sigma$ is that the pair $\left(\frac{\hat{\mu}-\mu}{\sigma}, \frac{\hat{\sigma}}{\sigma}\right)$ is a pivot of the estimator pair $(\hat{\mu}, \hat{\sigma})$ [8]. In particular this holds for both the ML estimators and the GLS estimators of the log-Weibull parameters [11]. The existence of these pivots imply that the distribution of the estimator pair does not depend on the unknown parameters. An important implication of the pivot pair is that we can study the distribution of the estimators for any arbitrary parameter pair by looking at the behaviour of only one case.

The pivotal functions also form the theoretical basis for bias correction of biased estimators. The maximum likelihood estimators are biased for the Weibull parameters. That is the expected value of the estimator is not equal to the true value of the estimated parameter. In the literature several unbiasing factors that use pivots have been proposed to reduce the bias of the ML estimator [12, 6, 14].

## 4.2. Linear model

Now suppose we have a censored sample of random variables with a Weibull distribution that consists of the first $r$ order statistics $T_{1:n}, T_{2:n}, \dots, T_{r:n}$. Let $X_{i:n} = \log T_{i:n}$ and denote $I = \{1, 2, \dots, r\}$. Then for all $i \in I$ we can write

$$\begin{aligned} X_{i:n} &= \mu + \sigma Z_{i:n} \\ &= \mu + \sigma \mathbb{E}\left[Z_{i:n}\right] + \sigma(Z_{i:n} - \mathbb{E}\left[Z_{i:n}\right]) \end{aligned}$$

If we denote $\epsilon_i = \sigma(Z_{i:n} - \mathbb{E}\left[Z_{i:n}\right])$ and define

$$\alpha_{i:n} = \mathbb{E}\left[Z_{i:n}\right], \quad \beta_{i,j:n} = \mathrm{Cov}(Z_{i:n}, Z_{j:n}), \quad \text{for} \quad i, j \in I,$$

as well as $\boldsymbol{\alpha} = (\alpha_{i:n})_{i \in I}$ and $\mathbf{B} = (\beta_{i,j:n})_{i,j \in I}$. Then the following linear model can be constructed

$$\mathbf{X} = \mu \mathbf{1} + \sigma \boldsymbol{\alpha} + \boldsymbol{\epsilon}, \tag{4.1}$$

In the above model $Z_{i:n}$ are random variables from a reversed standard Gumbel distribution. Hence the expected values for each $i \in I$ are known and given by

$$\mathbb{E}[Z_{i:n}] = -\gamma + i\binom{n}{i} \sum_{j=0}^{i-1} \binom{i-1}{j} (-1)^{i-1-j} \left( \frac{\log(n-j)}{n-j} \right), \tag{4.2}$$

where $\gamma$ is Euler's constant [16, 12].[1] Similarly the second moment can be calculated as follows

$$\mathbb{E}[Z_{i:n}^2] = i\binom{n}{i} \sum_{l=0}^{2} \left[ \binom{2}{l} (-1)^{2-m} \frac{\partial^l}{\partial s^l} \Gamma[s+1] \bigg|_{s=0} \sum_{j=0}^{i-1} \binom{i-1}{j} (-1)^{i-1-j} \left( \frac{(\log[n-j])^{k-l}}{n-j} \right) \right], \tag{4.3}$$

where[2]

$$\begin{aligned} \Gamma^{(0)}[1] &= 1 \\ \Gamma^{(1)}[1] &= -\gamma \\ \Gamma^{(2)}[1] &= \gamma^2 + \frac{\pi^2}{6}. \end{aligned}$$

The entries of the covariance matrix $\mathbf{B}$ can be calculated as follows

$$\beta_{i,j:n} = \mathrm{Cov}(Z_{i:n}, Z_{j:n}) = \mathbb{E}\left[Z_{i:n} Z_{j:n}\right] - \mathbb{E}\left[Z_{i:n}\right] \mathbb{E}\left[Z_{j:n}\right]. \tag{4.4}$$

A way to compute the first term was developed by Lieblein [9]. He found that

$$\begin{aligned} \mathbb{E}\left[Z_{i:n} Z_{j:n}\right] = & \frac{n!}{(i-1)!(j-i-1)!(n-j)!} \\ & \int_{\infty}^{\infty} \int_{\infty}^{y} x y e^{-y-e^{-y}} e^{-x-ie^{-x}} (e^{e^{-y}} - e^{e^{-x}})^{j-i-1} (1 - e^{e^{-y}})^{n-j} dx dy. \end{aligned} \tag{4.5}$$

For the evaluation of this double integral the reader is referred to Liebleins paper.

[1] See Appendix A for the complete calculation of this formula.
[2] See Appendix A for the complete calculation of this formula.

## 4.3. Ordinary least squares

The model in Equation (4.1) constitutes a regression model with design matrix $D := \begin{pmatrix}\mathbf{1} & \boldsymbol{\alpha}\end{pmatrix}$, parameter vector $\boldsymbol{\theta} = \begin{pmatrix}\mu & \sigma\end{pmatrix}'$ and error vector $\boldsymbol{\epsilon}$. For ordinary least squares it is assumed the errors $\epsilon_1, \epsilon_2, \dots, \epsilon_r$ are uncorrelated with expectation zero and variance $\sigma^2$. Then the OLS estimator for $\boldsymbol{\theta}$ is the vector that minimises the residual sum of squares $\|\mathbf{X} - D\boldsymbol{\theta}\|^2$. This is given by

$$\hat{\boldsymbol{\theta}}_{OLS} = (D'D)^{-1}D'\mathbf{X}. \tag{4.6}$$

The estimated scale and shape parameter of the Weibull distribution of $T_i$ can then be computed by using the estimated values of the parameters, such that

$$\hat{\alpha} = e^{\hat{\mu}}, \quad \hat{\beta} = 1/\hat{\sigma}. \tag{4.7}$$

From the Gauss-Markov theorem it follows that the OLS estimator is a linear unbiased estimator and has among all estimators the smallest variance, under the assumption that $D$ is non-random and has rank $2$ and the errors $\epsilon_1, \epsilon_2, \dots, \epsilon_r$ are independent and identically distributed with expectation zero and equal variance. However, since these assumptions do not hold for the Weibull distributed data that is studied in this thesis. The estimator in Equation (4.6) is unbiased, but is not the estimator with the smallest variance.

## 4.4. Generalised and weighted least squares

### 4.4.1. Generalised Least Squares

Since the sample for the model of Equation (4.1) are a set of order statistics the covariance matrix of the error vector is not of the form $\sigma^2\mathbf{I}$, but of the form $\sigma^2\mathbf{B}$ [4, 7]. Indeed, for all $i, j \in I$ it holds that

$$\mathrm{Cov}(\epsilon_i, \epsilon_j) = \sigma^2\mathrm{Cov}(Z_{i:n}, Z_{j,n}) = \sigma^2\beta_{i,j:n} \tag{4.8}$$

Hence the assumptions on the error vector are reduced to $\mathbb{E}[\boldsymbol{\epsilon}] = 0$. Let $W$ be a symmetric positive definite matrix and define

$$\tilde{X} = W^{1/2}X, \quad \tilde{D} = W^{1/2}D, \quad \text{and} \quad \tilde{\boldsymbol{\epsilon}} = W^{1/2}\tilde{\boldsymbol{\epsilon}}.$$

Multiplying the linear model (4.1) by $W^{1/2}$ creates the model $\tilde{X} = \tilde{D}\boldsymbol{\theta} + \tilde{\epsilon}$ for which $\mathrm{Var}(\tilde{\boldsymbol{\epsilon}}) = \sigma^2 W^{1/2}\mathbf{B}W^{1/2}$. This satisfies the Gauss-Markov theorem if there is no (deterministic) linear dependence among the coordinates of $\boldsymbol{\epsilon}$. From the Gauss-Markov theorem it follows the generalised least squares estimator is the best linear unbiased estimator (BLUE). The estimator for the parameter vector of $\mu$ and $\sigma$ then is

$$\begin{pmatrix}\hat{\mu}_{GLS} \\ \hat{\sigma}_{GLS}\end{pmatrix} = (D'WD)^{-1}D'W\mathbf{X} = \left[\begin{pmatrix}\mathbf{1} & \boldsymbol{\alpha}\end{pmatrix}'\mathbf{B}^{-1}\begin{pmatrix}\mathbf{1} & \boldsymbol{\alpha}\end{pmatrix}\right]^{-1}\begin{pmatrix}\mathbf{1} & \boldsymbol{\alpha}\end{pmatrix}'\mathbf{B}^{-1}\mathbf{X}. \tag{4.9}$$

Which after working out yields the following generalised least squares estimates

$$\hat{\mu}_{GLS} = -\boldsymbol{\alpha}'\boldsymbol{\Delta}\mathbf{X}, \quad \hat{\sigma}_{GLS} = \mathbf{1}'\boldsymbol{\Delta}\mathbf{X}, \quad \text{where} \quad \boldsymbol{\Delta} = \frac{\mathbf{B}^{-1}(\mathbf{1}\boldsymbol{\alpha}' - \boldsymbol{\alpha}\mathbf{1}')\mathbf{B}^{-1}}{(\boldsymbol{\alpha}'\mathbf{B}^{-1}\boldsymbol{\alpha})(\mathbf{1}'\mathbf{B}^{-1}\mathbf{1}) - (\boldsymbol{\alpha}'\mathbf{B}^{-1}\mathbf{1})^2}. \tag{4.10}$$

### 4.4.2. Weighted least squares

In some cases the elements of the error vector are uncorrelated but have unequal variance, such that $\mathbf{B}$ is a diagonal matrix. Consequently its inverse $W$ is diagonal as well. Then the minimisation of the residual sum of squares of (4.1) is equal to minimising a weighted sum of squares, where the $i$-th residual $X_i - (D\boldsymbol{\theta})_i$ has weight $w_i = W_{ii}$. Even though the above does not hold for the log-Weibull case, a weighted least squares estimate can still be computed by ignoring the covariances and only using the diagonal values of the covariance matrix. Then the weights are $W_{ii} = 1/\mathrm{Var}(\epsilon_i)$ for $i \in I$. We compute these weights by

$$\mathrm{Var}(\epsilon_i) = \mathrm{Var}(Z_{i:n}) = \mathbb{E}\left[Z_{i:n}^2\right] - (\mathbb{E}[Z_{i:n}])^2.$$

# 5

# Simulation

In order to compare the different ML and LS estimation methods a simulation is conducted. This chapter contains a description of the simulation experiment, the data generation and the parameter estimation. The next chapter (6) discusses the results of the simulation.

## 5.1. Simulation experiment

Our simulation is based on a type II censored experiment that starts of with a random Weibull sample with shape parameter $\beta = 1$ and scale parameter $\alpha = 1$. The sample consists of $n$ random variables at starting time $t = 0$. For the first $r$ elements that fail the failure time is registered, which provides for the order statistics

$$t_{1:n} \leq t_{2:n} \leq \ldots \leq t_{r:n}.$$

Due to the pivotal properties (See Section (4.1.1)) of both estimation methods, the choice of Weibull parameters can be done without loss of generality.

This research focuses on small and highly censored data. Therefore we conduct simulations for the following sample sizes and degrees of censoring:

- $n \in \{50, 75, 100, 125\}$; and
- $r \in \{4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$.

For each combination of these values, we carried out simulations and computed both the ML estimates and LR estimates for 250,000 samples. The outcomes were stored in files for later analysis and summary.

## 5.2. Data generation and estimation

The Weibull data are generated using the transformation $t_i = \alpha(-\log(1 - u_i))^{(1/\beta)}$. Where $u_i$ is a uniformly distributed random variate on the interval $(0, 1)$, which are generated using the uniform random number generator *Mersenne-Twister* [10]. The $r$ elements of the sample with the lowest failure time are then used for in the estimation. Estimates are computed using the maximum likelihood method as described in chapter 3 and using the generalised least squares method as described in chapter 4. For the computation of the GLS estimator the covariance matrices of the Gumbel order statistics were provided by P. Ypma [17], for which the author expresses its gratitude.

Because the GLS estimators are BLUE for the log-Weibull domain we will compare the performance of estimators for the location-scale parameters $\mu$ and $\sigma$. For the ML estimates we used the inverse of the transformations in Equation 4.7 to obtain the corresponding estimates for the log-Weibull domain.[1]

[1] For the R script that was used to perform the data generation and estimation see Appendix C.

## 5.3. Comparison criteria

Several prior simulation studies to compare parameter estimation for the Weibull distribution have been done for type II censored data [1] as well as for other censoring schemes [5]. Each of these studies tends to use different comparison criteria on which the conclusions were based. Below we introduce the evaluation criteria for our simulation.

### 5.3.1. Mean squared error and efficiency

One of the most commonly used metrics used for evaluating the accuracy of an estimator is the mean squared error (MSE). The MSE for an estimator $\hat{\theta}$ of a parameter $\theta$ is defined as

$$MSE\left(\hat{\theta}\right) = \mathbb{E}\left[\left(\hat{\theta} - \theta\right)^2\right]. \tag{5.1}$$

It follows that the MSE is the sum of the variance of the estimator and the square of the bias of the estimator summed:

$$\begin{aligned} MSE\left(\hat{\theta}\right) &= \mathbb{E}\left[\left(\hat{\theta} - \mathbb{E}\left[\hat{\theta}\right]\right)^2\right] + \left(\mathbb{E}\left[\hat{\theta}\right] - \theta\right)^2 \\ &= \mathrm{Var}\left(\hat{\theta}\right) + \left(\mathbb{E}\left[\hat{\theta}\right] - \theta\right)^2. \end{aligned} \tag{5.2}$$

Sometimes the root mean squared error, $RMSE(\hat{\theta}) = \sqrt{MSE(\hat{\theta})}$, is used as a measure of accuracy as this is easier to interpret, because the unit of the RMSE and the parameter are the same. In comparing two estimators we use the notion of efficiency, which is defined as follows. If $\hat{\theta}_1$ and $\hat{\theta}_2$ are two estimators for the same parameter $\theta$. Then estimator $\hat{\theta}_2$ is *more efficient* then estimator $\hat{\theta}_1$ if $MSE(\hat{\theta}_2) < MSE(\hat{\theta}_1)$. The GLS estimator of the Weibull parameters is a BLUE, thus it has zero bias. Moreover it has been shown that there exist unbiasing factors for the ML estimators (Section 4.1.1), thus creating an estimator with zero bias. Hence for the comparison of the ML estimator and the GLS estimator we will use only the variance of these estimators to compute the relative efficiency of ML with respect to GLS.

$$RE(\theta) = \frac{\mathrm{Var}\left(\hat{\theta}_{GLS}\right)}{\mathrm{Var}\left(\hat{\theta}_{ML}\right)}. \tag{5.3}$$

### 5.3.2. Predicting the next failure

In type II censored experiments it can be useful to predict when the next failure will occur. Below we derive an expression for the $p$-th quantile of the distribution of the next failure. Let $T_{1:n}, T_{2:n}, \ldots, T_{r:n}$ be the first $r$ order statistics of a random Weibull sample of size $n$. Let $t_r$ be the observed failure time of $T_{r:n}$. Then the probability of $T_{r+1:n} \leq t$ for some $t > t_r$ is

$$\begin{aligned} \mathbb{P}(T_{r+1:n} \leq t | T_{r:n} = t_r) &= 1 - \prod_{i=1}^{n-r} \mathbb{P}(T_i > t | T_i > t_r) \\ &= 1 - \left[e^{-\left(\frac{t}{\alpha}\right)^\beta + \left(\frac{t_r}{\alpha}\right)^\beta}\right]^{n-r} \\ &= 1 - \exp\left(-(n-r)\frac{t^\beta - t_r^\beta}{\alpha^\beta}\right) \end{aligned} \tag{5.4}$$

To obtain the $p$-th quantile $(t_p)$ we solve $\mathbb{P}(T_{r+1:n} \leq t | T_r = t_r) = p$ for $t$. This yields

$$\begin{aligned} t_p &= \alpha\left[\left(\frac{t_r}{\alpha}\right)^\beta - \frac{1}{n-r}\log(1-p)\right]^{1/\beta} \\ &= t_r\left[1 - \frac{\log(1-p)}{\varphi}\right]^{1/\beta}, \end{aligned} \tag{5.5}$$

where $\varphi = (n-r)\left(\frac{t_r}{\alpha}\right)^\beta$. Thus we have shown that $t_p = k(t_r, n, r, p, \alpha, \beta)$, where $k$ is some function. The plug-in method then tells us to predict the $p$-th quantile by plugging in the parameter estimators.

