Lower quantile estimation within an artificially censored framework

by
Jarod Smith

14016665

Submitted in partial fulfillment of the requirements for the degree

Magister Scientiae
Mathematical Statistics

In the Faculty of Natural & Agricultural Sciences
Department of Statistics
University of Pretoria
Pretoria

10th February 2020

Lower quantile estimation within an artificially censored framework

by

Jarod Mark Smith
E-mail: jarodsmith706@gmail.com

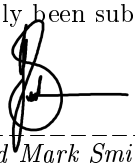
Abstract

Quantile estimation is a vital aspect of statistical analyses in a variety of fields. For example, lower quantile estimation is crucial to ensure the safety and reliability of wood-built structures. Various statistical techniques, which include parametric, non-parametric and mixture modelling are available for estimation of lower quantiles. An intuitive approach would be to consider models that fit the tail of the sample instead of the entire range. Quantiles of interest can be estimated by artificially censoring observations beyond a chosen threshold. The choice of threshold is crucial to ensure efficient and unbiased quantile estimates, and usually the 10^{th} empirical percentile is chosen as the threshold. [16] proposes a bootstrap approach in order to obtain a better threshold for the censored Weibull MLE, however, this approach is computationally expensive. A new threshold selection technique is proposed that makes use of a standardised-weighted adjusted truncated Kolmogorov-Smirnov test (SWAKS-MLE). The SWAKS-MLE outperforms in the bootstrap threshold censored Weibull MLE method, in addition to being vastly less computationally intensive.

Keywords: Adjusted Kolmogorov-Smirnov threshold selection technique, Artificial censoring, Bootstrap, Lower quantile, Semi-parametric.

Declaration

I, *Jarod Mark Smith,* declare that this mini-dissertation (100 credits), which I hereby submit for the degree Magister Scientiae in Mathematical Statistics at the University of Pretoria, is my own work and has not previously been submitted by me for a degree at this or any other tertiary institution.



_____
Jarod Mark Smith

_____
Dr. J.T. Ferreira

_____
Prof. A. Bekker

2020/02/10
_____
Date

Acknowledgements

Without the support of the following list of contributors, this work wouldn't have been possible at all:

- To Micaela, thank you for your unwavering support and encouragement. Without your love and patience, this work may not have been a possibility.
- Dr. Jaco Visagie, thank you for your friendship.
- Prof. A. Bekker, thank you for the inspiration, guidance and encouragement over the past two years. Thank you for believing in me and granting me the opportunities that will forever be invaluable. You are more than just a supervisor, it is truly a privilege working with you.
- Dr J.T. Ferreira, I would like to extend my appreciation and gratitude for your tireless contributions, remarks and suggestions.
- Prof. M. Arashi, thank you for your valuable input, especially concerning the Kolmogorov-Smirnov adjustments.
- To my parents, thank you for believing me in every step of this journey.
- Yang Liu, kudos on the brilliant, well structured and insightful dissertation: "Lower Quantile Estimation of Wood Strength Data".
- The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Extreme tail quantile estimation is very important in material strength studies (for example, lumber strength studies [16]). Wood or lumber products are widely used in the building and construction of roofs, walls and floors. To ensure the safety and reliability (duration) of wood built structures it is necessary to study the strength properties of the wood products that are used to build these structures. A more in depth description of wood strength measures can be found in [4], among those is the *Modulus of Rupture* strength (MOR) or the bending strength of wood. The MOR strength measure is particularly important because it measures the amount of force that can be applied vertically to the grain of the wood before any structural failure occurs. The pressure that a wooden board can handle is the most common found on wooden rooftops, floors and even wooden bridges and walkways [16]. In order to measure the MOR of a wooden board implies a destructive process, therefore the amount of data available for such a study will naturally be limited.

Due to the plethora of tree types - within and between species - the MORs will differ. In other words, two pieces of wood from the same tree may have different MORs and the MOR of a wooden board may therefore be treated as random [16]. This random behavior allows the use of statistical methods, such as extreme tail quantile estimation, to estimate extreme lower tail quantiles. Due to the stochastic behaviour of strength properties of wood, the probability of failure can only be gauged to be within a very small interval or percentage. For example, if the probability of failure should be less than 5%, then the pressure load on the wooden boards should not exceed the 5^{th} percentile [16]. This dissertation will focus primarily on estimating the 5^{th} percentile.

The main contribution of this study is to introduce a novel, data-driven, threshold selection technique for lower quantile estimation. That being said, a large proportion of the work presented in [16] will be revisited. In particular, Chapters 2, 3, 4 and 5 of [16]will be thoroughly re-worked in this study. The key ideas and concepts presented in these chapters are necessary for the insight and intuition required for new threshold selection approach presented in Chapter 4 of this study.

To estimate quantiles of interest, engineers will typically randomly sample wooden boards (for example, 300) from a wood mill and proceed to test the MOR strength of each board through a destructive process. Two datasets for this study, namely MOR1 and MOR2 contain 98 and 282 observations respectively are re-used from [16]. The MOR datasets will be used in this dissertation to illustrate statistical principles concerned with lower quantile estimation. The histograms of each dataset are provided in Figure 1.0.1.

1.1 Motivation

Although quantile estimation is not a particularly new concept in statistical theory and application, very little literature exists on the computation of extreme lower quantile estimation within a setting of limited sample data (small sample sizes). The study of statistical extremes is usually dealt within *extreme value theory*

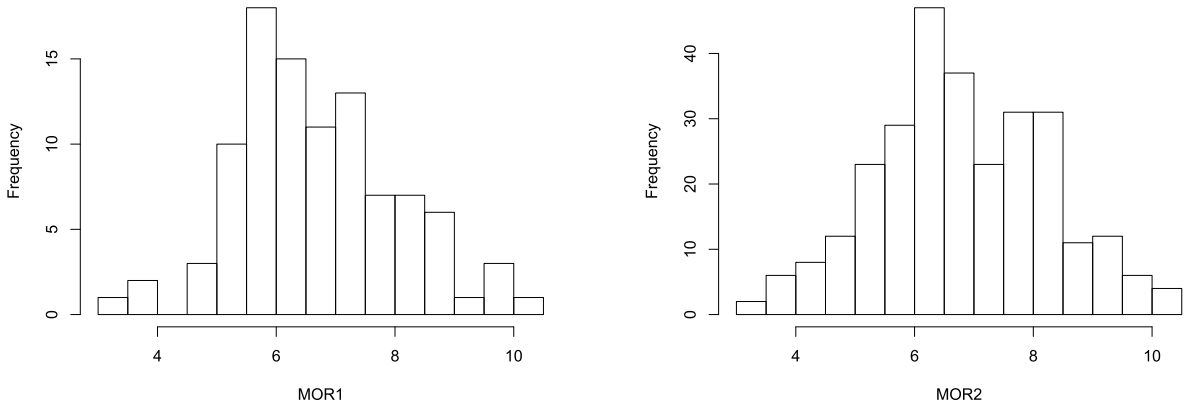


Figure 1.0.1: Histogram of MOR1 and MOR2 dataset.

(EVT) [7], for example, the *peaks-over-threshold* (POT) approach models the excess over a suitably high threshold using a *generalised Pareto distribution* (GPD). The work presented (novel and revisited concepts) in this dissertation focuses on the left tail of the distribution, i.e. estimating a lower quantile. The approach, originally presented in [16], investigates models that fit the left tail of the data, rather than the entire range. Thus if there is a desire to estimate the lower quantile, all observations to the right of a suitably chosen threshold are *artificially censored*. This allows the investigator to focus on the fit of the left tail and artificially censor the rest of the observations. This is not a new concept and has been presented and studied rigorously in [16]. What remains is to investigate a more computationally viable threshold selection technique for the estimation of the extreme lower quantiles.

1.2 Objectives

- revisit the work on lower quantile estimation presented in [16] to introduce and re-work the following fundamental concepts:
  - The statistical process of lower quantile estimation, including the notion of the quality pertaining to the estimated lower quantile.
  - Classical statistical techniques used for lower quantile estimation. In particular, parametric and non-parametric approaches to lower quantile estimation.
  - Explore the idea of a censored-adjusted semi-parametric approach to lower quantile estimation using methods focused on censored data.
  - Thoroughly study the mixture model approach to lower quantile estimation.
  - revisit the novel data-driven statistical techniques for optimum threshold selection for lower quantile estimation by making use of bootstrapping.
- Introduce a novel, data-driven, threshold selection technique for lower quantile estimation using an adjusted Kolmogorov-Smirnov statistic.

1.3 Contributions

- This dissertation is based largely on the superb 'Lower Quantile Estimation of Wood Strength Data' presented in [16]. That being said, a new threshold selection technique is proposed that is based on

a *standardised-weighted adjusted Kolmogorov-Smirnov* test. The new test outperforms in the *bootstrap threshold selection* technique presented in [16], in addition to being vastly less computationally intensive.

1.4 Dissertation outline

- In Chapter 2 the criteria used to measure the performance of a quantile estimate are introduced. More so, the use of typical non-parametric and parametric quantile estimating techniques are investigated. Thereafter, limitations and potential drawbacks of non-parametric and parametric quantile estimating techniques are discussed.
- Chapter 3 introduces the notion of a flexible censored-adjusted semi-parametric model for lower quantile estimation. Furthermore, the concept of artificial censoring is discussed. Thereafter, the semi-parametric model is fitted to the tail of a dataset using *censored maximum likelihood estimation*. Moreover, a comparison of various parametric, non-parametric and semi-parametric quantile estimation techniques is explored. An introduction of the censored and uncensored *Weibull mixture distribution* is presented. The *expectation-maximisation* (EM) algorithm for parameter estimation for the censored and uncensored Weibull mixture models is presented. Thereafter, a simulation comparison of the censored and uncensored Weibull mixture distribution's quantile estimates is given.
- Chapter 4 is devoted to the data-driven threshold selection techniques for lower quantile estimation. That is, an introduction to the *bootstrap threshold censored Weibull MLE* (B-MLE) is presented. Followed by the estimation of MSE using a bootstrap approach. Thereafter, the standardised-weighted adjusted Kolmogorov-Smirnov test statistic (SWAKS) is proposed and studied. Finally, a simulation comparative study of the data-driven threshold selection techniques is presented.
- Chapter 5 presents conclusive remarks and a summary on the material covered in this dissertation.

1.5 Simulation settings

There are several simulation studies in [16] that are re-worked and presented in this dissertation. The simulation settings remain constant throughout the study and are presented here for ease of readability later. Given that a large portion of the work covered in [16] is presented and compared to, it is prudent to use the same simulation settings present in that study. That is:

- The number of *Monte-Carlo repetitions* is fixed at $N = 10000$ (unless specified otherwise).
- The number of *bootstrap repetitions* is set to $B = 5000$ (unless specified otherwise).
- The parametric distributions used to imitate the real MOR datasets are the Weibull, Log-normal, Gamma, Minimum Gumbel, two-component normal mixture, two-component log-normal mixture and two-component Weibull mixture.
- The sample size of the simulated data is fixed at $n = 300$.
- The quantile estimate of interest is the 5^{th} quantile.

1.6 List of acronyms and notation

The acronyms and notation provided below are used frequently throughout this dissertation and are included here for ease of reference.

- **EVT**: Extreme value theory.
- **MOR**: Modulus of Rupture.