Hence we obtain $\hat{t}_p = k(t_r, n, r, p, \hat{\alpha}, \hat{\beta})$. With this plug-in method we can predict the $p$-th quantile using the different estimation methods. To determine the accuracy of these prediction we will use the same notions of RMSE and relative efficiency as introduced above.

# 6 Results

This chapters presents the results of the simulation carried out to compare the performance of the proposed estimators for the location scale parameter of the log Weibull distribution.

## 6.1. General Observations

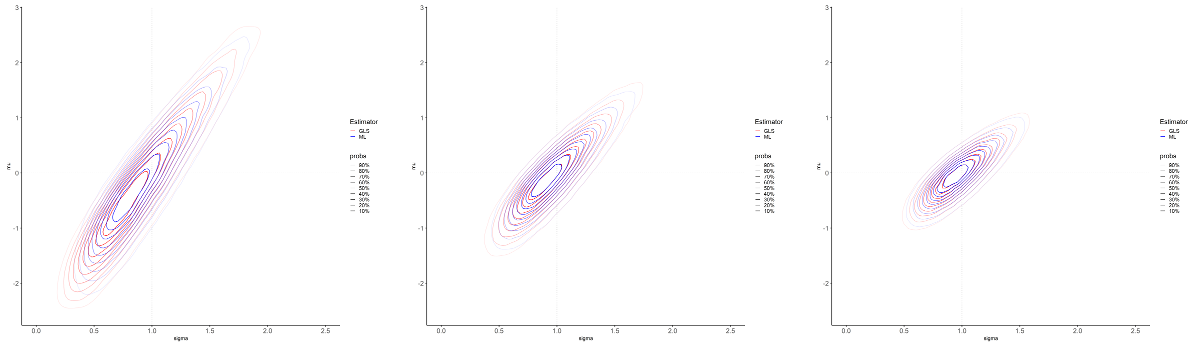
After performing the simulations we studied the estimates from the different estimation methods.[1] The ML estimator is biased, but we can estimate this bias using the simulation. The result of which are provided in Table 6.1. As explained in Section 4.1.1 there exists a pivot for the ML estimator, which can be used to create an unbiased estimator. To study the unbiased ML estimator we subtracted the estimated bias from the estimates. Below we will use the unbiased ML estimates to compare the results with the linear regression estimates.

**Table 6.1:** Bias of the estimated log-Weibull parameters for the maximum likelihood method. The maximum standard error for the listed $\hat{\mu}$'s is $0.003$ and for the $\hat{\sigma}$'s $0.0009$.

| **n** | 50 | | 75 | | 100 | | 125 | |
|---|---|---|---|---|---|---|---|---|
| **r** | **bias** $\hat{\mu}$ | **bias** $\hat{\sigma}$ | **bias** $\hat{\mu}$ | **bias** $\hat{\sigma}$ | **bias** $\hat{\mu}$ | **bias** $\hat{\sigma}$ | **bias** $\hat{\mu}$ | **bias** $\hat{\sigma}$ |
| 4 | -0.750 | -0.2461 | -0.857 | -0.2478 | -0.931 | -0.2494 | -0.986 | -0.2488 |
| 5 | -0.558 | -0.1978 | -0.640 | -0.1983 | -0.697 | -0.1990 | -0.741 | -0.1984 |
| 6 | -0.434 | -0.1644 | -0.497 | -0.1635 | -0.548 | -0.1651 | -0.581 | -0.1636 |
| 7 | -0.343 | -0.1380 | -0.408 | -0.1408 | -0.445 | -0.1403 | -0.480 | -0.1416 |
| 8 | -0.285 | -0.1221 | -0.339 | -0.1238 | -0.375 | -0.1244 | -0.403 | -0.1235 |
| 9 | -0.239 | -0.1073 | -0.284 | -0.1078 | -0.318 | -0.1091 | -0.343 | -0.1090 |
| 10 | -0.203 | -0.0965 | -0.246 | -0.0976 | -0.275 | -0.0975 | -0.299 | -0.0983 |
| 11 | -0.176 | -0.0872 | -0.216 | -0.0887 | -0.242 | -0.0895 | -0.263 | -0.0896 |
| 12 | -0.153 | -0.0797 | -0.189 | -0.0811 | -0.214 | -0.0814 | -0.232 | -0.0813 |
| 13 | -0.134 | -0.0729 | -0.170 | -0.0758 | -0.190 | -0.0752 | -0.211 | -0.0761 |
| 14 | -0.120 | -0.0678 | -0.151 | -0.0697 | -0.172 | -0.0700 | -0.191 | -0.0715 |
| 15 | -0.107 | -0.0629 | -0.136 | -0.0645 | -0.156 | -0.0651 | -0.170 | -0.0651 |

We studied an extensive set of evaluation plots to analyse the sampling distributions of the parameter estimators. As we did this, certain similarities and patterns emerged. In particular the plots displayed similar patterns for different sample sizes.

[1] For the R script that was used to analyse the simulation results see Appendix D.




**(a)** Contour plot for $n = 75, r = 5$ **(b)** Contour plot for $n = 75, r = 10$ **(c)** Contour plot for $n = 75, r = 15$

**Figure 6.1:** Contour plots of the density of the estimator pairs $(\hat{\sigma}_{ML}, \hat{\mu}_{ML})$ and $(\hat{\sigma}_{GLS}, \hat{\mu}_{GLS})$ for a sample size of $n = 75$ and three different numbers of observed values $r = 5, 10, 15$. The maximum likelihood (ML) estimates are shown in blue, and the generalised least squares (GLS) estimates are shown in red. The probabilities for each contour level are indicated in the legend.

Figure 6.1 displays two overlapping contour plots representing the densities of the estimator pairs $(\hat{\sigma}_{ML}, \hat{\mu}_{ML})$ and $(\hat{\sigma}_{GLS}, \hat{\mu}_{GLS})$ for a sample size of $n = 75$ and three different numbers of observed values: $r = 5, 10, 15$. The contour plots for other sample sizes exhibit similar shapes (see Figures B.1, B.2, and B.3). The centre of each contour plot is slightly lower and to the left of the true parameter values $(\mu = 0, \sigma = 1)$, indicating a skewness in the sampling distributions. This skewness diminishes, and the estimates become more concentrated around the true parameters as the number of observed failures increases. Moreover, the plots reveal that for small $r$ the uncertainty for $\mu$ is larger than the uncertainty for $\sigma$, and this difference in uncertainty decreases as $r$ increases.

Figure 6.1 also captures that the shapes of the sampling distributions for the ML estimator pair and the GLS estimator pair are remarkably similar across different combinations of $n$ and $r$. However, the GLS estimates exhibit slightly greater variability. Notably, this variability difference decreases as the number of observed failures increases. These observations are further supported by the marginal distributions of each estimator (see Appendices B.2 and B.3).

## 6.2. Relative efficiency estimators

Figure 6.2 shows the root mean squared error $RMSE = \sqrt{MSE(\hat{\mu})}$ for the different cases of $n$ and $r$ and all the four different estimation methods. Here we can clearly see the difference in accuracy for the different methods. Namely that the accuracy of the ML estimates is consistently better compared to all three least squares estimates. A clear trend can be observed from these figures that show that the accuracy increases with the number of observed failures. For different sample sizes the patterns of the plots are similar for each estimation method. Though the figures do show that the RMSE is higher for larger sample sizes. Comparing the RMSE of the ML estimates and the GLS estimate we see that the difference between $RMSE(\hat{\mu}_{ML})$ and $RMSE(\hat{\mu}_{GLS})$ is largest for lower values of $r$ and decreases as the number of observed failures increases. These observations can be explained by the fact that the accuracy of the estimates is adversely affected by a a greater difference between $n$ and $r$.

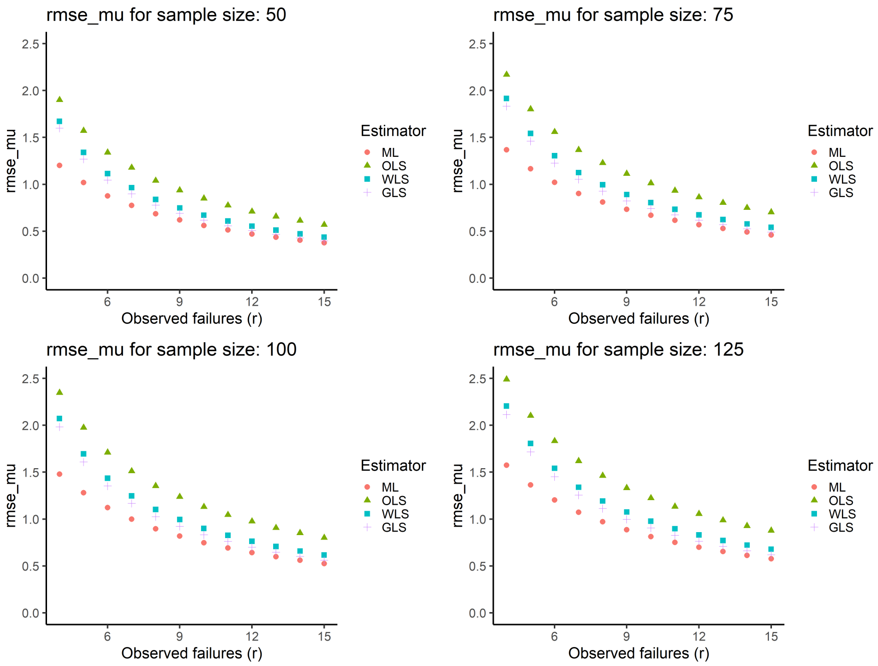



**Figure 6.2:** RMSE of estimates $\hat{\mu}$ of four different estimation methods: ML, OLS, WLS and GLS. Each subplot corresponds to a different sample size: 50, 75, 100, and 125.

In Figure 6.3 the root mean squared error of the different estimates of $\sigma$ are plotted. For estimating $\sigma$ the ML consistently results in the lowest root mean squared. Thus the ML estimator is the most accurate for all of the combinations of $n$ and $r$ that were studied. Just like for the $RMSE$ of $\hat{\mu}$ the Figure outlines a downward trend of the $RMSE$ for an increased number of observed failures. Hence the accuracy increases with $r$. Interestingly, to the eye, there appears to be no difference in the $RMSE$ of the estimates for the different sample sizes. Looking at the difference in RMSE of the ML estimates and the GLS estimates it appears the absolute difference is highest for $r = 4$ and this difference decreases as the number of observed failures increases.

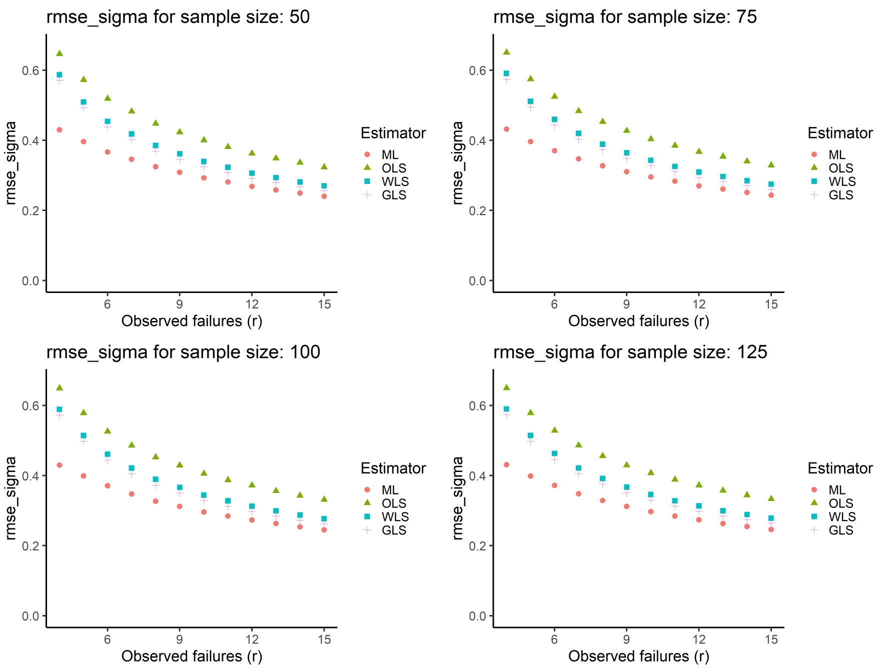
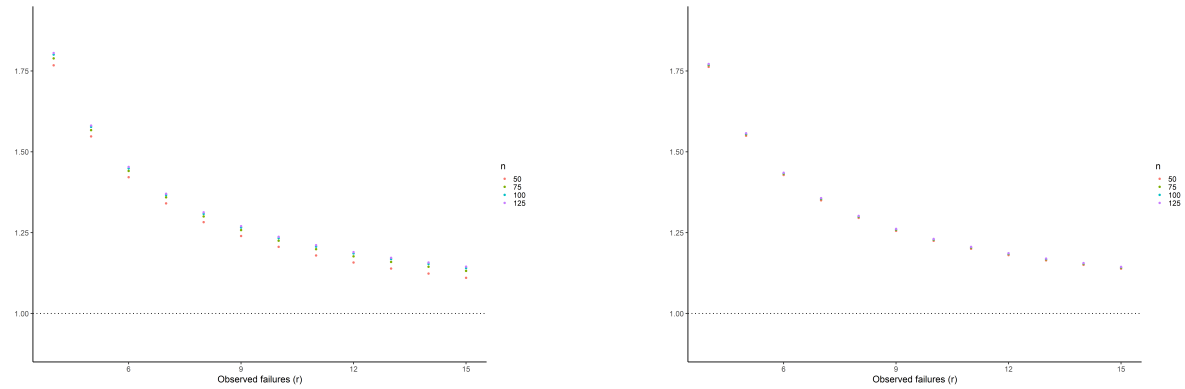



**Figure 6.3:** RMSE of estimates $\hat{\sigma}$ of four different estimation methods: ML, OLS, WLS and GLS.

Figure 6.4a displays the relative efficiency $RE = \mathrm{Var}(\hat{\mu}_{GLS})\,/\mathrm{Var}(\hat{\mu}_{ML})$ for the location parameter. The RE, which is greater then $1$ for all combinations of $n$ and $r$, shows that the ML estimator is more efficient than the GLS estimator for all of the studied cases. The RE does decrease as the number of observed failures is higher. Furthermore, from the figure it follows that, given a certain number of observed failures, the RE increases slightly as the sample size enlarges.

Figure 6.4b showcases a similar trend for the estimates of $\sigma$. This figure displays the relative efficiency, $RE = \mathrm{Var}(\hat{\sigma}_{GLS})\,/\mathrm{Var}(\hat{\sigma}_{ML})$, to be decreasing as $r$ rises. The RE, which is greater then $1$ for all combinations of $n$ and $r$, indicates that the ML estimator is a more efficient than the GLS estimator for all of the studied cases. As for different sample sizes, the figure reveals that certain number of observed failures, the RE increases just as the sample size enlarges.



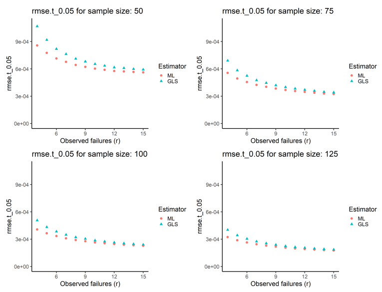

**(a)** $RE = \mathrm{Var}(\hat{\mu}_{GLS})\,/\mathrm{Var}(\hat{\mu}_{ML})$

**(b)** $RE = \mathrm{Var}(\hat{\sigma}_{GLS})\,/\mathrm{Var}(\hat{\sigma}_{ML})$

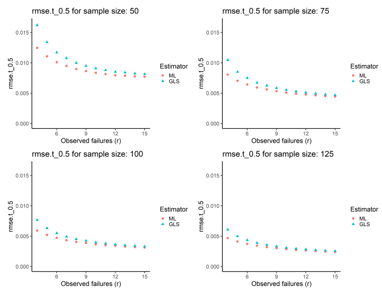
**Figure 6.4:** Relative efficiencies (GLS/ML) versus the observed number of failures for the different sample sizes $n = 50, 75, 100, 125$

## 6.3. Efficiency in estimating next failure

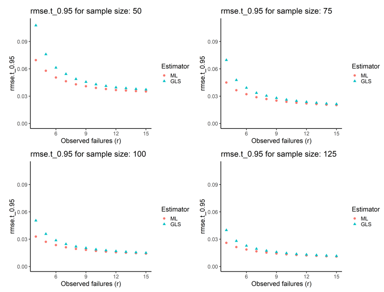
In predicting the next failure ($T_{r+1}$) we studied three different quantiles: the $5\%$-, $50\%$-, and $95\%$-quantile. Figure 6.5 depicts the root mean squared errors of the predictions for the different quantiles. These plots show a similar pattern, that indicate an improved accuracy in predicting the quantiles of the next failure as the sample size increases or the number of observed failures increases.




**(a)** RMSE for $\hat{t}_{0.05}$



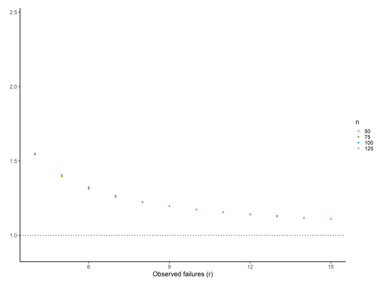

**(b)** RMSE for $\hat{t}_{0.5}$



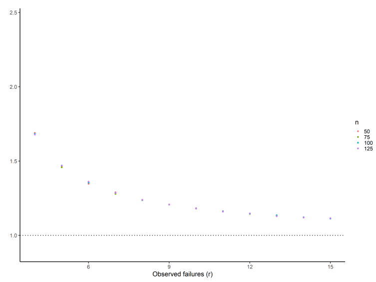

**(c)** RMSE for $\hat{t}_{0.95}$

**Figure 6.5:** RMSE of estimates $\hat{t}_p$ of ML method and GLS method. Each subplot corresponds to a different quantile: 5%, 50%, and 95%.

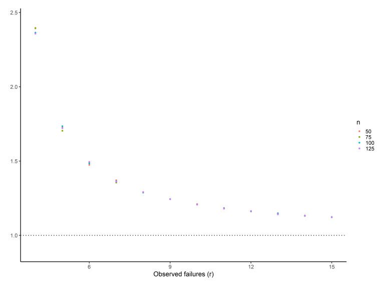
From these plots it also follows that the accuracy of the ML estimates is better than the GLS estimates for all the studied combinations of $n$ and $r$. The same conclusion can be made when analysing the relative efficiencies. Figure 6.6 displays the RE ($RE = MSE([\hat{t}_p]_{GLS})/MSE([\hat{t}_p]_{ML})$ for the prediction quantiles. $RE > 1$ for every sample size and number of observations. Hence the ML method provide for a more efficient estimator of the quantiles compared to the GLS estimation method.



**(a)** RE for $\hat{t}_{0.05}$



**(b)** RE for $\hat{t}_{0.5}$



**(c)** RE for $\hat{t}_{0.95}$

**Figure 6.6:** Relative efficiencies $RE = MSE([\hat{t}_p]_{GLS})/MSE([\hat{t}_p]_{ML})$. Each subplot corresponds to a different quantile: 5%, 50%, and 95%

# 7 Conclusion

In this thesis we compared the maximum likelihood method and generalised least squares estimation method for estimating the Weibull parameters of highly censored and small samples. A simulation was performed in which we used type II censored Weibull data to compare the estimators. The results of the simulation showed that the ML estimator for the log-Weibull parameters is more accurate compared to the GLS estimator. This is the case for the complete range of sample sizes and number of observed failures that were studied. For a higher number of observed failures the difference in accuracy of the ML estimator and the GLS estimator does decrease. However, the ML estimation method remains the more efficient estimator of the two for the total evaluation region.

Using the plugin method we were also able to calculate estimates for the quantiles of the next failure. The results of these calculation showed that the ML estimation method provides for a more accurate and efficient estimation method in predicting the quantiles of the next failures. Thus we can conclude that for our evaluation region the maximum likelihood estimator is better than the generalised least squares estimator.

As the conclusion above can only be made for the evaluation region used in our simulation study, one should be cautious when generalising these findings to sample sizes or numbers of observed failures outside this region. The robustness of the results may vary with different sample characteristics that were not part of the evaluation. Additionally, our study focused on Type II censoring, and the performance of the estimation methods could differ under other censoring schemes. Therefore, it is important to consider the specific context and characteristics of the dataset when applying these conclusions to practical scenarios. Further research is recommended to explore the behavior of these estimators across a broader range of conditions and censoring mechanisms.

# References


[1] RB Abernethy et al. “Weibull Analysis Handbook, Air Force Wright Aeronautical Laboratories Technical Report AFWAL-TR-83-2079”. In: *Available from the National Technical Information Service, Washington, DC* (1983).

[2] A. Balakrishnan N. & Clifford Cohen. *Order statistics and inference: Estimation methods*. Academic Press, Inc, 1991.

[3] E. Benton Cobb. “Estimation of the weibull shape parameter in small-sample bioassay”. In: *Journal of Statistical Computation and Simulation* 31.2 (1989), pp. 93–101. DOI: `10.1080/00949658908811126`.

[4] T. J. Engeman R. M & Keefe. “On generalized least squares estimation of the Weibull distribution”. In: *Communications in Statistics-Theory and Methods* 11.19 (1982), pp. 2181–2193.

[5] U. Genschel and W. Meeker. “A comparison of maximum likelihood and median-rank regression for Weibull estimation”. In: *Quality engineering* 22.4 (2010), pp. 236–255.

[6] H. Hirose. “Bias correction for the maximum likelihood estimates in the two-parameter Weibull distribution”. In: *IEEE Transactions on Dielectrics and Electrical Insulation* 6.1 (1999), pp. 66–68. DOI: `10.1109/94.752011`.

[7] Y. M. Kantar. “Generalized least squares and weighted least squares estimation methods for distributional parameters”. In: *REVSTAT-Statistical Journal* 13.3 (2015), pp. 263–282.

[8] Erich L Lehmann and George Casella. *Theory of point estimation*. Springer Science & Business Media, 2006.

[9] Julius Lieblein. “On the exact evaluation of the variances and covariances of order statistics in samples from the extreme-value distribution”. In: *The Annals of Mathematical Statistics* (1953), pp. 282–287.

[10] M. Matsumoto and T. Nishimura. “Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator”. In: *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 8.1 (1998), pp. 3–30.

[11] L. Meester. private communication. 2024.

[12] R. Ross. “Bias and standard deviation due to Weibull parameter estimation for small data sets”. In: *IEEE Transactions on Dielectrics and Electrical insulation* 3.1 (1996), pp. 28–42.

[13] R. Ross. *Reliability analysis for asset management of electric power grids*. John Wiley & Sons, 2019.

[14] Darrel R Thoman, Lee J Bain, and Charles E Antle. “Inferences on the parameters of the Weibull distribution”. In: *Technometrics* 11.3 (1969), pp. 445–460.

[15] W. Weibull. “A Statistical Distribution Function of Wide Applicability”. In: *ASME Journal of Applied Mechanics* 18 (July 1952), pp. 293–297.

[16] J. S. White. “The Moments of Log-Weibull Order Statistics”. In: *Technometrics* 11.2 (1969), pp. 373–386. ISSN: 00401706. URL: `http://www.jstor.org/stable/1267267` (visited on 05/24/2024).

[17] P. Ypma. private communication. 2024.

# A

# Mathematical background

## A.1. Moments of the standard Gumbel

Below we derive the expressions for the first and second moment of the Gumbel order statistics as shown in Equations (4.2) and (4.3). Let $Z_{i:n}$ be the $i$-th order statistic of a random sample of the reversed standard Gumbel distribution, with distribution function $F(z) = 1 - e^{-e^{-z}}$ and density function $f(z) = -e^{z+e^{-z}}$. Using the result by Balakrishnan and Cohen [2], and Ross [12], it follows that

$$\begin{aligned}
\mathbb{E}\left[Z_{i:n}\right] &= i\binom{n}{i}\int_{-\infty}^{\infty} z(F(z)^{i-1}(1-F(z))^{n-i}f(z)dz \\
&= i\binom{n}{i}\int_{-\infty}^{\infty} z\left(1-e^{-e^{-z}}\right)^{i-1}\left(e^{-e^{-z}}\right)^{n-i}\left(-e^{z+e^{-z}}\right)dz
\end{aligned} \tag{A.1}$$

By substituting $y = e^{-z} \iff x = -\log y$ and using the binomial identity to expand $(e^{-y}-1)$ the expression becomes [12]

$$\begin{aligned}
\mathbb{E}\left[Z_{i:n}\right] &= i\binom{n}{i}\int_{0}^{\infty} \log(y)(1-e^{-y})^{i-1}e^{-y(n-i+1)}\,dy \\
&= i\binom{n}{i}\sum_{j=0}^{i-1}\binom{i-1}{j}(-1)^{i-1-j}\int_{0}^{\infty} e^{-y(n-j)}\log(y)\,dy \\
&= i\binom{n}{i}\sum_{j=0}^{i-1}\binom{i-1}{j}(-1)^{i-1-j}\int_{0}^{\infty} e^{-(n-j)y}\left(\log[(n-j)y]-\log[n-j]\right)\frac{d(n-j)y}{n-j} \\
&= -\gamma + i\binom{n}{i}\sum_{j=0}^{i-1}\binom{i-1}{j}(-1)^{i-1-j}\log\left(\frac{n-j}{n-j}\right),
\end{aligned} \tag{A.2}$$

where $\gamma$ is Euler's constant, which comes from the definite integral

$$-\gamma = \int_{0}^{\infty} e^{-x}\log x\,dx.$$

In a similar manner, again using the result from Balakrishnan and Cohen, as well as Ross, we can compute the second moment for the $i$-th order statistic

$$
\begin{aligned}
\mathbb{E}\left[Z_{i:n}^2\right] &= i\binom{n}{i}\int_{-\infty}^{\infty} z^2 (F(z)^{i-1}(1-F(z))^{n-i} f(z) dz \\
&= i\binom{n}{i}\int_{-\infty}^{\infty} z^2 \left(1-e^{-e^{-z}}\right)^{i-1}\left(e^{-e^{-z}}\right)^{n-i}\left(-e^{z+e^{-z}}\right) dz \\
&= i\binom{n}{i}\sum_{j=0}^{i-1}\binom{i-1}{j}(-1)^{i-1-j}\int_{0}^{\infty} e^{-y(n-j)}(\log(y))^2\, dy \\
&= i\binom{n}{i}\sum_{j=0}^{i-1}\binom{i-1}{j}(-1)^{i-1-j}\int_{0}^{\infty} e^{-u}(\log(u)-\log(n-j))^2 \frac{du}{n-j} \\
&= i\binom{n}{i}\sum_{j=0}^{i-1}\binom{i-1}{j}(-1)^{i-1-j}\sum_{l=0}^{2}\binom{2}{l}\frac{(-\log(n-j))^{2-l}}{n-j}\int_{0}^{\infty}(\log(u))^l e^{-u} du \\
&= i\binom{n}{i}\sum_{l=0}^{2}\left[\binom{2}{l}(-1)^{2-m}\frac{\partial^l}{\partial s^l}\Gamma[s+1]\bigg|_{s=0}\sum_{j=0}^{i-1}\binom{i-1}{j}(-1)^{i-1-j}\left(\frac{(\log[n-j])^{2-l}}{n-j}\right)\right],
\end{aligned}
\tag{A.3}
$$

where

$$
\begin{aligned}
\Gamma^{(0)}[1] &= 1 \\
\Gamma^{(1)}[1] &= -\gamma \\
\Gamma^{(2)}[1] &= \gamma^2 + \frac{\pi^2}{6}.
\end{aligned}
$$

# B

# Simulation results

## B.1. Contour plots

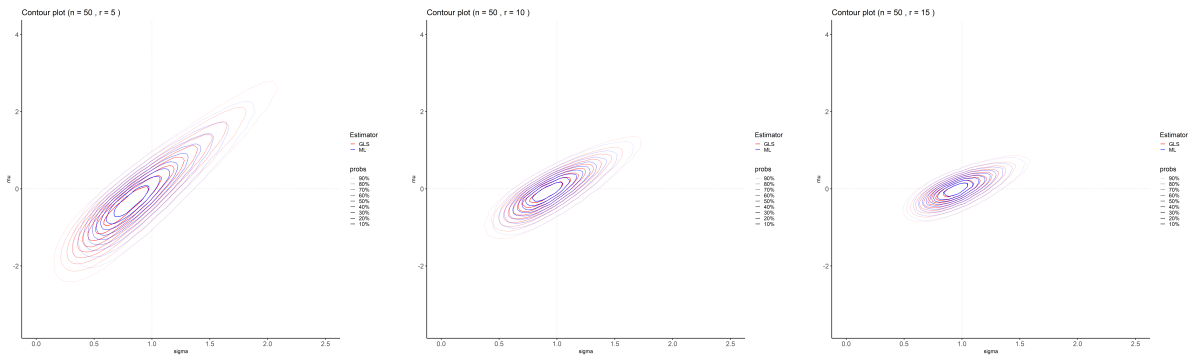



**Figure B.1:** Contour plots of the density of the estimator pairs $(\hat{\sigma}_{ML}, \hat{\mu}_{ML})$ and $(\hat{\sigma}_{GLS}, \hat{\mu}_{GLS})$ for a sample size of $n = 50$ and three different numbers of observed values $r = 5, 10, 15$. The maximum likelihood (ML) estimates are shown in blue, and the generalised least squares (GLS) estimates are shown in red. The probabilities for each contour level are indicated in the legend.