- **GDP**: Generalised Pareto distribution.
- **MLE**: Maximum likelihood estimation/estimates.
- **CW-MLE**: Censored Weibull MLE.
- **MIX**: Uncensored Weibull mixture.
- **MIX7**: Censored Weibull mixture using the 70^{th} percentile as the censoring threshold.
- **B-MLE**: Bootstrap threshold censored Weibull MLE.
- **KS**: Kolmogorov-Smirnov test.
- **SWAKS**: Standardised-weighted adjusted Kolmogorov-Smirnov test or Standardised-weighted log-adjusted Kolmogorov-Smirnov test (used interchangeably).
- **SWAKS-MLE**: SWAKS threshold censored Weibull MLE.
- **EMP**: Empirical quantile estimate using the Type 9 definition.
- **PDF**: Probability density function.
- **CDF**: Cumulative distribution function.
- **ECDF**: Empirical cumulative distribution function.
- **KDE**: Kernel density estimation.
- **KDe**: Kernel density estimate.
- **MSE**: Mean square error.
- **RMSE**: Root mean square error.
- **AIC**: Akaike information criterion.
- **BIC**: Bayesian information criterion.
- $D(\cdot)$: Kolmogorov-Smirnov statistic.
- $D^{TRUNC}(\cdot)$: Truncated Kolmogorov-Smirnov statistic.
- $D^{AKS}(\cdot)$: Adjusted truncated Kolmogorov-Smirnov statistic.
- $D^{WAKS}(\cdot)$: Weight adjusted truncated Kolmogorov-Smirnov statistic.
- $D^{SWAKS}(\cdot)$: Standardised-weighted adjusted truncated Kolmogorov-Smirnov statistic.
- $L(\cdot)$: Likelihood function.
- $L_I(\cdot)$: Type I censoring likelihood.
- $L_{II}(\cdot)$: Type II censoring likelihood.
- $l(\cdot)$: Log likelihood function.
- $l^c(\cdot)$: Complete log-likelihood.
- $l^I(\cdot)$: Incomplete log-likelihood.
- $\mathcal{L}(\cdot)$: Lagrangian multiplier.

- $-2log\,(\lambda)$: Log likelihood ratio statistic.
- $\rightarrow$: Converges
- $\overset{asymptotically}{\rightarrow}$: Converges asymptotically.
- $\overset{d}{\rightarrow}$: Converges in distribution.
- $\underline{\theta}$: Vector of true population parameters.
- $\hat{\underline{\theta}}$, $\tilde{\underline{\theta}}$: Vector of parameter estimates.
- $\underline{\theta}^{(r)}$: Parameter estimates after the r^{th} EM algorithm iteration.
- π_i : Mixture probabilities.
- q : True population quantile.
- $\widetilde{q}$: Sample quantile estimate.
- $\bar{q}$: KDe quantile estimate.
- $I(\cdot)$: Indicator function.
- C : Censoring threshold.
- $F(\cdot)$, $G(\cdot)$, $H(\cdot)$: General population cumulative distribution function.
- $\hat{F}(\cdot)$: Empirical cumulative distribution function.
- $\tilde{F}(\cdot)$: Parametric cumulative distribution function.
- $Q(\cdot)$: Expected value of the complete log-likelihood function of $\underline{\theta}$.
- b : Bandwidth parameter.
- $k(\cdot)$: Kernel function.
- $\tilde{k}_b(\cdot)$: Kernel density estimate.
- $\tilde{K}_b(\cdot)$: Kernel density cumulative distribution function.

Chapter 2

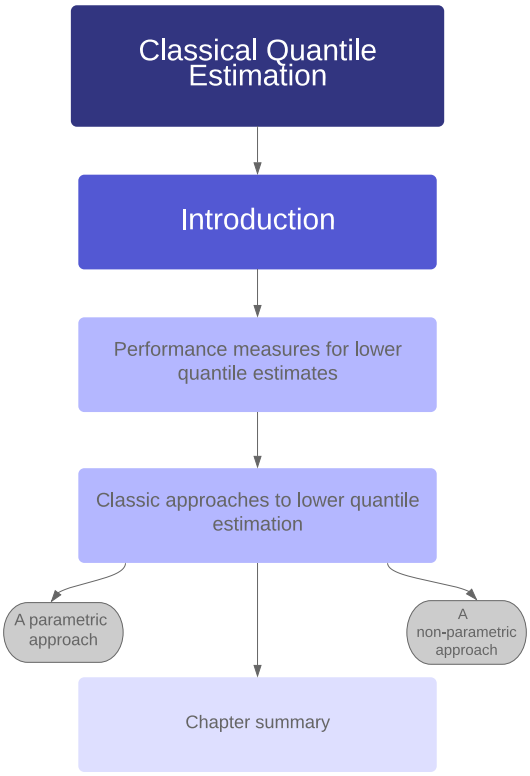
Classical Quantile Estimation

Chapter highlights

The highlights for Chapter 2 include:

- Introducing criteria that are used to measure the performance of a quantile estimate.
- The use of typical non-parametric and parametric quantile estimation techniques.
- Limitations and potential drawbacks of non-parametric and parametric quantile estimation techniques.

Chapter outline



2.1 Introduction

This chapter revisits a few core concepts and methodologies presented in Chapters 1 and 2 in [16]. It is crucial to make use of a statistically sound approach to measure the performance of lower quantile estimates. The *mean square error* (MSE) and *root mean square error* (RMSE) are introduced in Section 2.2 to study this qualitative behaviour of the lower quantile estimates. Furthermore, this chapter revisits and discusses some advantages and limitations for both parametric (Subsection 2.3.1) and non-parametric (Subsection 2.3.2) techniques that can be used for quantile estimation. In the parametric approach, four distributions are initially used in an attempt to approximate the true distribution of the MOR datasets seen in Chapter 1, namely the Weibull, log-normal, minimum Gumbel and gamma distribution, from which the parametric quantile estimates can be obtained. For the non-parametric case, the focus is mainly on *empirical quantile estimates*, as well as quantile estimates obtainable from a *kernel density estimate*.

2.2 Performance measures for lower quantile estimates

Before introducing the various techniques and methodologies used to obtain quantile estimates, it is necessary to discuss how to evaluate the performance and quality of a quantile estimate. The *mean square error* (MSE) will be used to evaluate the quality of the quantile estimate, since the statistical measure encompasses both the bias and variance [2]:

$$E_H\left(\widetilde{q}_n - q\right)^2 = Var\left(\widetilde{q}_n\right) + \left[E_H\left(\widetilde{q}_n\right) - q\right]^2, \tag{2.2.1}$$

where $\widetilde{q}_n$ is the quantile estimate from a sample of size n and q is the true quantile under the model H of the population. In order to thoroughly revisit the work presented in [16], the *root mean square error* (RMSE) will also be considered in the simulation results presented in Chapters 3 and 4. The MSE and RMSE of the quantile estimate $\widetilde{q}_n$ under the population model will be estimated as follows:

$$\widehat{MSE} = \bar{d} = \frac{1}{N}\sum_{i=1}^{N} d_i \tag{2.2.2}$$

$$\widehat{RMSE} = \sqrt{\widehat{MSE}} = \sqrt{\bar{d}}, \tag{2.2.3}$$

where $d_i = \left(\widetilde{q}_n^i - q\right)^2$ and the superscript indicates the index of the quantile and the subscript the sample size. The bias (referred to as the **accuracy** hereafter) and the standard error of the quantile estimator (also referred to as the **efficiency** hereafter) will be included in this dissertation to study the accuracy and efficiency of the quantile estimates.

Furthermore, the efficiency of the $\widehat{MSE}$ and $\widehat{RMSE}$ can be evaluated by studying the Monte Carlo error of these estimates and the interested reader is referred to Chapter 1 in [16] for a detailed explanation.

2.3 Classic approaches to lower quantile estimation

In this section, the work covered in Chapter 2 in [16] is revisited. In particular a parametric and non-parametric approach to lower quantile estimation is discussed.

2.3.1 A parametric approach

A parametric approach to lower quantile estimation is initially explored to accommodate the natural flow that most readers are typically accustomed to.

Following from Section 2.2, the performance of quantile estimates can be gauged by utilising various *probability density functions* (PDF) in an attempt to imitate the MOR datasets presented in Chapter 1. It

is therefore natural to explore parametric approximations of the true distribution of the data in hope of obtaining more efficient quantile estimates [22].

According to [16] the Weibull, Gamma, Log-Normal and Minimum Gumbel distribution are frequently used in modeling the strength of materials and are presented below.

2.3.1.1 Weibull distribution

The two-parameter Weibull distribution, [24] with PDF is given by

$$f(x, \beta, \eta) = \frac{\beta}{\eta}\left(\frac{x}{\eta}\right)^{\beta-1} exp\left(-\left(\frac{x}{\eta}\right)^{\beta}\right), \quad x \geq 0, \beta \geq 0, \eta \geq 0 \tag{2.3.1}$$

and the *cumulative distribution function* (CDF) is

$$F(x, \beta, \eta) = 1 - exp\left(-\left(\frac{x}{\eta}\right)^{\beta}\right), \quad x \geq 0, \beta \geq 0, \eta \geq 0. \tag{2.3.2}$$

The history and introduction of the Weibull distribution can be found in [14] and [19]. The Weibull distribution is a popular choice for survival analysis [19] and the American Society for Testing and Materials [1], make use of the Weibull distribution for testing strength properties of materials.

2.3.1.2 Gamma distribution

The Gamma distribution is a fundamental distribution in statistics [12] and is closely related to many other distributions including the exponential, Erlang, chi-square, Nakagami-m and generalised gamma [23], to name a few. The gamma distribution is also a popular distribution in survival analysis and the PDF is given by

$$f(x, \theta, \kappa) = \frac{x^{\theta-1}}{\Gamma(\theta)\kappa^{\theta}} exp\left(\frac{x}{\kappa}\right), \quad x \geq 0, \theta > 0, \kappa > 0, \tag{2.3.3}$$

and the CDF is

$$F(x, \theta, \kappa) = \frac{\gamma\left(\theta, \frac{x}{\kappa}\right)}{\Gamma(\theta)}, \quad x \geq 0, \theta > 0, \kappa > 0. \tag{2.3.4}$$

where $\Gamma(\theta) = \int_0^{\infty} t^{\theta-1}e^{-t}dt$ and $\gamma\left(\theta, \frac{x}{\kappa}\right)$ denote the gamma and lower incomplete gamma function respectively.

2.3.1.3 Log-normal distribution

The log-normal distribution is the exponential transformation of a random variable that is normally distributed [12], if $X \sim Lognormal(\mu, \sigma)$ then $log(X) \sim Normal(\mu, \sigma)$. The log-normal distribution is used in survival analysis as well as in EVT [5]. The PDF of the log-normal distribution is given by

$$f(x, \mu, \sigma) = \frac{1}{x\sigma\sqrt{2\pi}} exp\left(-\frac{(log(x) - \mu)^2}{2\sigma^2}\right), \; x \geq 0, \mu \in (-\infty, \infty), \sigma > 0, \tag{2.3.5}$$

and the CDF is

$$F(x, \mu, \sigma) = \Phi\left(\frac{log(x) - \mu}{\sigma}\right), \; x \geq 0, \mu \in (-\infty, \infty), \sigma > 0, \tag{2.3.6}$$

where $\Phi(.)$ is the CDF of a standard normal distribution.

2.3.1.4 Minimum Gumbel distribution

The Gumbel distribution was introduced by [9] and [8] and is well rooted in EVT, since it corresponds to the *Type I extreme value distribution* ($\xi = 0$). In the simulations for this study the Minimum Gumbel distribution will be considered since the focus is on lower quantile estimation. The PDF of the Minimum Gumbel distribution is given by

$$f(x;\, a,\, b) = \frac{1}{b} exp\left(\frac{x-a}{b} - exp\left(\frac{x-a}{b}\right)\right),\ x \in \Re,\ a \in (-\infty,\, \infty),\ b > 0 \tag{2.3.7}$$

and the CDF is

$$F(x;\, a,\, b) = 1 - exp\left(-exp\left(\frac{x-a}{b}\right)\right),\ x \in \Re,\ a \in (-\infty,\, \infty),\ b > 0.$$

The simplistic and pleasant notation for the minimum Gumbel distribution is adopted from [16] to enhance the quality of the reading experience.