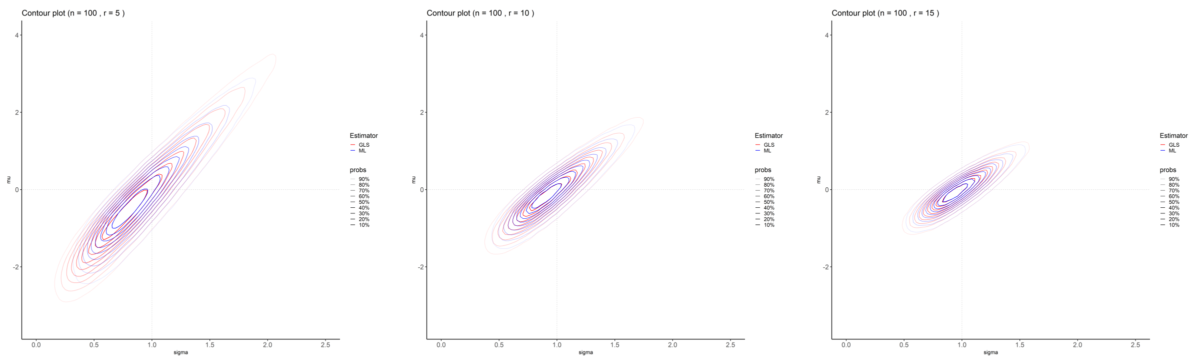



**Figure B.2:** Contour plots of the density of the estimator pairs $(\hat{\sigma}_{ML}, \hat{\mu}_{ML})$ and $(\hat{\sigma}_{GLS}, \hat{\mu}_{GLS})$ for a sample size of $n = 100$ and three different numbers of observed values $r = 5, 10, 15$. The maximum likelihood (ML) estimates are shown in blue, and the generalised least squares (GLS) estimates are shown in red. The probabilities for each contour level are indicated in the legend.

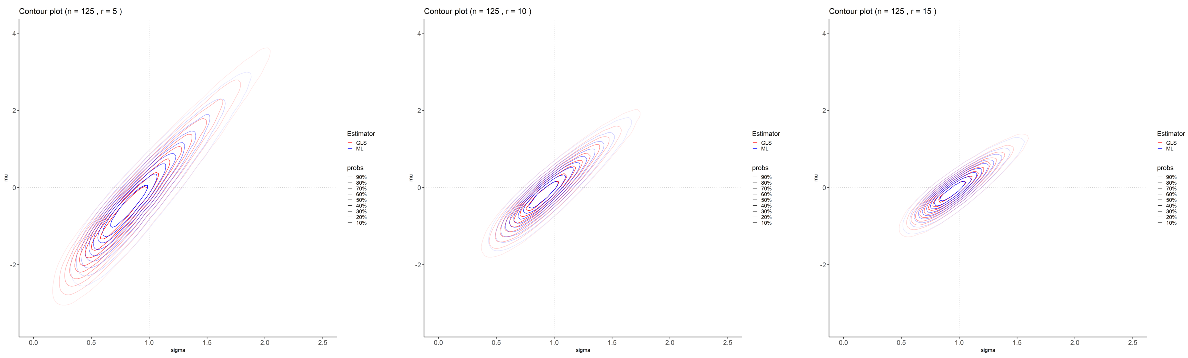



**Figure B.3:** Contour plots of the density of the estimator pairs $(\hat{\sigma}_{ML}, \hat{\mu}_{ML})$ and $(\hat{\sigma}_{GLS}, \hat{\mu}_{GLS})$ for a sample size of $n = 125$ and three different numbers of observed values $r = 5, 10, 15$. The maximum likelihood (ML) estimates are shown in blue, and the generalised least squares (GLS) estimates are shown in red. The probabilities for each contour level are indicated in the legend.

## B.2. Density plots location parameter

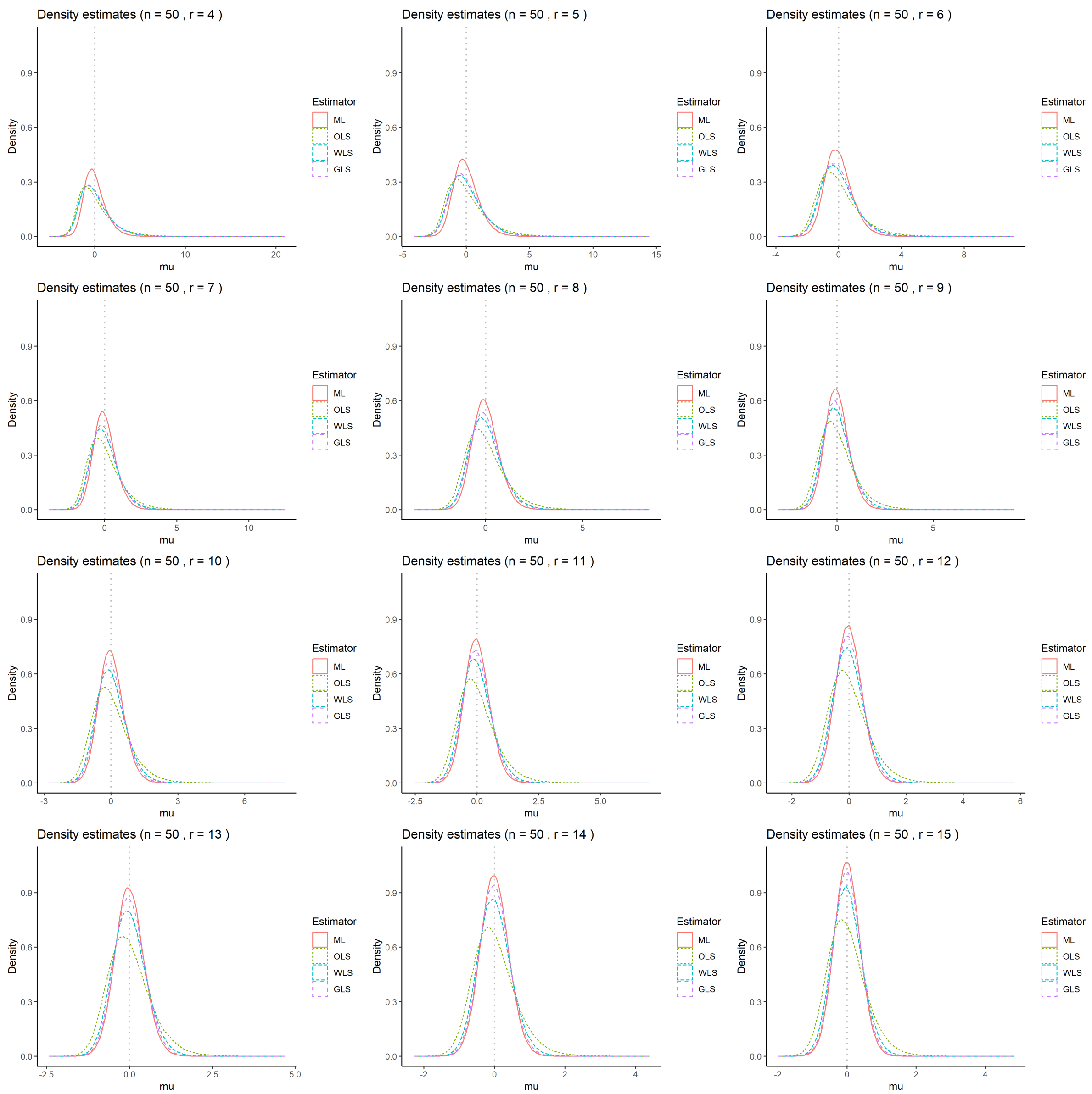



**Figure B.4:** Density of sampling distribution for different estimation methods of the location parameter ($\mu$) for sample size (n = 50) and across various number of observed failures (r = 4 to r = 15). The estimators compared are the maximum likelihood estimator (ML), ordinary least squares estimator (OLS), weighted least squares estimator (WLS), and generalised least squares estimator (GLS).

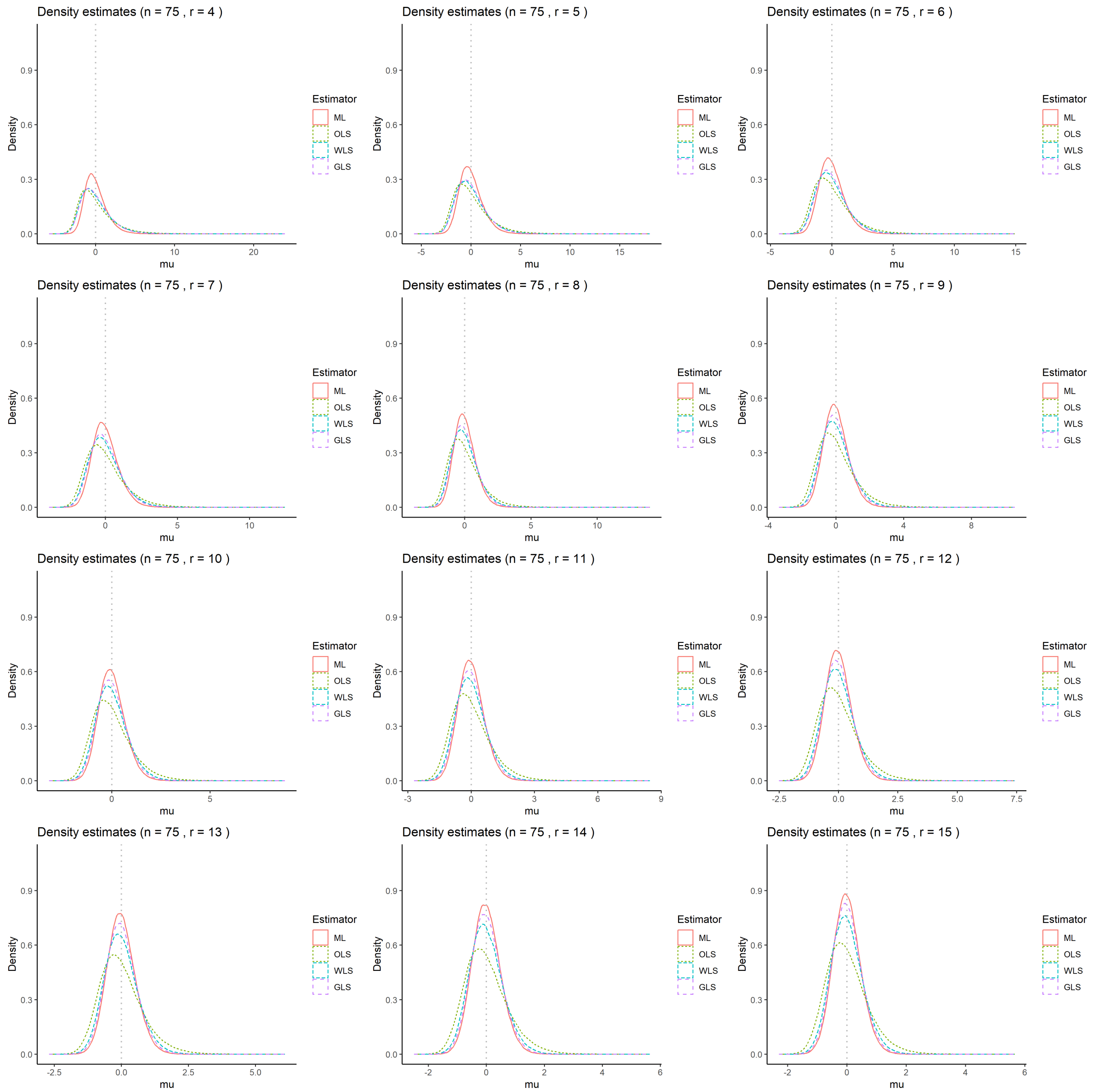



**Figure B.5:** Density of sampling distribution for different estimation methods of the location parameter ($\mu$) for sample size (n = 75) and across various number of observed failures (r = 4 to r = 15). The estimators compared are the maximum likelihood estimator (ML), ordinary least squares estimator (OLS), weighted least squares estimator (WLS), and generalised least squares estimator (GLS).

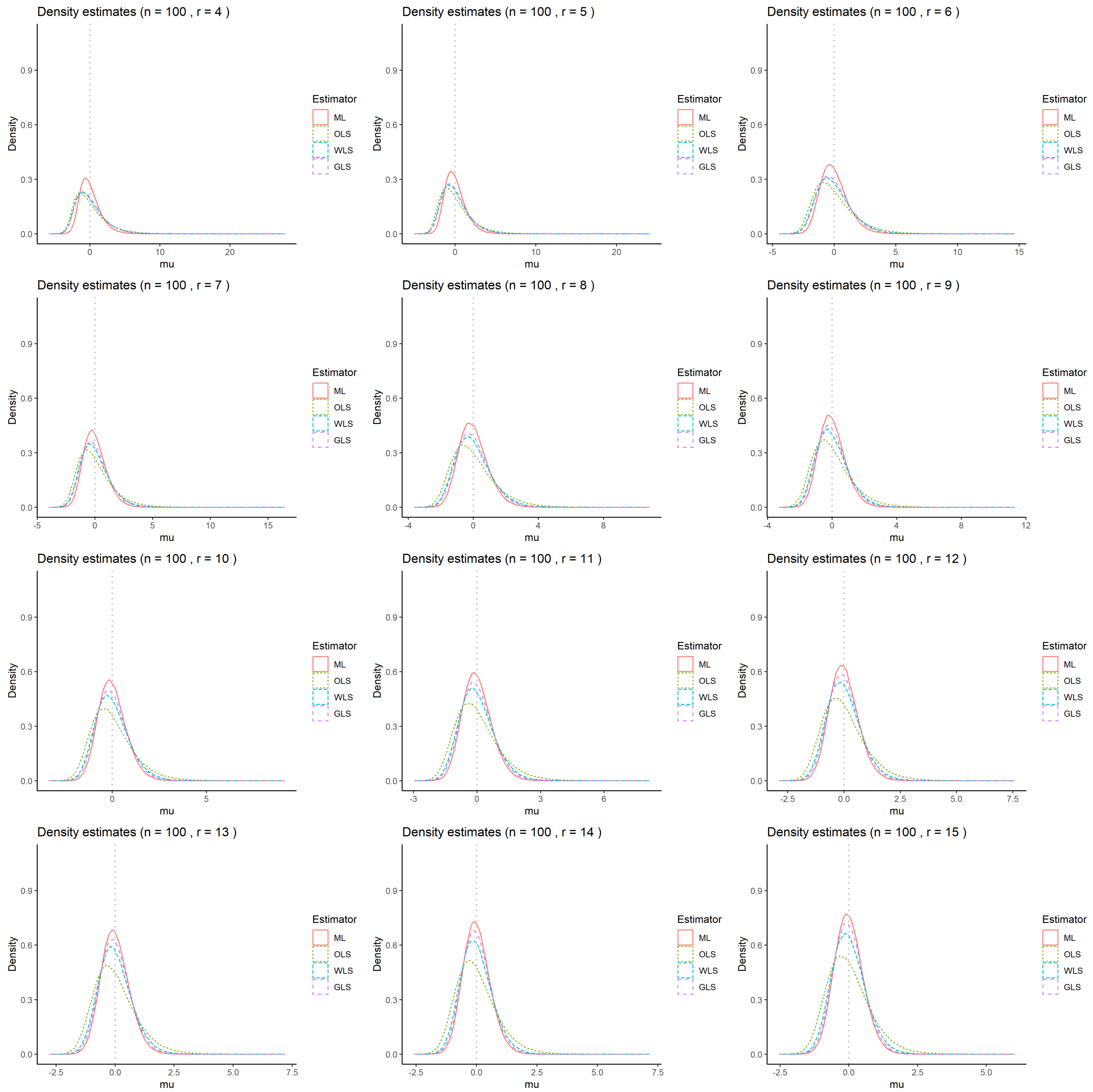



**Figure B.6:** Density of sampling distribution for different estimation methods of the location parameter ($\mu$) for sample size (n = 100) and across various number of observed failures (r = 4 to r = 15). The estimators compared are the maximum likelihood estimator (ML), ordinary least squares estimator (OLS), weighted least squares estimator (WLS), and generalised least squares estimator (GLS).