These four distributions are fitted to the real MOR datasets to identify which models fit the data most accurately, more importantly the left tail, given that the interest here is lower quantile estimation. The parameters for the aforementioned distributions are fitted via *maximum likelihood estimation* (MLE) and have been conveniently summarised in Tables 2.3.1 and 2.3.2.

Model	Parameters		$D\left(\tilde{F}_n - \hat{F}_n\right)$	*AIC*	$\hat{q}_{0.05}$
Weibull	$\hat{\beta} = 5.19$	$\hat{\eta} = 7.26$	0.082	350.18	4.10
Log-normal	$\hat{\mu} = 1.881$	$\hat{\sigma} = 0.212$	0.060	347.29	4.62
Gamma	$\hat{\theta} = 22.99$	$\hat{\kappa} = 0.291$	0.053	344.98	4.58
Minimum Gumbel	$\hat{\alpha} = 7.412$	$\hat{b} = 1.422$	0.106	366.41	3.19

Table 2.3.1: Goodness-of-fit and quantile estimates for parametric models for MOR1.

Model	Parameters		$D\left(\tilde{F}_n - \hat{F}_n\right)$	*AIC*	$\hat{q}_{0.05}$
Weibull	$\hat{\beta} = 5.23$	$\hat{\eta} = 7.39$	0.071	1014.71	4.19
Log-normal	$\hat{\mu} = 1.89$	$\hat{\sigma} = 0.22$	0.049	1021.15	4.63
Gamma	$\hat{\theta} = 21.46$	$\hat{\kappa} = 0.22$	0.046	1012.95	4.59
Minimum Gumbel	$\hat{\alpha} = 7.53$	$\hat{b} = 1.41$	0.1	1050.36	3.36

Table 2.3.2: Goodness-of-fit and quantile estimates for parametric models for MOR2.

Figure 2.3.1 illustrates the MLE fitted curves for both MOR1 and MOR2 datasets. It is clear that the fit of these models are different, however visually it is difficult to justify which performs the best. In order to compare the models, the goodness-of-fit achieved by each is compared by making use of the *Akaike information criterion* (AIC) and the *Kolmogorov-Smirnov statistic* (KS):

$$D\left(\tilde{F}_n - \hat{F}_n\right) = \sup_x \left\{\left|\tilde{F}_n(x) - \hat{F}_n(x)\right|\right\}.$$

$\hat{F}_n(x)$ is the ECDF and $\tilde{F}_n(x)$ is the parametric CDF. The KS statistic measures the absolute distance between the ECDF and the parametric CDF of interest. A similar notation to [16] is adopted here due to its simplistic reading style. From a goodness-of-fit point of view summarised in Tables 2.3.1 and 2.3.2, it is quite challenging to select a clear 'winner' for either MOR1 and MOR2. Goodness-of-fit is not of much use

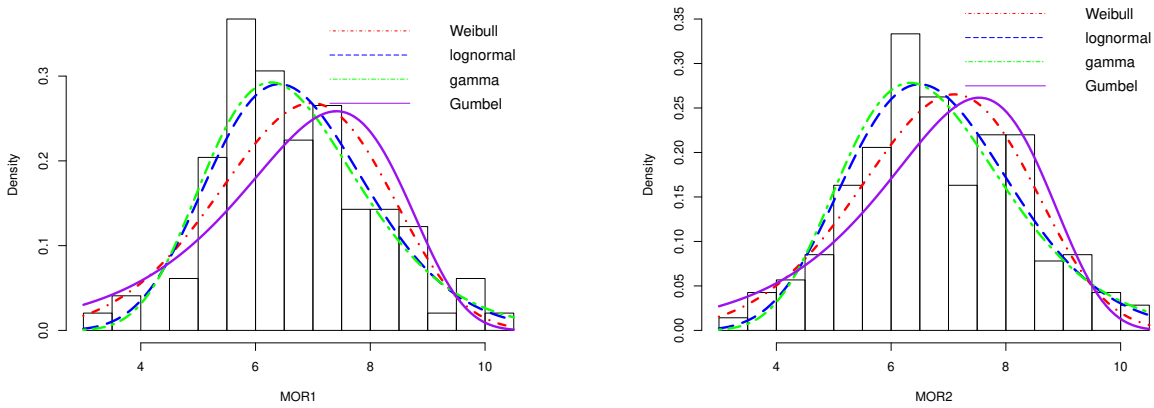


Figure 2.3.1: Various fitted parametric models for MOR1 and MOR2 datasets.

in this case and the quantile estimates for both datasets vary considerably. Choosing a parametric PDF can be challenging and may have adverse effects in practice. For example, an increased bias for estimators from miss-specified models [16, 17].

2.3.2 A non-parametric approach

As mentioned in Subsection 2.3.1, the work covered in this section is revisits Chapter 2 in [16]. More so, a few concepts are studied and elaborated in more depth.

2.3.2.1 An empirical approach to lower quantile estimation

The *empirical cumulative distribution function* (ECDF), $\hat{F}_n$, for an independent and identically distributed (*i.i.d.*) sample $X_1, \cdots, X_n$ is given by [2]:

$$\hat{F}_n(x) = \frac{1}{n}\sum_{i=1}^{n} I\{X_i \leq x\}, \tag{2.3.8}$$

where $I(\cdot)$ is an *indicator function*. The value of the indicator function is 1 when $X_i \leq x$ and 0 otherwise. The quantile of a distribution is defined as [10]:

$$Q(p) = F^{-1}(p) = \inf\{x : F(x) \geq p\}, \quad 0 < p < 1, \tag{2.3.9}$$

where F is the true CDF approximated by $\hat{F}_n(x)$. It follows that the empirical quantile estimate of interest is given by:

$$\hat{Q}(p) = \hat{F}_n(p) = \inf\left\{x : \hat{F}_n(x) \geq p\right\}, \quad 0 < p < 1.$$

There are various definitions of the empirical quantiles that are obtainable from different functions of the inverse of the ECDF. Statistical software packages usually contain nine common definitions of sample quantile estimates. For example, the "quantile" function in the *R* software package provides the nine variations on how to calculate the empirical quantile. Three of these definitions are based on rounding and six on linear interpolation. In *R*, all sample quantiles are defined as weighted averages of consecutive *order statistics*. The sample quantile for definition i (Type i) is given by:

$$\hat{Q}_{Type\ i}(p) = (1-\gamma)\, x_j + \gamma x_{j+1}, \tag{2.3.10}$$

where $\frac{j-m}{n} \leq p < \frac{j-m+1}{n}$, $x_{(j)}$ is the j^{th} order statistic, n is the sample size, γ is a function of $j = \lfloor pn + m \rfloor$ and $g = np + m - j$ and $m \in \Re$ is a constant determined by the sample quantile definition. Note that, $\lfloor g \rfloor$ is a floor function which denotes the largest integer no greater than g. These definitions are provided and studied in detail in [10]. This study will focus mainly on Type 3 and Type 9. The Type 3 definition is given by:

$$\hat{Q}_{Type\ 3}(p) = \begin{cases} x_j & \gamma = 0 \\ x_{j+1} & \gamma > 0 \end{cases}, \tag{2.3.11}$$

where $m = -1/2$, $\gamma = 0$ if $g = 0$ and j is even, and 1 otherwise. In other words, Type 3 provides the nearest even order statistic.

The Type 9 sample quantile is obtained by linear interpolation between the points (p_k, x_k) and its definition is given by (2.3.10), with $\gamma = g$, $m = p/4 + 3/8$, $p_k = \frac{k-3/8}{n+1/4}$. The Type 3 definition only uses one order statistic to compute the sample quantile, which has been questioned due to the inefficient use sample information and may lead to biased estimates. As a result of this drawback, alternative definitions involving a linear interpolation between two order statistics are preferred, for example the Type 9 definition.

Simulation study to determine the most suitable empirical quantile estimate

The choice of the above quantile definitions were not chosen at random. Several simulations, under various distributional models, were performed in order to choose the most suitable definition for this study. The simulation results for the 5^{th} quantile are provided in Figure 2.3.2 and Table 2.3.4. The PDFs that were used in the simulation study are provided in Table 2.3.3. The choice of parameters for these PDFs will be explored in Chapter 3. The simulation procedure is summarised in Algorithm 2.1.

Distribution	Parameters
Weibull	$(\alpha, \eta) = (7, 7)$
Gamma	$(\theta, \kappa) = (14, 0.6)$
Log-Normal	$(u, \sigma) = (2, 0.3)$
Minimum Gumbel	$(a, b) = (7, 0.5)$

Table 2.3.3: Distributions used in quantile estimation study.

Algorithm 2.1 Simulation study to determine the most suitable empirical quantile estimate.

1. A sample of size $n = 300$ observations is randomly generated from each of the models in Table 2.3.3.
2. The 5^{th} quantile estimates are computed as $\widetilde{q}_n$ for the nine empirical quantile definitions available in R and for each model in Step 1.
3. Calculate $d_i = \left(\widetilde{q}_n^i - q\right)^2$ where q is the true quantile under each respective model.
4. This process is repeated for $N = 10000$ repetitions in order to obtain the $\widehat{RMSE}$ estimate for each empirical quantile definition and model for 5^{th} quantile.

Figure 2.3.2 displays the boxplot of the distribution of the 5^{th} empirical percentile from the simulations for the models in Table 2.3.3. The horizontal coloured lines indicate the true 5^{th} empirical percentile under each model. The Type 9 definition (based on linear interpolation) produces quantile estimates that are almost

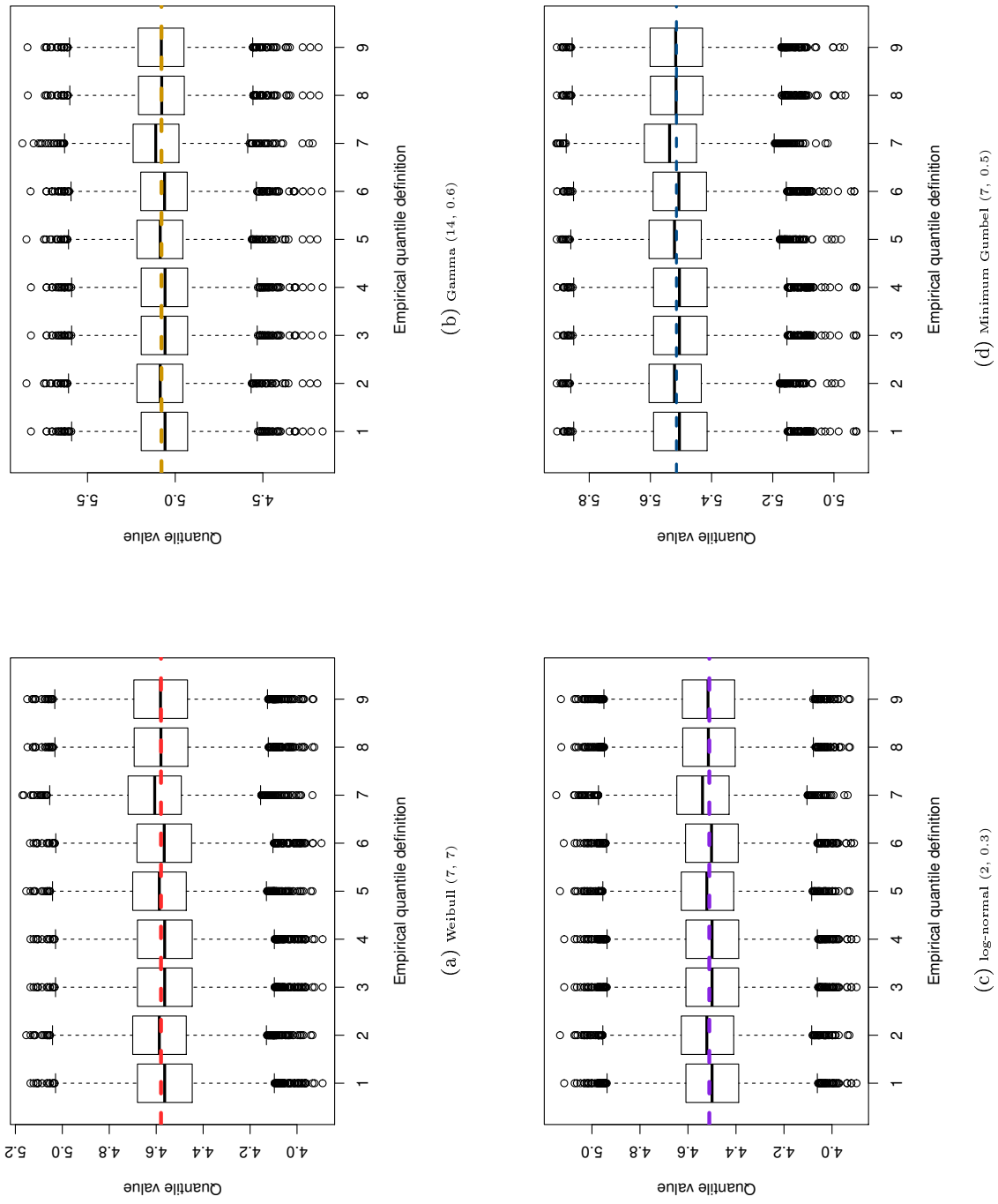


Figure 2.3.2: Boxplots of 5^{th} quantile estimate for the nine empirical quantile definitions for various models.

Type	Weibull (7, 7)	gamma (14, 0.6)	log-normal (2, 0.3)	Min Gumbel (7,0.5)
1	0.173	0.195	0.165	0.131
2	0.169	**0.192**	**0.162**	**0.127**
3	0.173	0.195	0.165	0.131
4	0.173	0.195	0.165	0.131
5	0.169	**0.192**	**0.162**	**0.127**
6	0.172	0.195	0.164	0.131
7	0.169	0.194	0.164	**0.127**
8	0.169	**0.192**	**0.162**	0.128
9	**0.168**	**0.192**	**0.162**	**0.127**

Table 2.3.4: RMSE for the 5^{th} quantile per empirical quantile definition for various models and where $n = 300$ and $N = 10000$.

unbiased (the coloured horizontal lines overlap the median almost perfectly) under each model in Table 2.3.3. Table 2.3.4 provides the RMSE for the 5^{th} quantile estimates per empirical quantile definition for each model in Table 2.3.3. From the simulations it is clear that Type 9 performs the best (has the smallest RMSE values for each model) and is, therefore, the best all round performer in terms of accuracy and efficiency. Furthermore, according to [20],

$$\sqrt{n}\left(\widetilde{q}_n - q\right) \xrightarrow{d} N\left(0, \frac{p(1-p)}{f^2(q)}\right) \tag{2.3.12}$$

where $f(\cdot)$ is the true PDF for the data under consideration. For this study, however, there are very few observations in the tail, as seen in Figure 1.0.1, and as a result $f(q)$ is likely to be very small. This implies that the variance of the quantile estimates may become very large and ultimately unreliable. For a detailed explanation readers are referred to [16].

2.3.2.2 Kernel density estimates of quantiles

Kernel density estimation (KDE) is a non-parametric approach of estimating the PDF of a random variable and is a useful statistical tool for creating smooth curves for a given dataset, in addition to visualizing the shape of the data. Moreover, KDE is the continuous analog for a discrete histogram of the data. Intuitively speaking, a kernel density estimate (KDe) is the summation of "bumps", where a "bump" is assigned to each observation. The size of the "bump" represents the probability assigned in the neighborhood of values around each observation. The more data points at each observation, the higher the amplitude of the "bump". Each "bump" is centered at the observation and symmetrically covers the observation's neighboring values. The "bump" is referred to as a kernel, a symmetric function that integrates to one. Each kernel has a bandwidth, which determines the width of the "bump" (the width of the neighborhood of values to which probability is assigned). A larger bandwidth leads to a shorter and wider "bump", in other words the kernel spreads out farther from the center and assigns more probability to the neighboring values.

The KDe can be written as [21]:

$$\tilde{k}_b\left(x\right) = \frac{1}{nb}\sum_{i=1}^{n}\kappa\left(\frac{x - X_i}{b}\right). \tag{2.3.13}$$

where $k\left(\cdot\right)$ is the kernel function. The concept of weighting the distances of the observations from a particular point, x, is given by (2.3.13).

The focus of this study is on lower quantile estimation, therefore, only the Gaussian kernel is considered. The smoothing parameter is denoted by b. The smoothing parameter is selected by making use of the "Solve-the-Equation" approach recommended by [13] and is readily available in R.

The KDe is a consistent estimator of the true PDF, that is, $\tilde{k}_b(x) \overset{asymptotically}{\rightarrow} f(x)\ \forall\ x$ if the bandwidth $b \rightarrow 0$ and $nb \rightarrow \infty$ when $n \rightarrow \infty$, see [11, 22]. The quantile estimate from the KDe, $\overline{q}$, is obtained by numerically solving the inverse of the estimated KDe CDF:

$$\tilde{K}_b(x) = \int_{-\infty}^{x} \tilde{k}_b(u)\, du. \tag{2.3.14}$$

Simulation study to determine kernel density and quantile estimates

According to the "Solve-the-Equation" approach, the bandwidth for MOR1 is 0.51648 while the bandwidth for MOR2 is 0.07232. The KDe curves are provided in Figure 2.3.3. The performance of the KDe quantile estimates for the MOR datasets will be studied in Chapter 3.

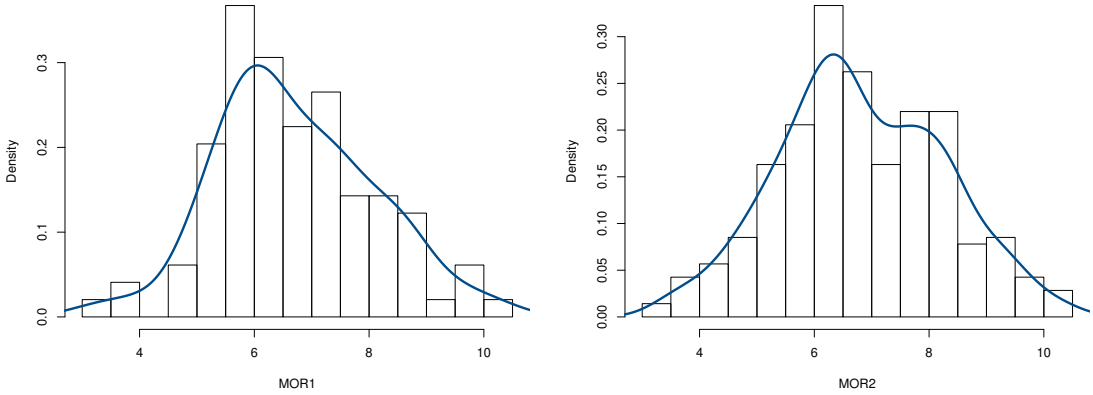


Figure 2.3.3: Fitted kernel density estimation curves for MOR1 and MOR2.

2.4 Chapter summary

Chapter 2 revisits key concepts presented in Chapters 1 and 2 of [16]. That is, this chapter introduces criteria that is used to measure the performance of a quantile estimate, as well as typical parametric and non-parametric approaches to quantile estimation. The non-parametric approach includes empirical quantile estimation which is commonly used and employed by major statistical software packages such as *R*. The idea behind the empirical quantile estimation procedure is to make use of functions that use at most two sample order statistics to estimate the quantiles of interest. Furthermore, quantile estimation using KDE is also a popular non-parametric quantile estimation technique. The quantile estimates can be made more efficient, compared to the empirical estimates, by smoothing over the entire dataset when using KDE. Alternatively, a parametric distribution can be fitted to the sample data in order to obtain quantile estimates from the fitted PDF.

Moreover, this chapter discusses a few advantages and limitations for both said techniques. Due to the nature of extreme lower quantile estimation, that is there is typically a scarcity of data available in the tail, the variance (and standard errors) of the KDe and empirical quantile estimates may become large and ultimately unreliable. In Chapter 3 it will be shown that the parametric quantile estimate, even though efficient, can be significantly biased if the incorrect model is chosen. These issues provide the motivation for a technique that addresses these limitations present in the parametric and non-parametric quantile estimates. This technique will also be discussed in detail in the next chapter.

Chapter 3

Censored and Mixture Models for Lower Quantile Estimation

Chapter highlights

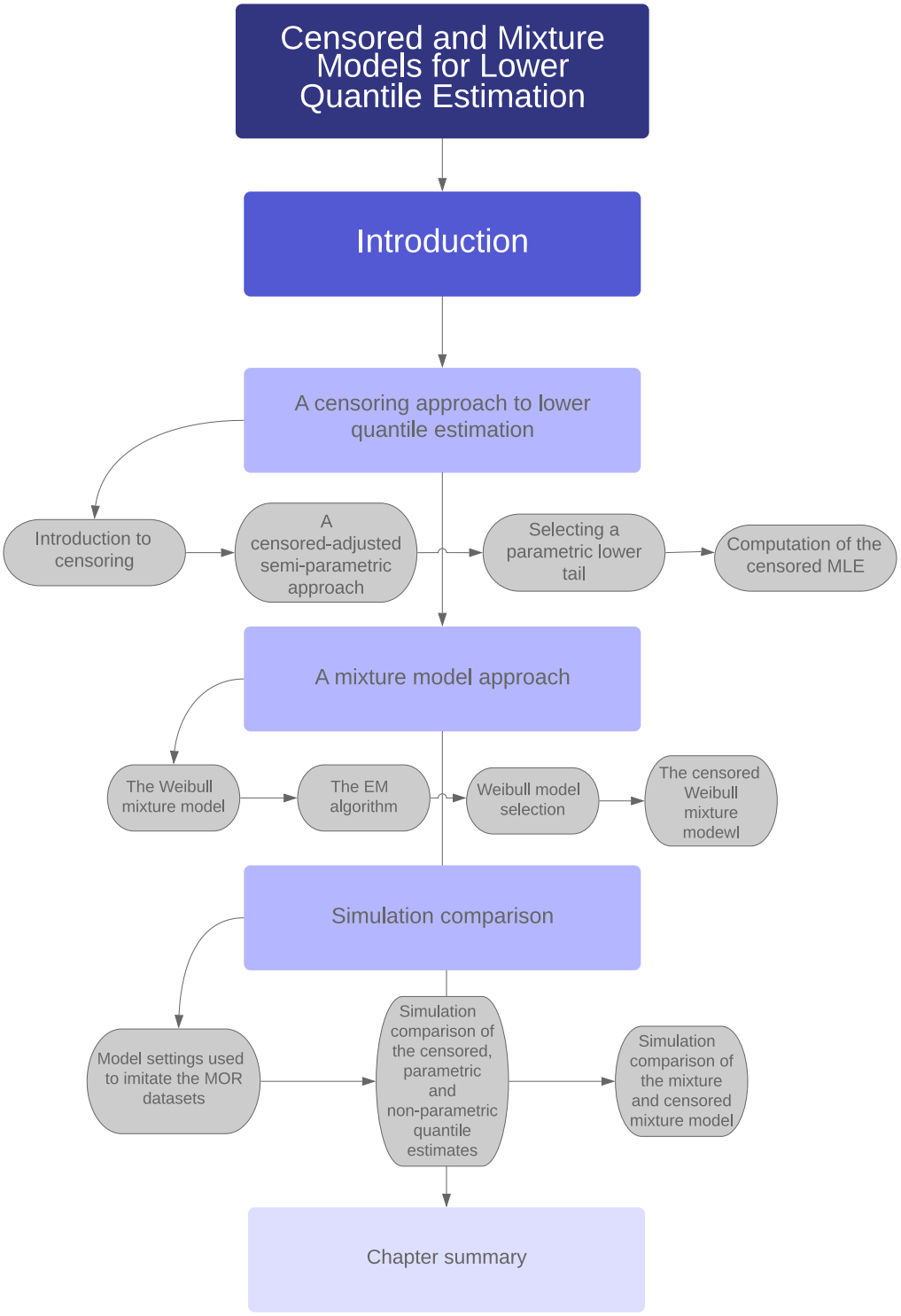
The highlights for Chapter 3 include:

- Introduction to artificial censoring and a flexible ensored-adjusted semi-parametric model for lower quantile estimation..
- Fitting a parametric distribution to the tail of a dataset using censored maximum likelihood estimation.
- A comparison of various parametric, non-parametric and censored-adjusted semi-parametric quantile estimation techniques.
- Introduction of the censored and uncensored Weibull mixture models.
- The Expectation Maximisation (EM) algorithm for parameter estimation for the censored and uncensored Weibull mixture models.
- A simulation comparison of the censored and uncensored Weibull mixture models' quantile estimates.