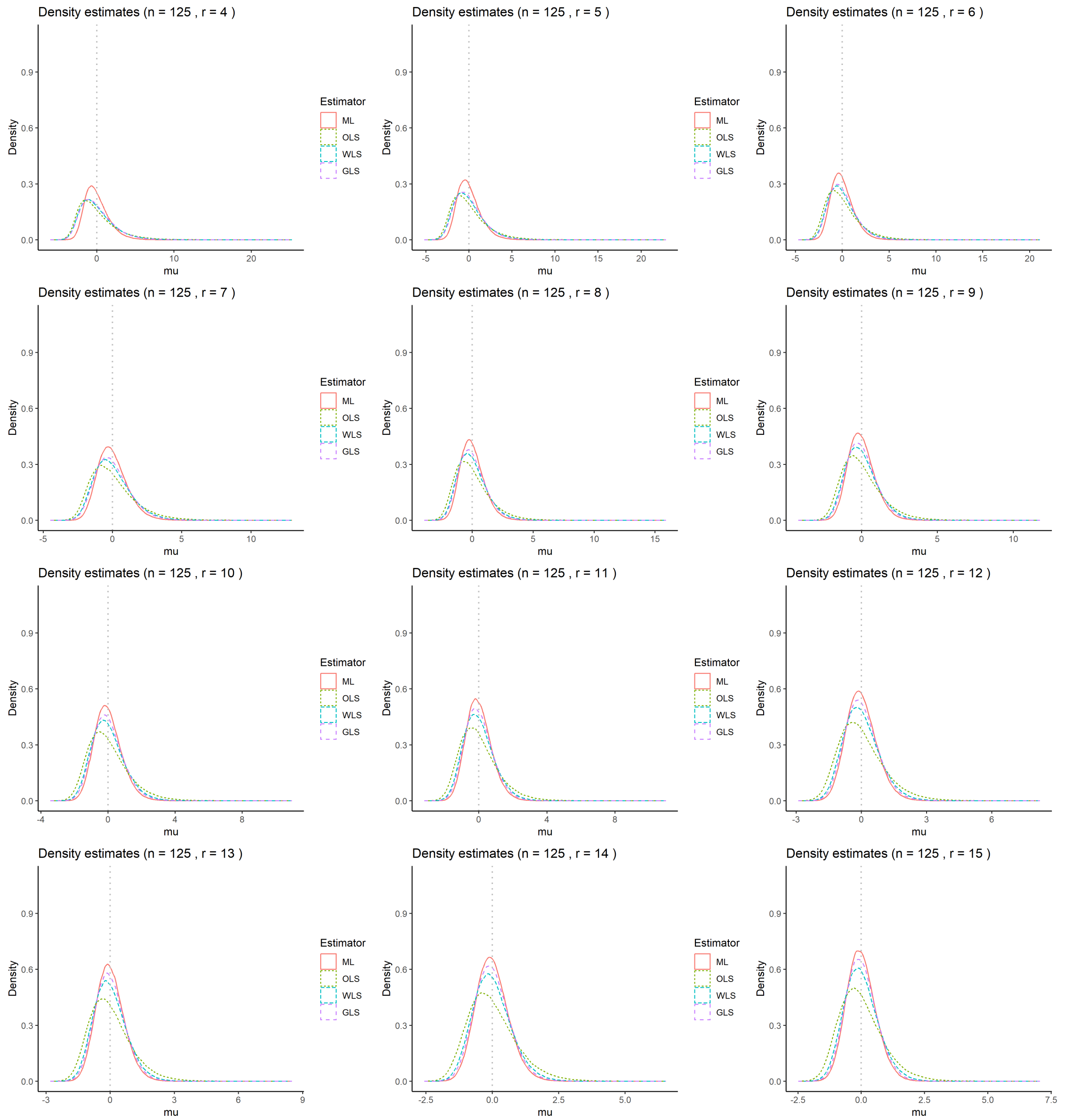



**Figure B.7:** Density of sampling distribution for different estimation methods of the location parameter ($\mu$) for sample size (n = 125) and across various number of observed failures (r = 4 to r = 15). The estimators compared are the maximum likelihood estimator (ML), ordinary least squares estimator (OLS), weighted least squares estimator (WLS), and generalised least squares estimator (GLS).

# B.3. Density plots scale parameter

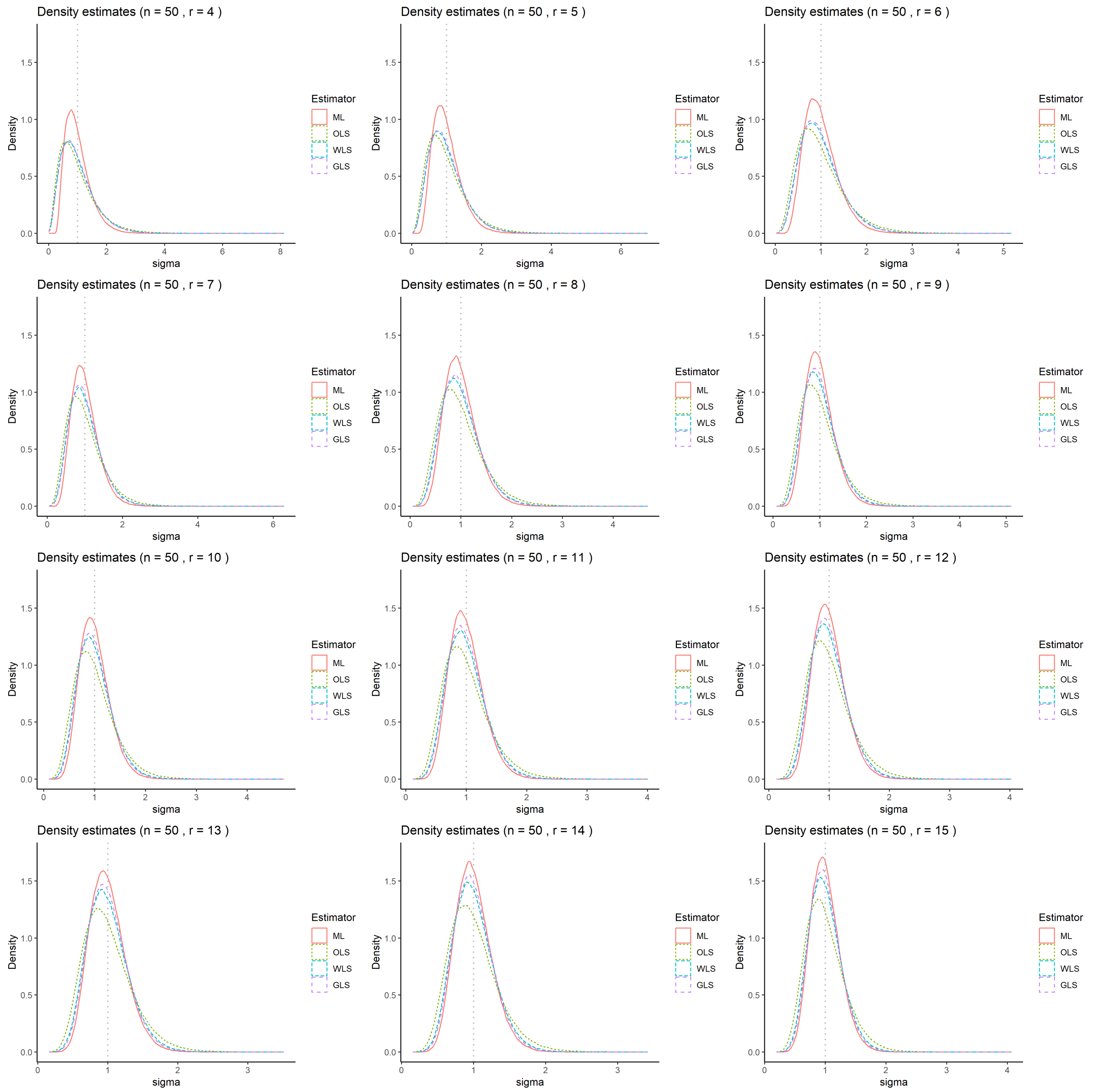



**Figure B.8:** Density of sampling distribution for different estimation methods of the scale parameter ($\sigma$) for sample size (n = 50) and across various number of observed failures (r = 4 to r = 15). The estimators compared are the maximum likelihood estimator (ML), ordinary least squares estimator (OLS), weighted least squares estimator (WLS), and generalised least squares estimator (GLS).

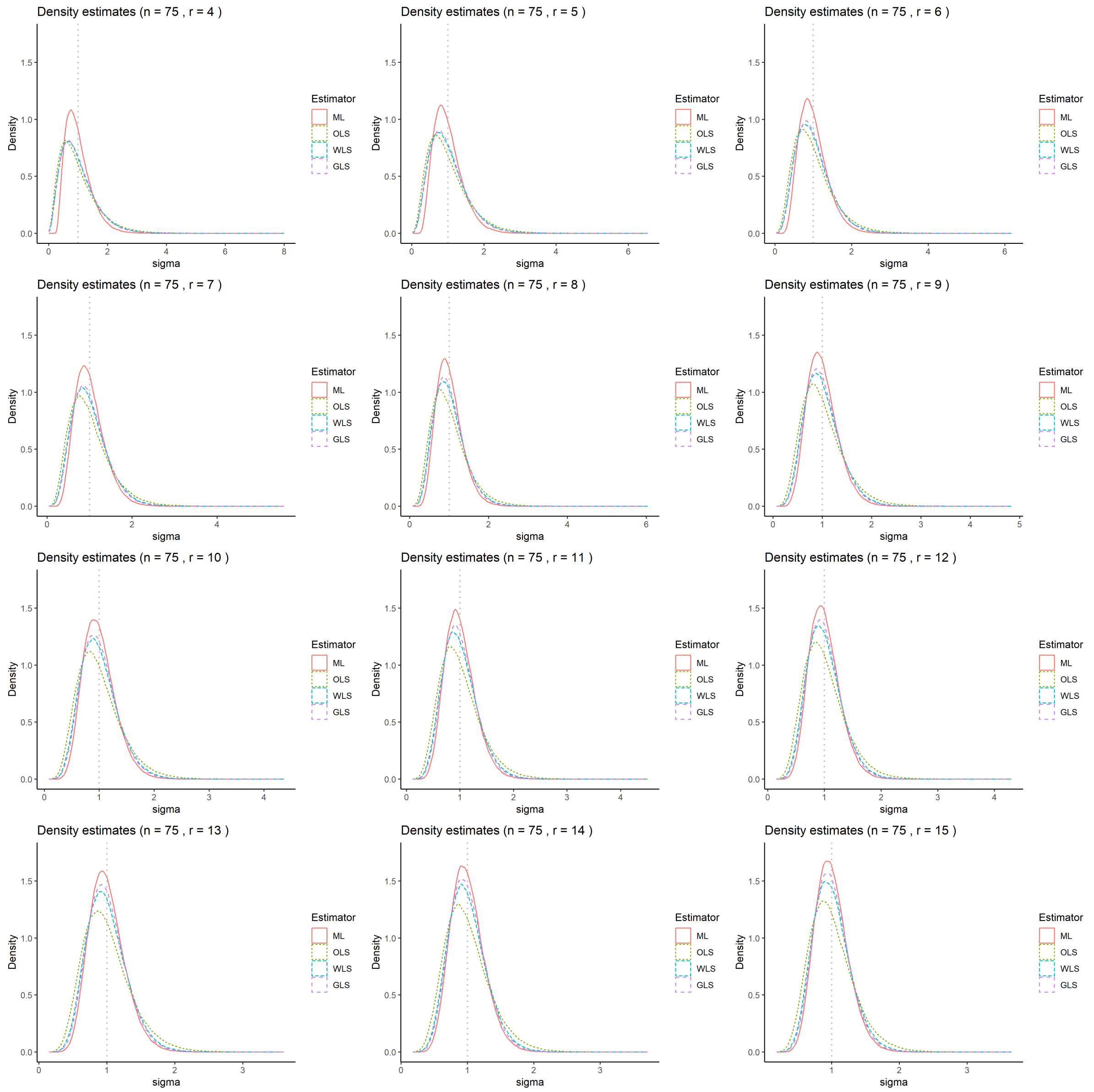



**Figure B.9:** Density of sampling distribution for different estimation methods of the scale parameter ($\sigma$) for sample size (n = 75) and across various number of observed failures (r = 4 to r = 15). The estimators compared are the maximum likelihood estimator (ML), ordinary least squares estimator (OLS), weighted least squares estimator (WLS), and generalised least squares estimator (GLS).

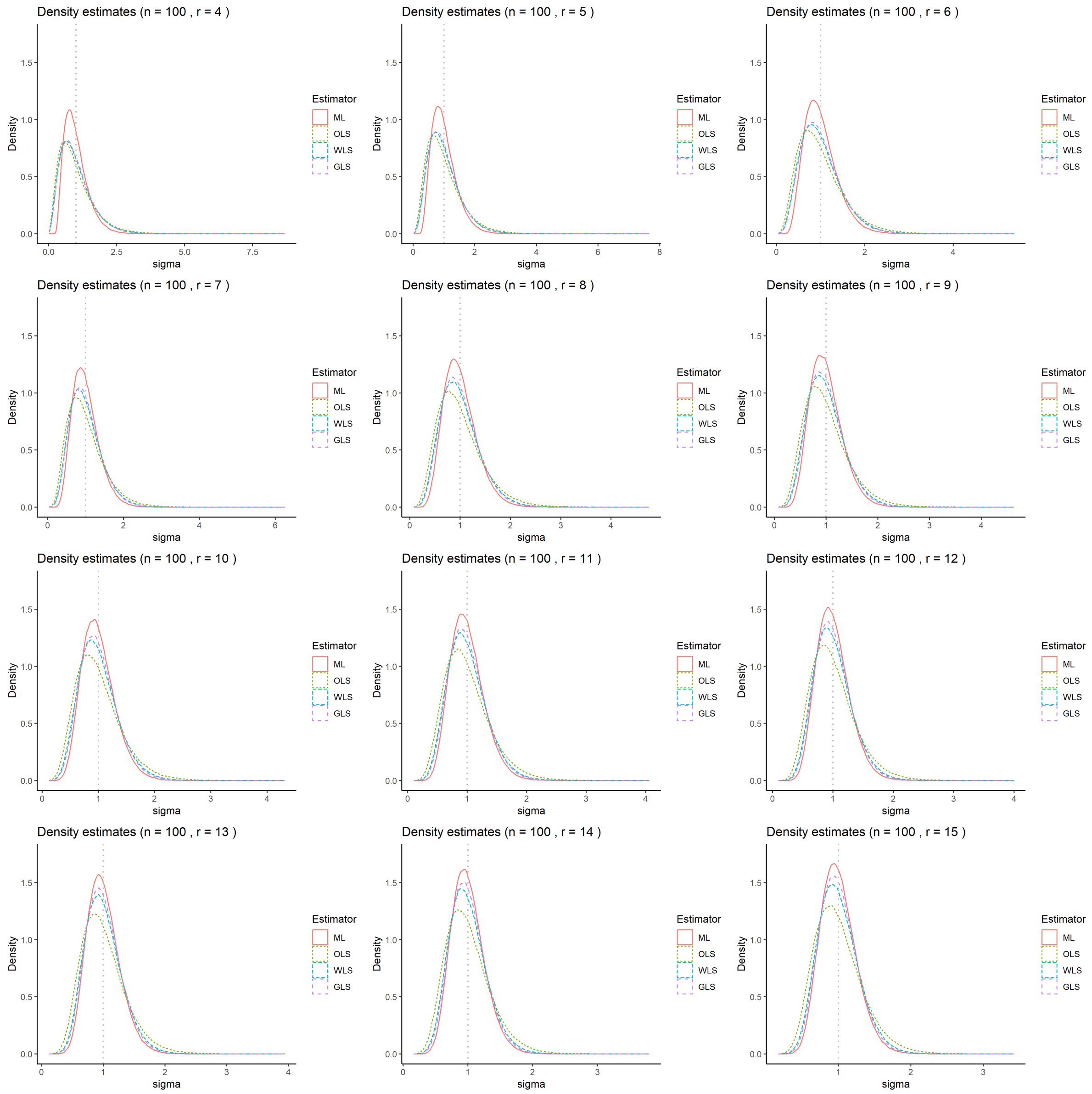



**Figure B.10:** Density of sampling distribution for different estimation methods of the scale parameter ($\sigma$) for sample size (n = 100) and across various number of observed failures (r = 4 to r = 15). The estimators compared are the maximum likelihood estimator (ML), ordinary least squares estimator (OLS), weighted least squares estimator (WLS), and generalised least squares estimator (GLS).