3.1 Introduction

Chapter 2 discussed a few shortcomings of parametric and non-parametric approaches to lower quantile estimation. One particular solution would be to consider models that only fit the left tail of the data suitably well instead of the entire distribution thereof. This alternative suggests that lower quantiles can be determined sufficiently by the PDF below the true quantile of interest [16]. Section 3.2 introduces the concept of artificial censoring that is used in order to focus on the estimation of the parametric tail. In Subsection 3.2.3, the motivation behind the choice of the Weibull distribution as the parametric left tail is motivated from a statistical and empirical perspective. The computation and formulation of the censored Weibull MLE is discussed in Subsection 3.2.4. The censored Weibull MLE approach does not make use of all the sample information due to the large proportion of censoring, and may therefore not be fully efficient. One particular approach to making use of the entire data set, whilst being mindful of the accuracy of the quantile estimate, is to consider a mixture model. The uncensored Weibull mixture model is introduced in Subsection 3.3.1. The EM algorithm that is used for parameter estimation is introduced in Subsection 3.3.2. The censored Weibull mixture model and its parameter estimates are discussed in Subsection 3.3.4. A simulation study to initially compare four (non-mixture) quantile estimation techniques is provided in Section 3.4. Finally, a simulation study to compare the censored Weibull MLE, censored and uncensored Weibull mixture quantile estimation techniques is provided in Subsection 3.4.2.

Chapter outline



Chapter 3. Censored and Mixture Models for Lower Quantile Estimation

3.2 A censoring approach to lower quantile estimation

A short introduction to censoring is discussed below, followed by the concept of *subjective censoring*. Furthermore, the *censored-adjusted semi-parametric* model used by [16] as a solution to the potential drawbacks of classical parametric and non-parametric lower quantile estimation is thoroughly re-worked and investigated.

3.2.1 Introduction to censoring

In statistics, data is censored if the exact values of each observation are not known, however, some information with respect to every observation in relation to certain bounds is available. In other words, partially observed observations will still be used in the statistical analysis. Censoring plays a vital role in statistical inference see [14]. For example, a wooden board that does not break under progressive levels of stress will not be stressed until breakage, but instead the information that it has survived up to a certain level of stress will be used in the statistical analysis.

Censoring can be considered as *informative* or *non-informative* [14]. Generally speaking, informative censoring occurs if the censored observations give additional information about the remaining uncensored observations which are to be analysed. Censoring is considered to be non-informative if the censored observations give no additional information about remaining uncensored observations. For example, during the modulus of rupture strength tests, if the darker hue wooden boards are thrown away this does not imply the lighter hue boards have a higher or lower stress load strength. Furthermore, non-informative censoring can be classified into *Type I* and *II* censoring: Type 1 censoring occurs when the censoring mechanism is known in advance. For example, a wooden board's strength is only known when it is smaller than a specified/predetermined strength value [16]. Type II censoring occurs if the observational study is continued until a predetermined number of subjects have experienced the event of interest. For example, in an experiment consisting of n wooden boards; if the stress applied to the wooden boards increase from 0 consistently (on all boards simultaneously) then the experiment will stop once r out of n boards break. The partially observed observations in this case would consist of the strength measures of the remaining $n-r$ boards. It is important to note that the strengths of the remaining $n-r$ boards are known only to be larger than $X_{(r)}$ where $X_{(1)}, X_{(2)}, \ldots, X_{(r)}$ are the smallest r order statistics [16].

The censoring mechanism used throughout the rest of this study and in Chapter 3 of [16] cannot be directly assigned to one of the categories mentioned above. Reason being, the strength of every wooden board in the MOR datasets are known, that is there is no censoring present. The methodology behind the subjective censoring mechanism is to focus on the behaviour of the left tail of the sample. This can be achieved by 'artificially' (subjectively) censoring observations larger than some threshold C. The 10^{th} empirical percentile (Type 3 definition in R) is used as the censoring threshold to estimate the 5^{th} percentile [16]. The choice of this censoring threshold is not random and as mentioned in [16], it is a requirement by the industrial standards document [1].

To incorporate the subjective censoring, where 90% of the observations are artificially censored, Type I censoring can be used. Again the 10^{th} empirical percentile (Type 3 definition in R) is used as the censoring threshold. The *Type I and II censoring likelihoods* are given by [14]:

$$L_I(\boldsymbol{X}, \underline{\theta}) = \prod_{i=1}^{n} [f(X_i; \underline{\theta})]^{\delta_i} [1 - F(C, \underline{\theta})]^{1-\delta_i} \tag{3.2.1}$$

$$L_{II}(\boldsymbol{X}, \underline{\theta}) = \frac{n!}{(n-r)!} \prod_{i=1}^{r} f\left(X_{(i)}; \underline{\theta}\right) \left\{1 - F\left(X_{(r)}\right); \underline{\theta}\right\}^{n-r}, \tag{3.2.2}$$

where $\delta_i = I(X_i \leq C)$ is an indicator function, $\boldsymbol{X} = (X_1, \ldots, X_n)$, $f(\cdot; \underline{\theta})$ and $F(\cdot; \underline{\theta})$ are the PDF and CDF respectively and $X_{(i)}, i = 1, 2, \ldots, r$, are the order statistics. The censoring can equivalently be viewed as Type II, where the experiment will be concluded once 10% of the n wooden boards break. Here, it should be noted that if $C = X_{(r)}$, then (3.2.1) and (3.2.2) only differ by the constant $\frac{n!}{(n-r)!}$, since $\delta_i = 1$ only r times out of n. Interested readers are referred to Chapter 3 of [16] for a detailed explanation. The Type I censoring will be used throughout this study except for Chapter 4 where Type II will be used.

3.2.2 A censored-adjusted semi-parametric approach to lower quantile estimation

It is well known that non-parametric estimators have a tendency to suffer from larger varainces compared to their parametric counterparts, however, parametric estimators may suffer from larger biases under miss-specified models [16]. An interesting solution to these shortcomings would be to consider the censored-adjusted semi-parametric model used in [16]. That is, to consider models that only fit the left tail of the data suitably well instead of the entire distribution thereof. This alternative suggests that lower quantiles can be determined sufficiently by the PDF below the true quantile of interest [16]. Therefore, a parametric distribution only has to be specified for the lower tail, which results in a censored-adjusted semi-parametric model [16]:

$$g\left(x;\underline{\theta},C\right)=\begin{cases}f\left(x;\underline{\theta}\right) & x\leq C\\ h\left(x\right) & x>C\end{cases}. \tag{3.2.3}$$

Here $f\left(x;\underline{\boldsymbol{\theta}}\right)$ is the parametric PDF for the lower tail. C is referred to as the censoring threshold, where the non-parametric part, $h\left(x\right)$, starts and the parametric PDF ends ($h\left(x\right)$ can be an unspecified PDF). The following two requirements ensure that $g\left(x\right)$ is a well defined PDF [16]:

1. $h\left(x\right)\geq 0, \quad x>C$
2. $\int_{C}^{\infty} h\left(x\right)dx=1-G\left(C\right)=1-F\left(C,\underline{\theta}\right).$

The elegance of this censored-adjusted semi-parametric approach allows a quantile estimate to be obtained using only the parametric PDF fitted to the lower tail of a dataset. When estimating the lower quantile of interest, it is not necessary to estimate $h\left(x\right)$ nor C for this semi-parametric model [16]. *Subjective/artificial censoring* methods will be used for parameter estimation of the parametric PDF in (3.2.3).

A note going forward

In order to distinguish the subjective censoring approach from ordinary Type 1 censoring, the above approach of maximising the subjective censored likelihood will be referred to as the censored MLE where the threshold is the p^{th} empirical percentile. It should be noted that the empirical quantile that will be used as the threshold is the Type 3 definition discussed in Chapter 2, as specified in the industrial standard document [1] and [16]. The empirical quantile that is treated as a quantile estimate is the Type 9 definition, which performed better than Type 3 according to Chapter 2.

For ease of readability, the *censored Weibull MLE* (the motivation behind this choice of parametric model is discussed below) with the threshold set at the 10^{th} empirical percentile will be denoted by CW-MLE. The MLE without censoring will be referred as the *ordinary MLE* (O-MLE).

3.2.3 Selecting a parametric lower tail

The two fundamental components of (3.2.3) are the subjective censoring and the choice of the parametric tail. The four distributions considered were presented in Chapter 2 Subsection 2.3.1. These distributions are considered to align with the work presented in [16].

All models discussed in Chapter 2 Subsection 2.3.1 are well rooted in literature, in particular for modeling extreme lower tail behaviour. Each model is fitted to the lower tail of the MOR1 and MOR2 datasets, whilst making use of artificial censoring. The 10^{th} empirical percentile is used as the censoring threshold. The plots of the censored MLE for the four models can be seen in Figure 3.2.1, which provide a further (empirical) justification for the choice of parametric model.

$D^{TRUNC}\left(\widetilde{F}_n - \hat{F}_n(x)\right)$			Quantile Estimate	
Model	MOR1	MOR2	MOR1	MOR2
Weibull	0.014	0.011	4.50	4.64
Log-normal	0.013	0.009	4.48	4.59
Gamma	0.013	0.009	4.51	4.57
Minimum Gumbel	0.015	0.013	4.53	4.71

Table 3.2.1: Left tail goodness-of-fit of the parametric models and corresponding 5^{th} quantile estimates for MOR1 and MOR2 datasets.

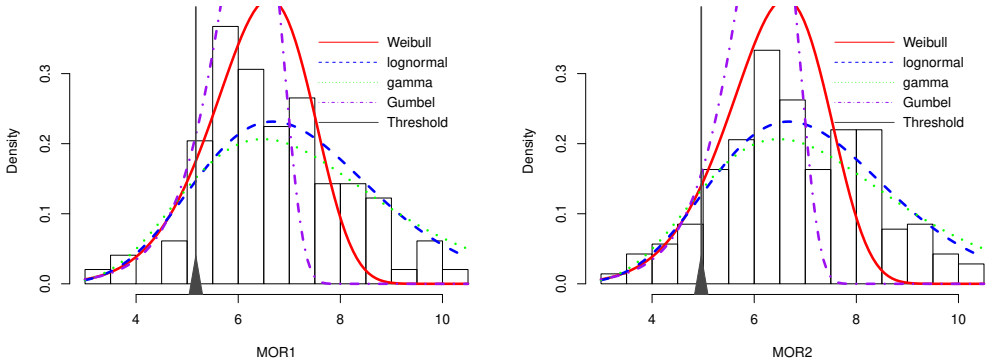


Figure 3.2.1: Subjectively censored parametric models for MOR1 and MOR2 datasets.