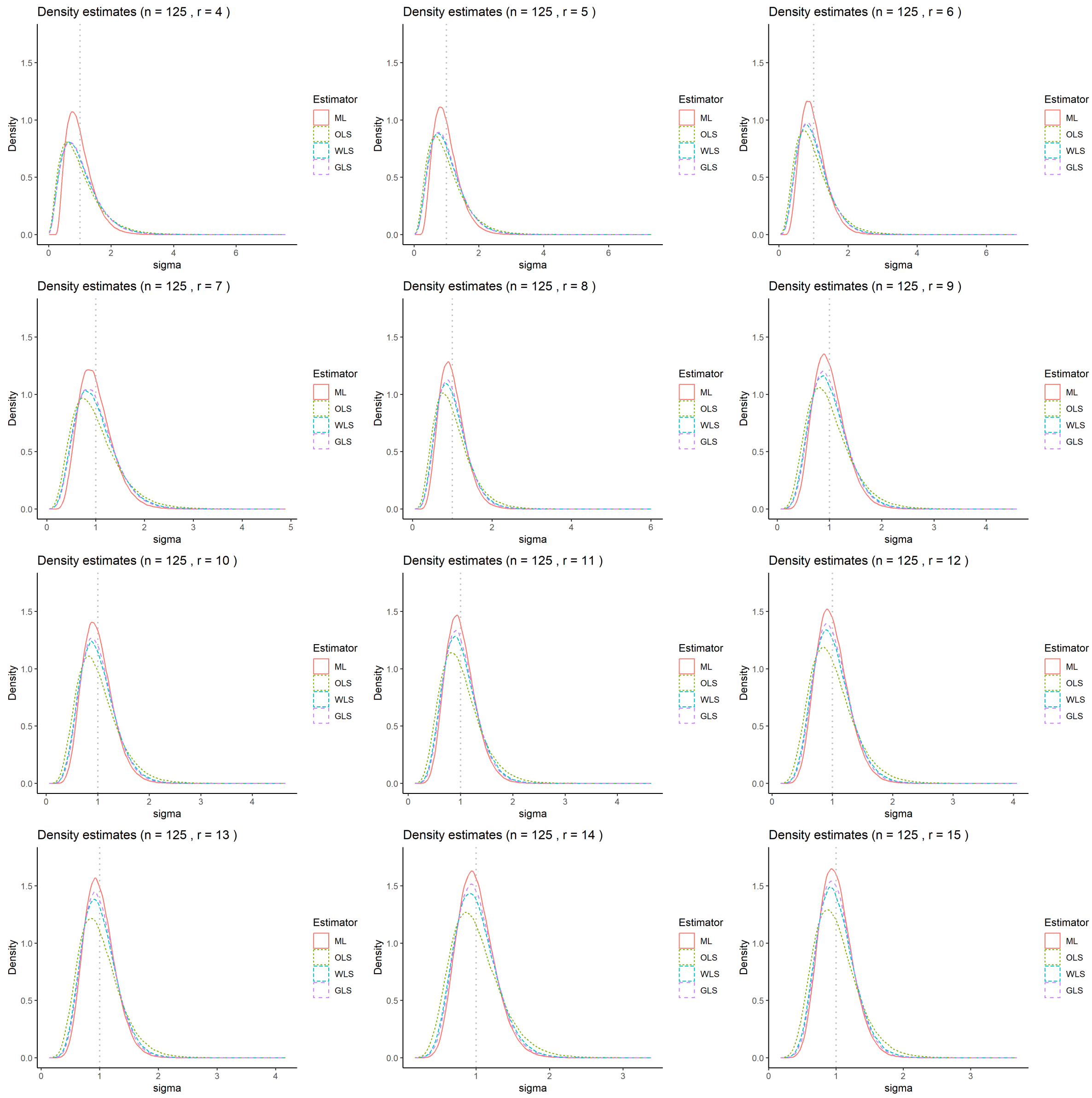



**Figure B.11:** Density of sampling distribution for different estimation methods of the scale parameter ($\sigma$) for sample size (n = 125) and across various number of observed failures (r = 4 to r = 15). The estimators compared are the maximum likelihood estimator (ML), ordinary least squares estimator (OLS), weighted least squares estimator (WLS), and generalised least squares estimator (GLS).

# C

# Simulation code

```r
library(R.matlab)
library(matlib)
library(pryr)


################################################################################
#------------------FUNCTION DEFINITIONS----------------------------------------
################################################################################

## Maximum Likelihood Estimation
ML_weibull <- function(data, n, r, N){
  ## Arguments:
  # data: matrix vector of censored Weibull data
  # n: sample size
  # r: degree of censoring
  # N: simulation sample size

  ## Returns:
  # result: dataframe with estimates for location scale parameters (log Weibull)

  ## Pre-processing for Maximum Likelihood Estimation (MLE)
  tc <- data[r,]                        # Extract the r-th order statistic (censoring threshold)
  E <- matrix(tc, r, N, byrow=TRUE)     # Repeat tc across rows for vectorized operations
  Z <- -log(data/E)                     # Compute log-residuals for MLE
  Zmeans <- apply(Z, 2, mean)           # Compute mean of log-residuals for each sample
  Zbar <- matrix(Zmeans, r, N, byrow=TRUE)  # Repeat means across rows for subsequent operations
  U <- Z/Zbar                           # Standardize residuals by their means

  ## Newton-Raphson setup for iterative MLE computation
  maxit <- 20                       # Maximum number of iterations
  tol <- 1e-10                      # Tolerance level for convergence check
  tol2 <- 1e-5                      # Secondary tolerance for convergence check
  phio <- rep(1, N)                 # Initialize all phi estimates to 1
  Phires <- phio                    # Matrix to store results of phi across iterations
  Dphi <- NULL                      # Initialise matrix to store differences in phi estimates per iteration

  ## Newton-Raphson iteration block
  for (i in 1:maxit) {
    Phis <- matrix(phio, r, N, byrow=TRUE)  # Repeat current phi estimates across rows
    wts <- exp(-Phis*U)                     # Compute weights exp(-phi * standardized residual)
    K0plusnminr <- apply(wts, 2, sum) + n - r  # Compute part of the denominator for phi update
    uwts <- U * wts                         # Weighted residuals
    K1 <- apply(uwts, 2, sum)               # Compute numerator component for phi update
    uuwts <- U * uwts                       # Squared weighted residuals
```

```
    K2 <- apply(uuwts, 2, sum)              # Compute adjustment factor for numerator
    noemer <- phio * (K2 - K1)              # Final denominator for phi update
    teller <- phio * noemer + phio * K1 + K0plusnminr  # Final numerator for phi update
    noemer <- noemer + K0plusnminr          # Add constant to denominator
    phinew <- teller / noemer               # Update phi estimates
    Phires <- rbind(Phires, phinew)         # Record new phi estimates
    dphi <- phinew - phio                   # Calculate change in phi
    Dphi <- rbind(Dphi, dphi)               # Record changes for convergence checking
    if(max(abs(dphi)) < tol) break          # Break if changes are within tolerance
    phio <- phinew                          # Update phi for next iteration
  }

  ## Computation of estimates
  betahat <- phinew / Zmeans
  # Initialise empty vector
  alphahat <- numeric(length = length(betahat))  # Initialize alphahat vector
  # Loop through each column j to compute alphahat_j
  for (j in 1:length(betahat)) {
    # Extract the j-th column of data
    x <- data[, j]

    # Apply the formula to calculate alphahat_j
    alphahat[j] <- (1/r * (sum(x^betahat[j]) + (n - r) * tc[j]^betahat[j]))^(1/betahat[j])
  }

  ## Converting to location-scale parameters
  muhat = log(alphahat)
  sigmahat = 1/betahat

  result <- data.frame("muhat" = muhat, "sigmahat" = sigmahat)
  return(result)
}

## Ordinary Least Squares Estimation
OLS_Weibull <- function(data, n, r, N, means){
  ## Arguments:
  # data: matrix vector of censored Weibull data
  # n: sample size
  # r: degree of censoring
  # N: simulation sample size
  # means: array of vectors containing the expected plotting position Z_{i:n}

  ## Returns:
  # result: dataframe with estimates for location scale parameters (log Weibull)

  ## Transform data to log(Weibull)
  Y <- log(data)

  ## Design matrix
  mean_vector = means[[n]][1:r]
  D <- cbind(rep(1, r), mean_vector)

  ## Matrix computations
  matrix_product <- inv(t(D) %*% D) %*% t(D)
  coef_matrix <- matrix_product %*% Y

  ## Creating results vector
  results <- data.frame(muhat = t(coef_matrix)[,1], sigmahat = t(coef_matrix)[,2])
  return(results)
}

## Weighted Least Squares Estimation
WLS_Weibull <- function(data, n, r, N, means, variances){
  ## Arguments:
  # data: matrix vector of censored Weibull data
  # n: sample size
  # r: degree of censoring
  # N: simulation sample size
  # means: array of vectors containing the expected value of ordered Z_{i:n} (reversed
      standard Gumbel)
```

```
  # variances: array of vectors containing the variances of ordered Z_{i:n} (reversed
      standard Gumbel)

  ## Returns:
  # result: data frame with estimates for location scale parameters (log Weibull)

  ## Transform data to log(Weibull)
  Y <- log(data)

  ## Design matrix
  mean_vector <- means[[n]][1:r]
  D <- cbind(rep(1, r), mean_vector)

  ## Weights matrix
  weights = 1/(variances[[n]][1:r])
  W = diag(weights) # Diagonal weights matrix

  ## Matrix computations
  matrix_product <- inv(t(D) %*% W %*% D) %*% t(D) %*% W
  coef_matrix <- matrix_product %*% Y

  ## Creating results vector
  results <- data.frame(muhat = t(coef_matrix)[,1], sigmahat = t(coef_matrix)[,2])
  return(results)
}

## Generalised Least Squares Estimation
GLS_Weibull <- function(data, n, r, N, means, inv_cov){
  ## Arguments:
  # data: matrix vector of censored Weibull data
  # n: sample size
  # r: degree of censoring
  # N: simulation sample size
  # means: array of vectors containing the expected value of ordered Z_{i:n} (reversed
      standard Gumbel)
  # inv_cov: array of matrices that are the inverse covariance matrices of Cov(Z_{i:n}, Z_{j:
      n}) for i,j = 1, ..., n (reversed standard Gumbel)

  ## Returns:
  # result: data frame with estimates for location scale parameters (log Weibull)

  ## Transform data to log(Weibull)
  Y <- log(data)

  ## Design matrix
  mean_vector <- means[[n]][1:r]
  D <- cbind(rep(1, r), mean_vector) # Design matrix
  B = inv(inv_cov[[n]])         # Inverse covariance matrix
  W = inv(B[1:r,1:r])

  ## Matrix computations
  matrix_product <- inv(t(D) %*% W %*% D) %*% t(D) %*% W
  coef_matrix <- matrix_product %*% Y

  ## Creating results vector
  results <- data.frame(muhat = t(coef_matrix)[,1], sigmahat = t(coef_matrix)[,2])
  return(results)
}

################################################################################
#-------------------FUNCTION DEFINITIONS---------------------------------------
################################################################################
#-------------------PREPARING ESTIMATION---------------------------------------
## Set a specific random seed
seed <- 170720241
set.seed(seed)

## Parameters
# Weibull parameters
alpha_0 <- 1          # Scale
beta_0 <- 1           # Shape
```

```r

# Sample parameters
n_values <- c(50,75,100,125)   # Sample sizes
r_values <- c(4,5,6,7,8,9,10,11,12,13,14,15)  # Degrees of censoring

# Simulation parameters
N <- 250000           # Runs of the simulation

## Loading computed values
# Loading the data from White Matlab structure
file_path <- "C:/Users/samva/OneDrive/Bureaublad/Studiejaar␣5␣2023-2024/AM3000␣Bachelorproject/Working␣Directory␣R/WhiteDataV3.mat"
white <- readMat(file_path)$white
white_mu <- sapply(seq(1, length(white), by = 5), function(n) white[[n]])
white_sigma2 <- sapply(seq(2, length(white), by = 5), function(n) white[[n]]^2)
white_w <- sapply(seq(3, length(white), by = 5), function(n) white[[n]])

# Loading the computed exp. values and variances computed in Python
mean_vector_array <- read.csv("C:/Users/samva/OneDrive/Bureaublad/Studiejaar␣5␣2023-2024/AM3000␣Bachelorproject/Project␣Folder␣Python/Project␣Folder␣Python/mean_vector_weibull.csv")
variances_array <- read.csv("C:/Users/samva/OneDrive/Bureaublad/Studiejaar␣5␣2023-2024/AM3000␣Bachelorproject/Project␣Folder␣Python/Project␣Folder␣Python/variances.csv")

#-------------------PERFORMING ESTIMATION---------------------------------------

# Loop over each combination of n and r
for (n in 100) {
  for (r in c(14,15)) {
    start_iter <- proc.time()
    # Print the simulation parameters for verification
    print(paste("alpha=", alpha_0, ",␣beta=", beta_0, ",␣r=", r, ",␣n=", n, ",␣N=", N, ",␣seed=", seed, sep=""))

    ## Generate data samples for simulation
    U <- matrix(runif(n * N), nrow = n)           # Generate an n by N matrix of uniform(0,1) random numbers
    data <- alpha_0 * (-log(1 - U))^(1 / beta_0)  # Apply the Weibull inverse CDF transformation
    sorted_data <- apply(data, 2, sort)           # Sort the data for every column
    censored_data <- sorted_data[1:r,]            # Censoring the data

    ## Perform ML-estimation
    ML_estimates <- ML_weibull(censored_data, n, r, N)

    ## Perform OLS-estimation
    OLS_estimates <- OLS_Weibull(censored_data, n, r, N, mean_vector_array)

    ## Perform WLS-estimation
    WLS_estimates <- WLS_Weibull(censored_data, n, r, N, mean_vector_array, variances_array)

    ## Perform GLS-estimation
    GLS_estimates <- GLS_Weibull(censored_data, n, r, N, mean_vector_array, white_w)

    # Create a data frame to store the result
    estimates <- data.frame("last_failtime" = censored_data[r,],
                            "ML_mu" = ML_estimates$muhat,   "ML_sigma" = ML_estimates$sigmahat,
                            "OLS_mu" = OLS_estimates$muhat, "OLS_sigma" = OLS_estimates$sigmahat,
                            "WLS_mu" = WLS_estimates$muhat, "WLS_sigma" = WLS_estimates$sigmahat,
                            "GLS_mu" = GLS_estimates$muhat, "GLS_sigma" = GLS_estimates$sigmahat
    )

    # Save the raw weibull data frame to a CSV file
    filename_raw <- sprintf("C:/Users/samva/OneDrive/Bureaublad/Studiejaar␣5␣2023-2024/AM3000␣Bachelorproject/Simulation_BEP/Results_Simulations/Simulation_%d/rawdata/rawdata_n%d_r%d_seed%d.csv", seed, n, r, seed)
    write.csv(censored_data, file = filename_raw, row.names = FALSE)
```

```

    # Save the estimates data frame to a CSV file
    filename_estimate <- sprintf("C:/Users/samva/OneDrive/Bureaublad/Studiejaar 5 2023-2024/AM3000 Bachelorproject/Simulation_BEP/Results_Simulations/Simulation_%d/estimates/estimates_n%d_r%d_seed%d.csv", seed, n, r, seed)
    write.csv(estimates, file = filename_estimate, row.names = FALSE)

    print(paste("Running time:",signif((proc.time()-start_iter)[1],3), "seconds"))
  }
}
```