In Figure 3.2.1 the curves of the models for both datasets differ significantly on the right of the censoring threshold (indicated by the vertical line) or the right tail. In the left tail, the curves are indistinguishable. The y axis has been limited to 0.3 to ensure as little distortion as possible. In order to study how well these models approximate the left tail of the dataset, a *truncated Kolmogorov-Smirnov statistic* is used [16]. This statistic measures the maximum absolute distance between the ECDF $\hat{F}_n(x)$ and the fitted parametric CDF, $\widetilde{F}_n(x)$, up to the censoring threshold C :

$$D^{TRUNC}\left(\widetilde{F}_n - \hat{F}_n\right) = \sup_{x \leq C}\left\{\left|\hat{F}_n - \widetilde{F}_n\right|\right\}. \tag{3.2.4}$$

The $D^{TRUNC}\left(\widetilde{F}_n - \hat{F}_n\right)$ statistic for each of the four models considered is presented in Table 3.2.1. Here, the $D^{TRUNC}(\cdot)$ notation is used to indicate that a truncated KS is used. It should be clear that the $D^{TRUNC}\left(\widetilde{F}_n - \hat{F}_n\right)$ for the different models are quite similar for each dataset. For both the MOR1 and MOR2 dataset the values are around 0.01, which is an indication that these models fit the left tail of the datasets reasonably well. Similarly, it should not be surprising that the 5^{th} quantile estimates for the different models do not differ significantly from one another for each dataset. Ultimately, the choice of the parametric lower tail will not have a major impact on the quantile estimate. Given that the fit of the models is quite indistinguishable, the recommendation by the [1] and [16] will be followed and the Weibull distribution will be used to model the lower tail in the CW-MLE (discussed below). The censored likelihood and quantile functions (where applicable) for the gamma, log-normal and minimum Gumbel distribution have also been provided for completeness.

3.2.4 Computation of the censored MLE

The approach followed in this section is to re-work and supplement Section 3.5 in [16]. That is, to maximise the Type I censored likelihood (3.2.1) in order to obtain the parameter estimates for the lower tail of the four distributions discussed in Chapter 2 Subsection 2.3.1. These parameter estimates can in turn be used to obtain quantile estimates. In particular the logarithm of (3.2.1) will be maximised.

3.2.4.1 The censored Weibull MLE

$$\begin{aligned} l(\mathbf{X};\beta,\zeta,C) &= log\left(\prod_{i=1}^{n}\left[f(X_i;\underline{\theta})\right]^{\delta_i}\left[1-F(C,\underline{\theta})\right]^{1-\delta_i}\right) \\ &= \sum_{i=1}^{r} log\left(f(X_i;\underline{\theta})\right) + \sum_{i=1}^{n-r} log\left(1-F(C;\underline{\theta})\right) \\ &= \sum_{i=1}^{r} log\left(\frac{\beta}{\eta}\left(\frac{x_i}{\eta}\right)^{\beta-1} exp\left(-\left(\frac{x_i}{\eta}\right)^{\beta}\right)\right) + \sum_{i=1}^{n-r} log\left(1-\left(1-exp\left(-\left(\frac{C}{\eta}\right)^{\beta}\right)\right)\right) \\ &= rlog(\beta) + rlog(\zeta) + (\beta-1)\sum_{i=1}^{r} log(x_i) - \zeta\sum_{i=1}^{r} x_i^{\beta} - (n-r)\zeta C^{\beta}. \end{aligned} \tag{3.2.5}$$

It is assumed that $X_1, X_2, \ldots, X_n$ are *i.i.d.* from a population model F (the choice of F will be discussed later) and $X_1, X_2, \ldots, X_r \leq C$ and $X_{r+1}, X_{r+2}, \ldots, X_n \geq C$ four all four models discussed here. Furthermore, $\zeta = \eta^{-\beta}$ is used for computational convenience. Lastly, if $C > max\{X_i\}$ for all i then it follows that $r = n$ and (3.2.5) is the likelihood of the ordinary Weibull MLE.

Taking the derivative of (3.2.5) with respect to β and ζ,

$$\begin{cases} \frac{\partial l}{\partial \beta} = \frac{r}{\beta} + \sum_{i=1}^{r} log(x_i) - (n-r)\zeta C^{\beta} log(C) - \zeta\sum_{i=1}^{r} log(x_i)\, x_i^{\beta} \\ \frac{\partial l}{\partial \zeta} = \frac{r}{\zeta} - \sum_{i=1}^{r} x_i^{\beta} - (n-r)C^{\beta}. \end{cases} \tag{3.2.6}$$

The solution to $\tilde{\beta}$ and $\tilde{\zeta}$ in the above system of equations are the maximum likelihood estimates according to [19]. Algebraic substitution yields the following,

$$\frac{1}{\beta} = \frac{\zeta}{r}\left[\sum_{i=1}^{r} log(x_i)\, x_i^{\beta} + (n-r)C^{\beta} log(C)\right] - \frac{\sum_{i=1}^{r} log(x_i)}{r} \tag{3.2.7}$$

$$\frac{1}{\zeta} = \frac{\sum_{i=1}^{r} x_i^{\beta} - (n-r)C^{\beta}}{r}. \tag{3.2.8}$$

The Newton-Raphson method can be used to solve (3.2.7) and (3.2.8) simultaneously.
According to [19] the value of β is guaranteed to converge under mild conditions. The original scale parameter can then be obtained (using the property of invariance [2]) as $\tilde{\eta} = \tilde{\zeta}^{-1/\tilde{\beta}}$. Finally, the quantile estimate $\tilde{q}$ for the censored Weibull can be calculated as

$$\tilde{q} = \tilde{\eta}\left[-log(1-p)\right]^{1/\tilde{\beta}}. \tag{3.2.9}$$

3.2.4.2 The censored Gamma MLE

$$\begin{aligned} l(\mathbf{X};\beta,\zeta,C) &= log\left(\prod_{i=1}^{n}\left[f(X_i;\underline{\theta})\right]^{\delta_i}\left[1-F(C,\underline{\theta})\right]^{1-\delta_i}\right) \\ &= \sum_{i=1}^{r} log\left(f(X_i;\underline{\theta})\right) + \sum_{i=1}^{n-r} log\left(1-F(C;\underline{\theta})\right) \\ &= \sum_{i=1}^{r} log\left(\frac{x_i^{\theta-1}}{\Gamma(\theta)\kappa^{\theta}} exp\left(\frac{x_i}{\kappa}\right)\right) + \sum_{i=1}^{n-r} log\left(1-\frac{\gamma\left(\theta,\frac{C}{\kappa}\right)}{\Gamma(\theta)}\right) \\ &= (\theta-1)\sum_{i=1}^{r} log(x_i) - \sum_{i=1}^{r}\frac{x_i}{\kappa} - rlog\left(\Gamma(\theta)\right) - r\theta log(\kappa) + (n-r)\,log\left[1-\frac{\gamma\left(\theta,\frac{C}{\kappa}\right)}{\Gamma(\theta)}\right]. \end{aligned} \tag{3.2.10}$$

Numerical methods (such as 'optim' in R) can be used to to obtain parameter estimates that maximise (3.2.10). Furthermore, the quantile function for the gamma distribution does not have a closed form expression and has to be estimated numerically, for example using the 'quantile' function in R.

3.2.4.3 The censored log-normal MLE

$$\begin{aligned} l(\mathbf{X};\beta,\zeta,C) &= log\left(\prod_{i=1}^{n}\left[f(X_i;\underline{\theta})\right]^{\delta_i}\left[1-F(C,\underline{\theta})\right]^{1-\delta_i}\right) \\ &= \sum_{i=1}^{r} log\left(f(X_i;\underline{\theta})\right) + \sum_{i=1}^{n-r} log\left(1-F(C;\underline{\theta})\right) \\ &= \sum_{i=1}^{r} log\left(\frac{1}{x_i\sigma\sqrt{2\pi}} exp\left(-\frac{\left(log(x_i)-\mu\right)^2}{2\sigma^2}\right)\right) + \sum_{i=1}^{n-r} log\left(1-\Phi\left(\frac{log(C)-\mu}{\sigma}\right)\right) \end{aligned}$$

$$\begin{aligned} &= \sum_{i=1}^{r} log\left(\frac{1}{x_i\sigma\sqrt{2\pi}} exp\left(-\frac{\left(log(x_i)-\mu\right)^2}{2\sigma^2}\right)\right) + \sum_{i=1}^{n-r} log\left(1-\Phi\left(\frac{log(C)-\mu}{\sigma}\right)\right) \\ &= -\frac{r}{2}log\left(2\pi\sigma^2\right) - \sum_{i=1}^{r} log(x_i) - \sum_{i=1}^{r}\frac{log(x_i)}{2\sigma^2} + \sum_{i=1}^{r}\frac{\mu log(x_i)}{\sigma^2} - \frac{r\mu^2}{2\sigma^2} \\ &+ (n-r)\,log\left[1-\Phi\left(\frac{log(C)-\mu}{\sigma}\right)\right]. \end{aligned} \tag{3.2.11}$$

Taking the derivative of (3.2.11) with respect to μ and σ,

$$\begin{cases} \frac{\partial l}{\partial \mu} = \frac{\sum_{i=1}^{r} log(x_i)}{\sigma^2} - \frac{r\mu}{\sigma^2} - (n-r)\left[1-\Phi\left(\frac{log(C)-\mu}{\sigma}\right)\right]^{-1}\left(\frac{exp\left(-\frac{1}{2}\left(\frac{log(C)-\mu}{\sigma}\right)^2\right)}{\sqrt{2\pi}\sigma}\right) \\ \frac{\partial l}{\partial \sigma} = -\frac{r}{\sigma} + \sum_{i=1}^{r}\frac{log(x_i)}{\sigma^3} - 2\sum_{i=1}^{r}\frac{log(x_i)\mu}{\sigma^3} + \frac{r\mu^2}{\sigma^3} + (n-r)\left[1-\Phi\left(\frac{log(C)-\mu}{\sigma}\right)\right]^{-1}(\mu - log(C)). \end{cases} \tag{3.2.12}$$

Numerical methods (such as 'optim' in R) can be used to to obtain parameter estimates for μ and σ in (3.2.12) simultaneously. Finally, the quantile estimate $\tilde{q}$ for the censored log-normal can be calculated as

$$\tilde{q} = exp\left(\tilde{\mu} + \tilde{\sigma}\Phi^{-1}(p)\right). \tag{3.2.13}$$

3.2.4.4 The censored minimum Gumbel MLE

$$\begin{aligned} l(\mathbf{X};\,\beta,\,\zeta,\,C) &= log\left(\prod_{i=1}^{n}\left[f\left(X_i;\,\underline{\theta}\right)\right]^{\delta_i}\left[1-F\left(C,\,\underline{\theta}\right)\right]^{1-\delta_i}\right) \\ &= \sum_{i=1}^{r} log\left(f\left(X_i;\,\underline{\theta}\right)\right) + \sum_{i=1}^{n-r} log\left(1-F\left(C;\,\underline{\theta}\right)\right) \\ &= \sum_{i=1}^{r} log\left(\frac{1}{b}exp\left(\frac{x_i-a}{b}-exp\left(\frac{x_i-a}{b}\right)\right)\right) + \sum_{i=1}^{n-r} log\left(1-\left(1-exp\left(-exp\left(\frac{C-a}{b}\right)\right)\right)\right) \\ &= -rlog\left(b\right) + \sum_{i=1}^{r}\frac{x_i-a}{b} - \sum_{i=1}^{r} exp\left(\frac{x_i-a}{b}\right) - (n-r)\,exp\left(\frac{C-a}{b}\right). \end{aligned} \tag{3.2.14}$$

Taking the derivative of (3.2.11) with respect to a and b,

$$\begin{cases} \frac{\partial l}{\partial b} = -\frac{r}{b} - \sum_{i=1}^{r}\frac{x_i-a}{b^2} + \sum_{i=1}^{r} exp\left(\frac{x_i-a}{b}\right)\left(\frac{x_i-a}{b^2}\right) + (n-r)\,exp\left(\frac{C-a}{b}\right)\left(\frac{C-a}{b^2}\right) \\ \frac{\partial l}{\partial a} = \frac{r}{b} + \sum_{i=1}^{r}\frac{exp\left(\frac{x_i-a}{b}\right)}{b} + \frac{(n-r)exp\left(\frac{C-a}{b}\right)}{b}. \end{cases} \tag{3.2.15}$$

Numerical methods (such as 'optim' in R) can be used to to obtain parameter estimates for a and b in (3.2.15) simultaneously. Finally, the quantile estimate $\tilde{q}$ for the censored minimum Gumbel can be calculated as

$$\tilde{q} = \widetilde{a} + \widetilde{b}log\left(-log\left(1-p\right)\right). \tag{3.2.16}$$

A note going forward

As mentioned in Subsection 3.2.3, the Weibull distribution will be used to model the lower tail in the CW-MLE. The simulation study will be provided at the end of this chapter, specifically Section 3.4 and Sunsection 3.4.2, once all prospective models have been introduced.

3.3 A mixture model approach to lower quantile estimation

The CW-MLE approach does not make use of all the sample information due to the large proportion of censoring, and may therefore not be fully efficient. One particular approach to making use of the entire data set, whilst being mindful of the accuracy of the quantile estimate, is to consider a mixture model. Both the MOR1 and MOR2 datasets display bi-modality, therefore, it is reasonable to assume that the data is generated from a mixture of two uni-modal distributions. For the sake of this dissertation, the mixing distributions are assumed to come from the same family of distributions, with PDF given by

$$f\left(x\right) = \pi_1 f_1\left(x\right) + \pi_2 f_2\left(x\right), \tag{3.3.1}$$

where $f_1\left(\cdot\right)$ and $f_2\left(\cdot\right)$ are the PDFs of the two contributing populations and π_1 and $\pi_2 = (1-\pi_1)$ represent the proportion contributed by the first and second population respectively. The above formation implies that the random variable X has a probability π_1 of belonging to the first population and $\pi_2 = (1-\pi_1)$ to the second population. For the sake of this dissertation, the mixing distributions are assumed to come from the same family of distributions. This is the same approach followed by [16].

The CW-MLE approach censors 90% of the information available in data and is therefore not completely efficient in terms of information usage. Introducing a more complex model, such as a mixture model, to make use of more sample information (less censoring) and provide additional flexibility to improve the desired quantile estimate qualities may prove to be beneficial. Mixture models benefit from this desired flexibility

due the additional parameters which can be used to improve the approximation of the lower tail of the given dataset.

It is worth mentioning that only a mixture of two univariate Weibull distributions are considered for estimating the 5^{th} quantile. The reason being, this section primarily focuses on revisiting and supplementing the content of Chapter 4 in [16].

3.3.1 The Weibull mixture model

Consider the following mixture of two univariate, two-parameter Weibull distributions with PDF given by:

$$f\left(x;\, p,\, \beta_1,\, \eta_1,\, \beta_2,\, \eta_2\right) = \pi f\left(x;\, \beta_1,\, \eta_1\right) + \left(1-\pi\right) f\left(x;\, \beta_2,\, \eta_2\right) \tag{3.3.2}$$

$$= \pi \left(\frac{\beta_1}{\eta_1}\right)\left(\frac{x}{\eta_1}\right)^{\beta_1 - 1} exp\left(-\left(\frac{x}{\eta_1}\right)^{\beta_1}\right) + \left(1-\pi\right)\left(\frac{\beta_2}{\eta_2}\right)\left(\frac{x}{\eta_2}\right)^{\beta_2 - 1} exp\left(-\left(\frac{x}{\eta_2}\right)^{\beta_2}\right) \quad x > 0.$$

The random variable X from this mixture distribution has a probability of π of belonging to a Weibull$(\beta_1,\, \eta_1)$ and a probability of $(1-\pi)$ to a Weibull$(\beta_2,\, \eta_2)$. It should be noted that the labels indicating to which distribution the random variable X belongs, are not observable, and is referred to as a latent variable. This sets the stage for the EM algorithm.

3.3.2 The EM algorithm and estimation of the Weibull mixture

To estimate the parameters $\underline{\theta} = (\pi,\, \beta_1,\, \eta_1,\, \beta_2,\, \eta_2)$ in (3.3.2) requires a direct maximisation of the following *incomplete log-likelihood*:

$$l^I\left(\boldsymbol{X}, \underline{\theta}\right) = \sum_{i=1}^{n} log\left(\pi f\left(x_i;\, \beta_1,\, \eta_1\right) + \left(1-\pi\right) f\left(x_i;\, \beta_2,\, \eta_2\right)\right), \tag{3.3.3}$$

which may prove to be numerically cumbersome. The mathematics would be more tractable if the membership distribution of each X_i were known. In other words, if the sub-population to which each X_i belonged were known. A common practice to circumvent this issue is to define an indicator variable, $Z_i = I\{X_i \sim$ Weibull $(\beta_1,\, \eta_1)\}$, so that the PDF in (3.3.2) can be re-written as:

$$f\left(x,\, z,\, \underline{\theta}\right) = \left\{\pi f\left(x;\, \beta_1,\, \eta_1\right)\right\}^{Z} \left\{\left(1-\pi\right) f\left(x;\, \beta_2,\, \eta_2\right)\right\}^{1-Z}.$$

Furthermore, $\underline{\theta}$ can be estimated by using the following *complete log-likelihood:*

$$\begin{aligned} l^C\left(\boldsymbol{X},\, \boldsymbol{Z};\, \underline{\theta}\right) &= \sum_{i=1}^{n} log\left(\left\{\pi f\left(x_i;\, \beta_1,\, \eta_1\right)\right\}^{Z_i} \left\{\left(1-\pi\right) f\left(x_i;\, \beta_2,\, \eta_2\right)\right\}^{1-Z_i}\right) \\ &= \sum_{i=1}^{n} Z_i \left\{log\left(\pi f\left(x_i;\, \beta_1,\, \eta_1\right)\right)\right\} + \left(1-Z_i\right)\left\{log\left(\left(1-\pi\right) f\left(x_i;\, \beta_2,\, \eta_2\right)\right)\right\}. \end{aligned} \tag{3.3.4}$$

To distinguish between the complete log-likelihood in (3.3.4), it is denoted with the super-script "C", while the incomplete log-likelihood in (3.3.3) is denoted by the super-script "I". This notation is adopted from [16] for its simplistic readability.

The Expectation-Maximisation (EM) algorithm was first introduced by [6] and was designed to perform the maximum likelihood estimation of parameters, where the form of the likelihood is given by (3.3.4). The EM algorithm is an iterative process that is used to obtain maximum likelihood estimators in the presence of incomplete data. The EM algorithm starts by selecting a set of initial parameters $\underline{\theta}^{(r)}$, that is

$\pi^{(r)}$, $\beta_1^{(r)}$, $\beta_2^{(r)}$, $\eta_1^{(r)}$, $\eta_2^{(r)}$. Here r denotes the r^{th} iteration of the algorithm. In the expectation step (E-step), the expected value of the complete log-likelihood function of $\underline{\theta}$, with respect to the conditional distribution of the latent variables $\boldsymbol{Z}$, given the observed sample $\boldsymbol{X}$ and the current estimates of the parameters $\underline{\theta}^{(r)}$ is calculated as:

$$\begin{aligned}
Q\left(\underline{\theta}, \underline{\theta}^{(r)}\right) &= E_{\boldsymbol{Z}|\boldsymbol{X}, \underline{\theta}^{(r)}}\left(l^C\left(\boldsymbol{X}, \boldsymbol{Z}; \underline{\theta}\right)\right) \\
&= E_{\boldsymbol{Z}|\boldsymbol{X}, \underline{\theta}^{(r)}}\left(\sum_{i=1}^{n} log\left(\left\{\pi^{(r)} f\left(x_i; \beta_1^{(r)}, \eta_1^{(r)}\right)\right\}^{Z_i}\left\{\left(1-\pi^{(r)}\right) f\left(x_i; \beta_2^{(r)}, \eta_2^{(r)}\right)\right\}^{1-Z_i}\right)\right) \\
&= E_{\boldsymbol{Z}|\boldsymbol{X}, \underline{\theta}^{(r)}}\left(\sum_{i=1}^{n} log\left(\prod_{k=1}^{2}\left\{\pi_k^{(r)} f\left(x_i; \beta_k^{(r)}, \eta_k^{(r)}\right)\right\}^{Z_{ik}}\right)\right) \\
&= E_{\boldsymbol{Z}|\boldsymbol{X}, \underline{\theta}^{(r)}}\left(\sum_{i=1}^{n}\sum_{k=1}^{2} Z_{ik} log\left(\left\{\pi_k^{(r)} f\left(x_i; \beta_k^{(r)}, \eta_k^{(r)}\right)\right\}\right)\right) \\
&= \sum_{i=1}^{n}\sum_{k=1}^{2} E_{\boldsymbol{Z}_i|\boldsymbol{X}, \underline{\theta}^{(r)}}\left(Z_{ik} log\left(\left\{\pi_k^{(r)} f\left(x_i; \beta_k^{(r)}, \eta_k^{(r)}\right)\right\}\right)\right).
\end{aligned}$$

Here, $\boldsymbol{X}$ is an $(n \times 1)$ column vector and $\boldsymbol{Z}$ is an $(n \times K = 2)$ matrix, where the k^{th} column of each row represents the unobserved probability of belonging to mixing component k. Also, $\pi_1 = \pi$ and $\pi_2 = (1-\pi)$. Given the current estimates of the parameters, $\underline{\theta}^{(r)}$, the conditional distribution of each row vector $\boldsymbol{Z}_i$ (also known as the membership weights) is determined using Bayes theorem [6]:

$$\begin{aligned}
E_{\boldsymbol{Z}_i|\boldsymbol{X}, \underline{\theta}^{(r)}}\left(Z_{ik} log\left(\left\{\pi_k^{(r)} f\left(x_i; \beta_k^{(r)}, \eta_k^{(r)}\right)\right\}\right)\right) &= 1 \times P\left(Z_{ik} = 1|\boldsymbol{x}, \underline{\theta}^{(r)}\right) + 0 \times P\left(Z_{ik} = 0|\boldsymbol{x}, \underline{\theta}^{(r)}\right) \\
&= P\left(Z_{ik} = 1|\boldsymbol{x}, \underline{\theta}^{(r)}\right) \\
&= \frac{\pi_k^{(r)} f\left(\boldsymbol{x}; \beta_k^{(r)}, \eta_k^{(r)}\right)}{\sum_{j=1}^{2} \pi_j^{(r)} f\left(\boldsymbol{x}; \beta_j^{(r)}, \eta_j^{(r)}\right)} \\
&= \gamma_{ik}^{(r)}.
\end{aligned}$$

Therefore it follows that:

$$Q\left(\underline{\theta}, \underline{\theta}^{(r)}\right) = \sum_{i=1}^{n}\sum_{k=1}^{2} \gamma_{ik}^{(r)} log\left(\left\{\pi_k^{(r)} f\left(x_i; \beta_k^{(r)}, \eta_k^{(r)}\right)\right\}\right).$$