# D

# Processing code

```
library(ggplot2)
library(tidyr)
library(dplyr)
library(gridExtra)
library(MASS)  # For kde2d function
library(ggdensity)
library(readr)

################################################################################
#------------------PREPROCESSING--------------------------------------------
################################################################################


## Parameters
# Weibull parameters
mu_0 <- 0          # Location (log(alpha_0))
sigma_0 <- 1            # Scale (1/beta_0)

# Sample parameters
n_values <- c(50,75,100,125)   # Sample sizes
r_values <- c(4,5,6,7,8,9,10,11,12,13,14,15)  # Degrees of censoring

# Simulation parameters
N <- 250000          # Runs of the simulation
seed <- 170720241    # Seed used in simulation

# Different methods of estimation used
methods <- c("ML", "OLS", "WLS", "GLS")

# Estimated parameters
parameters <- c("mu", "sigma")

################################################################################
#------------------LOADING SIMULATION DATA------------------------------------
################################################################################
# Initialize an empty list to store data frames
df_list <- list()

for (n in n_values){
  start_iter <- proc.time()
  for (r in r_values){
    # Construct the filename based on 'n' and 'r'
    filename <- sprintf("C:/Users/samva/OneDrive/Bureaublad/Studiejaar 5 2023-2024/AM3000 
        Bachelorproject/Simulation_BEP/Results_Simulations/Simulation_%d/estimates/estimates_
        n%d_r%d_seed%d.csv", seed, n, r, seed)
    # Read the CSV file into a dataframe
    df <-  data.frame(read.csv(filename))
    # Replacing infinite values with NA
    df <- do.call(data.frame,lapply(df, function(x) replace(x, is.infinite(x),NA)))
```

```
    df <- na.omit(df)
    # Unbiasing the ML estimates
    df[["ML_mu"]] <- df[["ML_mu"]] - (mean(df[["ML_mu"]] - mu_0, na.rm = TRUE))
    df[["ML_sigma"]] <- df[["ML_sigma"]] - mean(df[["ML_sigma"]] - sigma_0, na.rm = TRUE)
    # Append the dataframe to the list
    df_list[[paste0("df_n", n, "_r", r)]] <- df
    # Rename the dataframe to a unique name based on 'n' and 'r'
    assign(paste0("df_n", n, "_r", r), df)
  }
  print(paste("Running␣time:",signif((proc.time()-start_iter)[1],3), "seconds"))
}

combined_df <- bind_rows(df_list, .id = "source") %>%
  mutate(
    n = as.numeric(sub("df_n([0-9]+)_r([0-9]+)", "\\1", source)),
    r = as.numeric(sub("df_n([0-9]+)_r([0-9]+)", "\\2", source))
  ) %>%  dplyr::select(-source)

# Reshape data to long format
censored_df <- combined_df %>%
  pivot_longer(
    cols = starts_with("ML") | starts_with("OLS") | starts_with("WLS") | starts_with("GLS"),
    names_to = c("estimator", ".value"),
    names_pattern = "(.*)_(.*)"
  )

# Set the order of the estimators
censored_df$estimator <- factor(censored_df$estimator, levels = c("ML", "OLS", "WLS", "GLS"))

################################################################################
#-------------------FUNCTION DEFINITIONS----------------------------------------
################################################################################

## Function to create scatter plot for given sample size and performance metric
plot_performance_metric <- function(n, metric, data = censored_df){
  ## Arguments:
  # n: Sample size
  # metric: String that gives the name of the performance metric
  # data: performance metric dataframe containing at least a column n and

  ## Returns:
  # plot: scatterplot of the values of the performance metric for a sample size n.

  # Filter data based on sample size
  data <- data %>% filter(n == !!n)

  # Create scatter plot
  plot <- ggplot(data, aes_string(x = "r", y = metric, color = "estimator", shape = "estimator")) +
    geom_point(size = 3) +
    labs(
      title = paste(metric, "for␣sample␣size:", n),
      x = "Observed␣failures␣(r)",
      y = metric,
      color = "Estimator",
      shape = "Estimator"
    ) +
    theme_classic(base_size = 20)

  return(plot)
}

## Function to create density plots for a given sample size n and degree of censoring r
plot_density_curves <- function(n, r_values, param, data = censored_df, y_limits = NULL) {
  ## Arguments:
  # n: Sample size
  # r_values: list
  # param: parameter for which the density plot should be shown
  # data: simulation dataframe

  ## Returns:
```

```r
  # plots: grid of density plots of the estimates of param for sample size n.

  plots <- list()
  for (r in r_values) {
    # Filter data based on sample size and degree of censoring
    data <- censored_df %>% filter(n == !!n, r == !!r)

    # Create density plot
    p <- ggplot(data, aes_string(x = param, color = "estimator", linetype = "estimator")) +
      geom_density(alpha = 0.7) +
      labs(
        title = paste("Density estimates (n =", n, ", r =", r, ")"),
        x = param,
        y = "Density",
        color = "Estimator",
        linetype = "Estimator"
      ) +
      theme_classic(base_size = 20)

    if (param == "mu") {
      p <- p + geom_vline(xintercept = mu_0, linetype = "dotted", color = "grey", linewidth = 0.75)
    }
    if (param == "sigma") {
      p <- p + geom_vline(xintercept = sigma_0, linetype = "dotted", color = "grey", linewidth = 0.75)
    }


    # Adjust x-axis limits if provided
    if (!is.null(y_limits)) {
      p <- p + ylim(y_limits)
    }

    plots[[paste0("plot_", r)]] <- p
  }
  return(plots)
}

#----------------------------Relative efficiency plots-------------------------

# Create plot of Relative Efficiencies
RE_scatter <- function(n, param_col, RE = RE){
  RE_filtered <- RE %>% filter(n == !!n)
  points <- geom_point(data = RE_filtered, aes_string(x = "r", y = param_col, color = as.factor(n)), size = 2)
  curve <- geom_smooth(data = RE_filtered, aes_string(x = "r", y = param_col, color = as.factor(n)), method = "loess", se = FALSE, linetype = 2, linewidth = 0.5)
  return(list(points, curve))
}

RE_plot <- function(n, param_col, RE = RE){
  scatter_elements <- RE_scatter(n, param_col, RE)
  plot <- ggplot() +
    geom_hline(yintercept = 1, linetype = "dotted", color = "black", linewidth = 1) +
    labs(
      title = paste("Relative Efficiencies for", param_col),
      x = "Observed failures (r)",
      y = " ",
      color = "n"  # Label for the legend
    ) +
    theme_classic(base_size = 20) +
    scatter_elements[[1]]
}


#----------------------------Contour plot functions----------------------------

## Function to create a geom_contour layer using 2d kernel density estimation
contour_layer <- function(n, r, x_input, y_input, probs = seq(0.1, 0.9, by = 0.1), data = censored_df, color = 'blue') {
```

```r
  ## Arguments:
  # n: Sample size
  # r: Degree of censoring
  # x_input: list containing parameter and estimator combination for horizontal axis
  # y_input: list containing parameter and estimator combination for vertical axis
  # quantiles: Quantile levels for contour plots
  # data: simulation dataframe
  # color: color for the contour lines

  ## Returns:
  # layer: ggplot layer containing the contour lines

  # Extracting values
  param_x <- x_input[1]
  param_y <- y_input[1]
  estimator_x <- x_input[2]
  estimator_y <- x_input[2] # Note: y estimator should be same as x estimator for correct contour combination

  # Filter data based on sample size and degree of censoring
  filtered_df_x <- data %>% filter(n == !!n, r == !!r, estimator == !!estimator_x)
  filtered_df_y <- data %>% filter(n == !!n, r == !!r, estimator == !!estimator_y)

  # Check if the filtered data frames have the same length
  if (nrow(filtered_df_x) != nrow(filtered_df_y)) {
    stop("Filtered data frames for x and y have different lengths.")
  }

  # Combine filtered data for x and y
  combined_data <- data.frame(
    x = filtered_df_x[[param_x]],
    y = filtered_df_y[[param_y]],
    estimator = estimator_x  # Add estimator information
  )

  # Create ggplot layer using the kde and the contour levels
  layer <- geom_hdr_lines(data = combined_data, mapping = aes(x = x, y = y, colour = estimator), probs = c(0.9,0.8,0.7,0.6,0.5,0.4,0.3,0.2,0.1), linewidth = 0.75)
  return(layer)
}


contour_plot <- function(n, r, x1_input, y1_input, x2_input = NULL, y2_input = NULL, data = censored_df, probs =  seq(0.1, 0.9, by = 0.1)) {
  ## Arguments:
  # n: Sample size
  # r_values: list
  # x_input: list containing parameter and estimator combination for horizontal axis c(param_x, estimator_x)
  # y_input: list containing parameter and estimator combination for vertical axis c(param_y, estimator_y)
  # data: simulation dataframe

  ## Returns:
  # plots: grid of density plots of the estimates of param for sample size n.

  ## Contour plot for one x and y pair
  if (is.null(x2_input) & is.null(y2_input)) {
    # Create geom_contour layer
    contour_layer <- contour_layer(n, r, x1_input, y1_input, data, probs = probs)

    # Plot the contour plot
    plot <- ggplot() +
      contour_layer +
      coord_fixed() +           # Ensure the same scale on both axes
      geom_abline(slope = 1, intercept = 0, linetype = "dashed") +  # Add x=y dashed line
      labs(
        title = paste("Contour plot (n =", n, ", r =", r, ")"),
        x = paste(x1_input[1], x1_input[2]),
        y = paste(y1_input[1], y1_input[2])
      ) +
```

```
      theme_classic(base_size = 20) +
      theme(
        axis.text = element_text(size = 14),  # Adjust font size of axis text
        axis.title = element_text(size = 16),  # Adjust font size of axis titles
        aspect.ratio = 1  # Ensure the plot is square
      )
    return(plot)
  }

  ## Overlapping contour plot
  else {
    # Extracting values
    param_x1 <- x1_input[1]
    param_y1 <- y1_input[1]
    estimator_x1 <- x1_input[2]
    estimator_y1 <- y1_input[2]
    param_x2 <- x2_input[1]
    param_y2 <- y2_input[1]
    estimator_x2 <- x2_input[2]
    estimator_y2 <- y2_input[2]

    # Create the geom_contour layers
    contour1 <- contour_layer(n, r, x1_input, y1_input, data, color = 'blue', probs = probs)
    contour2 <- contour_layer(n, r, x2_input, y2_input, data, color = 'red', probs = probs)

    # Plot with correct legend
    plot <- ggplot() +
      # Horizontal and vertical lines at mu_0 and sigma_0
      geom_hline(yintercept = 0, linetype = "dotted", color = "grey", linewidth = 0.75) +
      geom_vline(xintercept = 1, linetype = "dotted", color = "grey", linewidth = 0.75) +
      contour2 +
      contour1 +
      coord_fixed() +
      labs(
        title = paste("Contour plot (n =", n, ", r =", r, ")"),
        x = paste(param_x1),
        y = paste(param_y1),
        colour = "Estimator"
      ) +
      scale_colour_manual(values = setNames(c("blue", "red"), c(estimator_x1, estimator_x2)))
          +  # Manually set the colours with correct names
      theme_classic(base_size = 20) +
      theme(
        axis.text = element_text(size = 14),  # Adjust font size of axis text
        axis.title = element_text(size = 16),  # Adjust font size of axis titles
        aspect.ratio = 1  # Ensure the plot is square
      )
    return(plot)
  }
}
#----------------------------QUANTILE NEXT FAILURE------------------------------
## Fucntion to compute the quantile of the next failure
q_next_failure <- function(t_r, n, r, p, mu, sigma){
  ## Arguments
  # t_r: last uncensored failure time (r-th ordered observation)
  # n: Sample size
  # r: Degree of censoring
  # p: probability for quantile
  # alpha: scale parameter Weibull distr.
  # beta: shape parameter Weibull distr.

  ## Returns:
  # t_p: p-th quantile of next failure (r+1)
  phi <- (n - r) * (t_r / exp(mu))^(1/sigma)
  t_p <- t_r * (1 - log(1 - p) / phi)^(sigma)
  return(t_p)
}

################################################################################
#-------------------MAIN PROGRAMM-----------------------------------------------
################################################################################
```

```r

#--------------------PERFORMANCE METRICS----------------------------------------
# Calculate the bias and variance of the estimators for the censored case
performance_metrics_censored <- censored_df %>%
  group_by(n, r, estimator) %>%
  summarise(
    mean_mu = mean(mu),
    bias_mu = mean(mu - mu_0),
    var_mu  = var(mu),
    SE_mu   = sqrt(var(mu))/sqrt(N),
    mse_mu  = mean((mu-mu_0)^2),
    rmse_mu = sqrt(mean((mu-mu_0)^2)),
    mad_mu  = mean((mu-mu_0)),
    mean_sigma  = mean(sigma),
    bias_sigma  = mean(sigma - sigma_0),
    var_sigma   = var(sigma),
    SE_sigma    = sqrt(var(sigma))/sqrt(N),
    mse_sigma   = mean((sigma-sigma_0)^2),
    rmse_sigma  = sqrt(mean((sigma - sigma_0)^2)),
    mad_sigma   = mean((sigma-sigma_0))
  )



# # Saving performance metrics df
# filename <- sprintf("C:/Users/samva/OneDrive/Bureaublad/Studiejaar 5 2023-2024/AM3000 Bachelorproject/Simulation_BEP/Results_Simulations/Simulation_%d/performance_metrics.csv", seed)
# write.csv(performance_metrics_censored, filename, row.names = FALSE)

metrics = c("bias_mu", "var_mu", "SE_mu", "rmse_mu", "bias_sigma", "var_sigma", "SE_sigma", "rmse_sigma")

for (metric in "rmse_mu"){
  # Create a list of plots of one performance metric
  plots_metric <- lapply(n_values, function(n) plot_performance_metric(n = n, metric = metric, performance_metrics_censored) + ylim(0,2.5))
  # Arrange plots in a grid
  metric_plot_grid <- do.call(grid.arrange, c(plots_metric, nrow = 2))
  # Display the figure
  metric_plot_grid

  # Save the complete grid to a PNG file
  output_filename <- sprintf("C:/Users/samva/OneDrive/Bureaublad/Studiejaar␣5␣2023-2024/AM3000␣Bachelorproject/Simulation_BEP/Results_Simulations/Simulation_%d/plots/plot_%s.png", seed, metric)
  ggsave(output_filename, plot = metric_plot_grid, width = 16, height = 12)
}

#-----------------------RELATIVE EFFICIENCIES-----------------------------------
# Filtering performance metrics per estimation method
metrics_ML  = performance_metrics_censored %>% filter(estimator == !!"ML")
metrics_OLS = performance_metrics_censored %>% filter(estimator == !!"OLS")
metrics_WLS = performance_metrics_censored %>% filter(estimator == !!"WLS")
metrics_GLS = performance_metrics_censored %>% filter(estimator == !!"GLS")

# Create Relative Efficiency Dataframe
RE = data.frame(n = metrics_ML$n, r = metrics_ML$r,
                RE_mu = metrics_GLS$var_mu/metrics_ML$var_mu,
                RE_sigma = metrics_GLS$var_sigma/metrics_ML$var_sigma)

## Plotting relative efficiencies for sigma
RE_plot_mu <- ggplot() +
  geom_hline(yintercept = 1, linetype = "dotted", color = "black", linewidth = 1) +
  labs(
    title = paste("Relative␣Efficiencies␣for␣mu"),
    x = "Observed␣failures␣(r)",
    y = "␣",
    color = "n"  # Label for the legend
  ) +
  theme_classic(base_size = 20) +
```

```r
  ylim(0.9,1.9)

# Assume n_values is defined
colors <- c("blue", "red", "green", "purple")

for (i in 1:length(n_values)){
  n <- n_values[i]
  scatter_elements_mu <- RE_scatter(n, "RE_mu", RE)
  RE_plot_mu <- RE_plot_mu + scatter_elements_mu[[1]] #+ scatter_elements_mu[[2]]
}

RE_plot_mu
# Save RE_plot_mu
output_filename <- sprintf("C:/Users/samva/OneDrive/Bureaublad/Studiejaar 5 2023-2024/AM3000 Bachelorproject/Simulation_BEP/Results_Simulations/Simulation_%d/plots/RE_plot_mu.png", seed)
ggsave(output_filename, plot = RE_plot_mu, width = 16, height = 12)

## Plotting relative efficiiencies for sigma
RE_plot_sigma <- ggplot() +
  geom_hline(yintercept = 1, linetype = "dotted", color = "black", linewidth = 1) +
  labs(
    title = paste("Relative Efficiencies for sigma"),
    x = "Observed failures (r)",
    y = " ",
    color = "n"  # Label for the legend
  ) +
  theme_classic(base_size = 20) +
  ylim(0.9,1.9)

# Assume n_values is defined
colors <- c("blue", "red", "green", "purple")

for (i in 1:length(n_values)){
  n <- n_values[i]
  scatter_elements_sigma <- RE_scatter(n, "RE_sigma", RE)
  RE_plot_sigma <- RE_plot_sigma + scatter_elements_sigma[[1]] #+ scatter_elements_sigma[[2]]
}

RE_plot_sigma
# Save RE_plot_sigma
output_filename <- sprintf("C:/Users/samva/OneDrive/Bureaublad/Studiejaar 5 2023-2024/AM3000 Bachelorproject/Simulation_BEP/Results_Simulations/Simulation_%d/plots/RE_plot_sigma.png", seed)
ggsave(output_filename, plot = RE_plot_sigma, width = 16, height = 12)




#---------------------------PLOTTING DENSITY CURVES-----------------------------
r_densities = r_values

# Create and save density plots for each combination of n and r
for (n in n_values) {
  densities_sigma <- plot_density_curves(n, r_densities, param = "sigma", y_limits = c(0,1.75))

  # Arrange plots in a grid
  grid_densities_sigma <- do.call(grid.arrange, c(densities_sigma, nrow = 4))
  # Show grid of plots
  grid_densities_sigma

  # Save the complete grid to a PNG file
  output_filename <- sprintf("C:/Users/samva/OneDrive/Bureaublad/Studiejaar 5 2023-2024/AM3000 Bachelorproject/Simulation_BEP/Results_Simulations/Simulation_%d/plots/density_plots_%d_sigma.png", seed, n)
  ggsave(output_filename, plot = grid_densities_sigma, width = 16, height = 16)
}

# Create and save density plots for each combination of n and r
for (n in n_values) {
```

```
  densities_mu <- plot_density_curves(n, r_densities, param = "mu", y_limits = c(0,1.1))

  # Arrange plots in a grid
  grid_densities_mu <- do.call(grid.arrange, c(densities_mu, nrow = 4))
  # Show grid of plots
  grid_densities_mu

  # Save the complete grid to a PNG file
  output_filename <- sprintf("C:/Users/samva/OneDrive/Bureaublad/Studiejaar␣5␣2023-2024/AM3000␣Bachelorproject/Simulation_BEP/Results_Simulations/Simulation_%d/plots/density_plot_%d_mu.png", seed, n)
  ggsave(output_filename, plot = grid_densities_mu, width = 16, height = 16)
}

#--------------------------CONTOUR PLOTS---------------------------------------

## Contour plot for sigma (ML, GLS)
for (n in n_values){
  for (r in r_values){
    p <- contour_plot(n, r, c("sigma", "ML"), c("sigma", "GLS"))
    print(p)

    # Saving the contour plots individually
    output_filename <- sprintf("C:/Users/samva/OneDrive/Bureaublad/Studiejaar␣5␣2023-2024/AM3000␣Bachelorproject/Simulation_BEP/Results_Simulations/Simulation_%d/plots/contour_plot_%d_%d_sigma.png", seed, n, r)
    ggsave(output_filename, plot = p, width = 16, height = 16)
  }
}

## Contour plot for mu (ML, GLS)
for (n in n_values){
  for (r in r_values){
    p <- contour_plot(n, r, c("mu", "ML"), c("mu", "GLS"))
    print(p)

    ## Saving the contour plots individually
    output_filename <- sprintf("C:/Users/samva/OneDrive/Bureaublad/Studiejaar␣5␣2023-2024/AM3000␣Bachelorproject/Simulation_BEP/Results_Simulations/Simulation_%d/plots/contour_plot_%d_%d_mu.png", seed, n, r)
    ggsave(output_filename, plot = p, width = 16, height = 16)
  }
}

## Combined contour plot
for (n in 75){
  for (r in c(5,10,15)){
    p <- contour_plot(n , r, c("sigma","ML"), c("mu", "ML"), c("sigma","GLS"), c("mu", "GLS")) + xlim(0,2.5) + ylim (-2.5,2.75)
    print(p)

    ## Saving the contour plots individually
    output_filename <- sprintf("C:/Users/samva/OneDrive/Bureaublad/Studiejaar␣5␣2023-2024/AM3000␣Bachelorproject/Simulation_BEP/Results_Simulations/Simulation_%d/plots/combined_contourplot_%d_%d_mu.png", seed, n, r)
    ggsave(output_filename, plot = p, width = 16, height = 16)
  }
}


#---------------------PREDICTING NEXT FAILURES--------------------------------

# Constructing initial dataframe for quantiles next failure
next_failure_quantiles = censored_df %>% filter(estimator %in% c("ML", "GLS"))
colnames(next_failure_quantiles) = c("t_r", "n", "r", "estimator", "mu", "sigma")

# Initialising prob. values for quantiles
p_values = c(0.05, 0.5, 0.95)

# Computing the p-th quantile of the next (r+1) failure
for (p in p_values){
```

```
  print(paste("p␣=", p))
  start_iter <- proc.time()
  col_name <- paste("t_", p, sep ="")

  t_r <- next_failure_quantiles$t_r
  n <- next_failure_quantiles$n
  r <- next_failure_quantiles$r
  mu <- next_failure_quantiles$mu
  sigma <- next_failure_quantiles$sigma

  next_failure_quantiles[col_name] <- q_next_failure(t_r, n, r, p, mu, sigma)

  # # Saving next_failur_quantiles file
  # filename <- sprintf("C:/Users/samva/OneDrive/Bureaublad/Studiejaar 5 2023-2024/AM3000 Bachelorproject/Simulation_BEP/Results_Simulations/Simulation_%d/next_failure_quantiles_p%f.csv", seed, p)
  # write.csv(next_failure_quantiles[col_name], filename, row.names = FALSE)

  print(paste("Running␣time:",signif((proc.time()-start_iter)[1],3), "seconds"))
}

# Computing the true p-th quantile of the next (r+1) failure
for (p in p_values){
  print(paste("True␣p␣=", p))
  start_iter <- proc.time()
  col_name <- paste("t_true_", p, sep ="")

  t_r <- next_failure_quantiles$t_r
  n <- next_failure_quantiles$n
  r <- next_failure_quantiles$r
  mu <- 0
  sigma <- 1

  next_failure_quantiles[col_name] <- q_next_failure(t_r, n, r, p, mu, sigma)

  # # Saving next_failur_quantiles file
  # filename <- sprintf("C:/Users/samva/OneDrive/Bureaublad/Studiejaar 5 2023-2024/AM3000 Bachelorproject/Simulation_BEP/Results_Simulations/Simulation_%d/next_failure_quantiles_true_p%f.csv", seed, p)
  # write.csv(next_failure_quantiles[col_name], filename, row.names = FALSE)

  print(paste("Running␣time:",signif((proc.time()-start_iter)[1],3), "seconds"))
}


print("Saving␣the␣all␣future␣failure␣quantiles")
start_iter <- proc.time()
# Saving next_failure_quantiles file
filename <- sprintf("C:/Users/samva/OneDrive/Bureaublad/Studiejaar␣5␣2023-2024/AM3000␣Bachelorproject/Simulation_BEP/Results_Simulations/Simulation_%d/next_failure_quantiles.csv", seed)
write.csv(next_failure_quantiles, filename, row.names = FALSE)
print(paste("Running␣time:",signif((proc.time()-start_iter)[1],3), "seconds"))



#------------------------------PROCESSING NEXT FAILURE QUANTILES---------------

# Loading next_failure_quantiles file
next_failure_quantiles <- read_csv(sprintf("C:/Users/samva/OneDrive/Bureaublad/Studiejaar␣5␣2023-2024/AM3000␣Bachelorproject/Simulation_BEP/Results_Simulations/Simulation_%d/next_failure_quantiles.csv", seed))

# Calculate the prediction mean square error
prediction_errors <- next_failure_quantiles %>%
  group_by(n, r, estimator) %>%
  summarise(
    mse_0.05 = mean((t_0.05-t_true_0.05)^2),
    mse_0.5 = mean((t_0.5-t_true_0.5)^2),
    mse_0.95 = mean((t_0.95-t_true_0.95)^2),
    rmse.t_0.05 = sqrt(mean((t_0.05-t_true_0.05)^2)),
```

```
    rmse.t_0.5 = sqrt(mean((t_0.5-t_true_0.5)^2)),
    rmse.t_0.95 = sqrt(mean((t_0.95-t_true_0.95)^2))
  )

metrics <- c("rmse.t_0.05", "rmse.t_0.5", "rmse.t_0.95")

for (metric in c("rmse.t_0.95")){
  # Create a list of plots of one performance metric
  plots_rmse <- lapply(n_values, function(n) plot_performance_metric(n = n, metric = metric, data = prediction_errors) + ylim(0,0.11))
  # Arrange plots in a grid
  rmse_plot_grid <- do.call(grid.arrange, c(plots_rmse, nrow = 2))
  # Display the figure
  rmse_plot_grid

  # Save the complete grid to a PNG file
  output_filename <- sprintf("C:/Users/samva/OneDrive/Bureaublad/Studiejaar␣5␣2023-2024/AM3000␣Bachelorproject/Simulation_BEP/Results_Simulations/Simulation_%d/plots/plot_prediction_%s.png", seed, metric)
  ggsave(output_filename, plot = rmse_plot_grid, width = 16, height = 12)
}

# Filtering prediction error
pred_error_ML  = prediction_errors %>% filter(estimator == !!"ML")
pred_error_GLS = prediction_errors %>% filter(estimator == !!"GLS")

# Create Relative Efficiency Dataframe
RE_pred = data.frame(n = pred_error_ML$n, r = pred_error_ML$r,
                RE_0.05 = pred_error_GLS$mse_0.05/pred_error_ML$mse_0.05,
                RE_0.5 = pred_error_GLS$mse_0.5/pred_error_ML$mse_0.5,
                RE_0.95 = pred_error_GLS$mse_0.95/pred_error_ML$mse_0.95)


## Plotting RE_pred in different graphs for different values of n
for (param_col in c("RE_0.05")){
  # Create a list of plots of one performance metric
  RE_plots <- lapply(n_values, function(n) RE_plot(n = n, param_col, RE = RE_pred))
  # Arrange plots in a grid
  RE_plot_grid <- do.call(grid.arrange, c(RE_plots, nrow = 2))
  # Display the figure
  RE_plot_grid

  # # Save the complete grid to a PNG file
  # output_filename <- sprintf("C:/Users/samva/OneDrive/Bureaublad/Studiejaar 5 2023-2024/AM3000 Bachelorproject/Simulation_BEP/Results_Simulations/Simulation_%d/plots/RE_prediction_%s.png", seed, param_col)
  # ggsave(output_filename, plot = rmse_plot_grid, width = 16, height = 12)
}

## Plotting RE_pred in the same graph for different values of n
for (param_col in c("RE_0.05")){
  RE_plots <- ggplot() +
    geom_hline(yintercept = 1, linetype = "dotted", color = "black", linewidth = 1) +
    labs(
      title = paste("Relative␣Efficiencies␣for␣t_0.05"),
      x = "Observed␣failures␣(r)",
      y = "␣",
      color = "n"  # Label for the legend
    ) +
    theme_classic(base_size = 20) +
    ylim(0.9, 2.45)

  # Assume n_values is defined
  colors <- c("blue", "red", "green", "purple")

  for (i in 1:length(n_values)){
    n <- n_values[i]
    scatter_elements <- RE_scatter(n, param_col, RE_pred)
    RE_plots <- RE_plots + scatter_elements[[1]] #+ scatter_elements_mu[[2]]
  }
  print(RE_plots)
```

```

  # Save the complete grid to a PNG file
  output_filename <- sprintf("C:/Users/samva/OneDrive/Bureaublad/Studiejaar␣5␣2023-2024/AM3000␣Bachelorproject/Simulation_BEP/Results_Simulations/Simulation_%d/plots/RE_prediction_%s.png", seed, param_col)
  ggsave(output_filename, plot = RE_plots, width = 16, height = 12)
}
```