The maximisation step (M-step) computes the parameters which maximise the expected log-likelihood in the E-step. That is, the parameters $\underline{\theta}^{(r)}$ are updated to $\underline{\theta}^{(r+1)}$ such that $\underline{\theta}^{(r+1)}$ maximises $Q\left(\underline{\theta}, \underline{\theta}^{(r)}\right)$. Finally, this procedure is repeated until the difference between successive log-likelihoods becomes significantly small. The EM algorithm is summarised in Algorithm 3.1. From the M-step, the updated membership weights, conditioned on $\underline{\theta}^{(r)}$, can be calculated as

$$\begin{aligned}
\gamma_{ik}^{(r+1)} &= E\left(z_{ik}|\boldsymbol{X}, \underline{\theta}^{(r)}\right). \\
&= \frac{\pi_k^{(r)} f\left(\boldsymbol{x}; \beta_k^{(r)}, \eta_k^{(r)}\right)}{\sum_{j=1}^{2} \pi_j^{(r)} f\left(\boldsymbol{x}; \beta_j^{(r)}, \eta_j^{(r)}\right)}.
\end{aligned} \tag{3.3.5}$$

Substituting $\gamma_{ik}^{(r+1)}$ into (3.3.4), leads to

$$l^C\left(\boldsymbol{X}, \boldsymbol{\Gamma}^{(r+1)}; \underline{\theta}\right) = Q\left(\underline{\theta}, \underline{\theta}^{*(r)}\right) = \sum_{i=1}^{n}\sum_{k=1}^{2} \gamma_{ik}^{(r+1)} log\left(\left\{\pi_k^{(r)} f\left(x_i; \beta_k^{(r)}, \eta_k^{(r)}\right)\right\}\right). \tag{3.3.6}$$

Where $\boldsymbol{\Gamma}^{(r+1)}$ is an $(n \times K = 2)$ matrix, where the k^{th} column of each row represents the updated membership weighs of belonging to mixing component k. Maximising $l^C\left(\boldsymbol{X}, \boldsymbol{\Gamma}^{(r+1)}; \underline{\theta}\right)$ with respect to $\pi^{(r)}$, taking into consideration that $\sum_{k=1}^{2} \pi_k^{(r)} = 1$, requires a Lagrangian multiplier (denoted by $\mathcal{L}(\cdot)$). That is

$$\begin{aligned}\mathcal{L}\left(\underline{\theta}^{(r)}, \lambda\right) &= Q^*\left(\underline{\theta}, \underline{\theta}^{*(r)}\right) \\ &= \sum_{i=1}^{n}\sum_{k=1}^{2} \gamma_{ik}^{(r+1)} log\left(\left\{\pi_k^{(r)} f\left(x_i; \beta_k^{(r)}, \eta_k^{(r)}\right)\right\}\right) + \lambda\left(\sum_{k=1}^{2} \pi_k^{(r)} = 1\right).\end{aligned}$$

It follows that

$$\frac{\partial Q^*\left(\underline{\theta}, \underline{\theta}^{*(r)}\right)}{\partial \pi_k^{(r)}} = \sum_{i=1}^{n} \frac{\gamma_{ik}^{(r+1)}}{\pi_k^{(r)}} + \lambda, \tag{3.3.7}$$

summing over k and multiplying by $\pi_k^{(r)}$ gives

$$\begin{aligned}\sum_{k=1}^{2}\sum_{i=1}^{n} \gamma_{ik}^{(r+1)} + \lambda \sum_{k=1}^{2} \pi_k^{(r)} &= n + \lambda \\ \therefore \qquad \lambda &= -n.\end{aligned}$$

Finally

$$\begin{aligned}\frac{\partial Q^*\left(\underline{\theta}, \underline{\theta}^{*(r)}\right)}{\partial \pi_k^{(r)}} &= \sum_{i=1}^{n} \frac{\gamma_{ik}^{(r+1)}}{\pi_k^{(r+1)}} - n \\ \therefore \qquad \pi_k^{(r+1)} &= \frac{1}{n}\sum_{i=1}^{n} \gamma_{ik}^{(r+1)}.\end{aligned}$$

Similarly, by differentiating the complete log-likelihood in (3.3.4) with respect to the remaining parameters estimates, $\beta_1^{(r)}$, $\beta_2^{(r)}$, $\eta_1^{(r)}$and $\eta_2^{(r)}$, the updated parameter estimates can be obtained as follows:

$$\begin{aligned}\frac{\partial Q\left(\underline{\theta}, \underline{\theta}^{*(r)}\right)}{\partial \beta_k^{(r)}} &= \frac{\partial \sum_{i=1}^{n}\sum_{k=1}^{2} \gamma_{ik}^{(r+1)} log\left(\left\{\pi_k^{(r)}\left(\frac{\beta_k^{(r)}}{\eta_k^{(r)}}\right)\left(\frac{X_i}{\eta_k^{(r)}}\right)^{\beta_k^{(r)}-1} exp\left(-\left(\frac{x_i}{\eta_k^{(r)}}\right)^{\beta_k^{(r)}}\right)\right\}\right)}{\partial \beta_k^{(r)}} \\ &= \frac{\partial \sum_{i=1}^{n}\sum_{k=1}^{2} \gamma_{ik}^{(r+1)} log\left(\left\{\pi_k^{(r)}\beta_k^{(r)} x_i^{\beta_k^{(r)}-1}\left(\eta_k^{(r)}\right)^{-\beta_k^{(r)}} exp\left(-\left(x_i^{\beta_k^{(r)}}\left(\eta_k^{(r)}\right)^{-\beta_k^{(r)}}\right)\right)\right\}\right)}{\partial \beta_k^{(r)}} \\ &= \frac{\partial \sum_{i=1}^{n}\sum_{k=1}^{2} \gamma_{ik}^{(r+1)} log\left(\left\{\pi_k^{(r)}\beta_k^{(r)} x_i^{\beta_k^{(r)}-1}\zeta_k^{(r)} exp\left(-\left(x_i^{\beta_k^{(r)}}\zeta^{(r)}\right)\right)\right\}\right)}{\partial \beta_k^{(r)}}\end{aligned}$$

$$= \frac{\partial \sum_{i=1}^{n}\sum_{k=1}^{2} \gamma_{ik}^{(r+1)} \left\{ log\left(\pi_k^{(r)}\right) + log\left(\beta_k^{(r)}\right) + log\left(x_i^{\beta_k^{(r)}-1}\right) + log\left(\zeta_k^{(r)}\right) - \left(x_i^{\beta_k^{(r)}}\zeta_k^{(r)}\right)\right\}}{\partial \beta_k^{(r)}}$$

$$= \sum_{i=1}^{n} \gamma_{ik}^{(r+1)} \left\{ \left(\frac{1}{\beta_k^{(r)}}\right) + log\left(x_i\right) - \zeta_k^{(r)} x_i^{\beta_k^{(r)}} log\left(x_i\right)\right\} \stackrel{set}{=} 0$$

$$\therefore \frac{1}{\beta_k^{(r+1)}} = \frac{\sum_{i=1}^{n} \gamma_{ik}^{(r+1)} x_i^{\beta_k^{(r)}} log\left(x_i\right)}{\sum_{i=1}^{n} \gamma_{ik}^{(r+1)} \left(\eta_k^{(r)}\right)^{\beta_k^{(r)}}} - \frac{\sum_{i=1}^{n} \gamma_{ik}^{(r+1)} log\left(x_i\right)}{\sum_{i=1}^{n} \gamma_{ik}^{(r+1)}} \tag{3.3.8}$$

$$= \frac{\sum_{i=1}^{n} \gamma_{ik}^{(r+1)} x_i^{\beta_k^{(r)}} log\left(X_i\right)}{\sum_{i=1}^{n} \gamma_{ik}^{(r+1)} x_i^{\beta_k^{(r)}}} - \frac{\sum_{i=1}^{n} \gamma_{ik}^{(r+1)} log\left(x_i\right)}{\sum_{i=1}^{n} \gamma_{ik}^{(r+1)}}. \tag{3.3.9}$$

Then, $\eta_k^{(r+1)}$ is obtained as follows

$$\frac{\partial Q\left(\underline{\theta}, \underline{\theta}^{*(r)}\right)}{\partial \zeta_k^{(r)}} = \frac{\partial \sum_{i=1}^{n}\sum_{k=1}^{2} \gamma_{ik}^{(r+1)} \left\{ log\left(\pi_k^{(r)}\right) + log\left(\beta_k^{(r+1)}\right) + log\left(X_i^{\beta_k^{(r+1)}-1}\right) + log\left(\zeta_k^{(r)}\right) - \left(X_i^{\beta_k^{(r+1)}}\zeta_k^{(r)}\right)\right\}}{\partial \zeta_k^{(r)}}$$

$$= \sum_{i=1}^{n} \gamma_{ik}^{(r+1)} \left(\frac{1}{\zeta_k^{(r)}} - X_i^{\beta_k^{(r+1)}}\right) \stackrel{set}{=} 0$$

$$\therefore \frac{1}{\zeta_k^{(r+1)}} = \frac{\sum_{i=1}^{n} \gamma_{ik}^{(r+1)} X_i^{\beta_k^{(r+1)}}}{\sum_{i=1}^{n} \gamma_{ik}^{(r+1)}}$$

$$\therefore \eta_k^{(r+1)} = \left(\frac{\sum_{i=1}^{n} \gamma_{ik}^{(r+1)} X_i^{\beta_k^{(r+1)}}}{\sum_{i=1}^{n} \gamma_{ik}^{(r+1)}}\right)^{\frac{1}{\beta_k^{(r+1)}}}. \tag{3.3.10}$$

Here, $\underline{\theta}^{*(r)}$ denotes the vector of parameter estimates where at least one parameter has been updated and ζ_k is used for computational convenience. These "E" and "M" steps will be repeated until convergence of the parameter estimates.

After the final parameter estimates $\left(\tilde{\pi},\, \tilde{\beta}_1,\, \tilde{\beta}_2,\, \tilde{\eta_1},\, \tilde{\eta_2}\right)$ are obtained from the EM algorithm, the quantile estimates can be computed by equating the CDF of the Weibull mixture to the desired percentage of the quantile of interest:

$$F\left(x;\, \tilde{\pi},\, \tilde{\beta}_1,\, \tilde{\beta}_2,\, \tilde{\eta_1},\, \tilde{\eta_2}\right) = \sum_{k=1}^{2} \tilde{\pi}_k F\left(x;\, \tilde{\beta}_k,\, \tilde{\eta_k}\right)$$

$$= \tilde{\pi} F\left(x;\, \tilde{\beta}_1,\, \tilde{\eta_1}\right) - (1-\tilde{\pi})\, F\left(x;\, \tilde{\beta}_2,\, \tilde{\eta_2}\right)$$

$$= 1 - \tilde{\pi} e^{-\left(\frac{x}{\tilde{\eta_1}}\right)^{\tilde{\beta_1}}} - (1-\tilde{\pi})\, e^{-\left(\frac{x}{\tilde{\eta_2}}\right)^{\tilde{\beta_2}}}.$$

This equation can be solved numerically using the bi-section method [3]. It should be noted that the above EM algorithm attempts to find the parameter estimates that will maximise (3.3.3), however, the log-likelihood diverges to infinity when $X_1 = \eta_1$ and $\beta_1 \to \infty$. That is, the maximum of (3.3.3) is not well defined. This issue will cause the failure of convergence of the EM algorithm and is also mention in [16]. In order to deal with this problem, the parameter space of β_1 and β_2 will be restricted to [0, 30] as used by [16].

Figure 3.3.1 illustrates the curves of the mixture PDFs, as well as the PDFs for each sub-population. The mixture distributions for both datasets exhibit similar behaviour: the major populations contributes approximately 80%, whereas the minor population is nested in the centre of the data. The minor population fits the left mode while the major population accounts for the right mode. These results agree with the findings of [16].

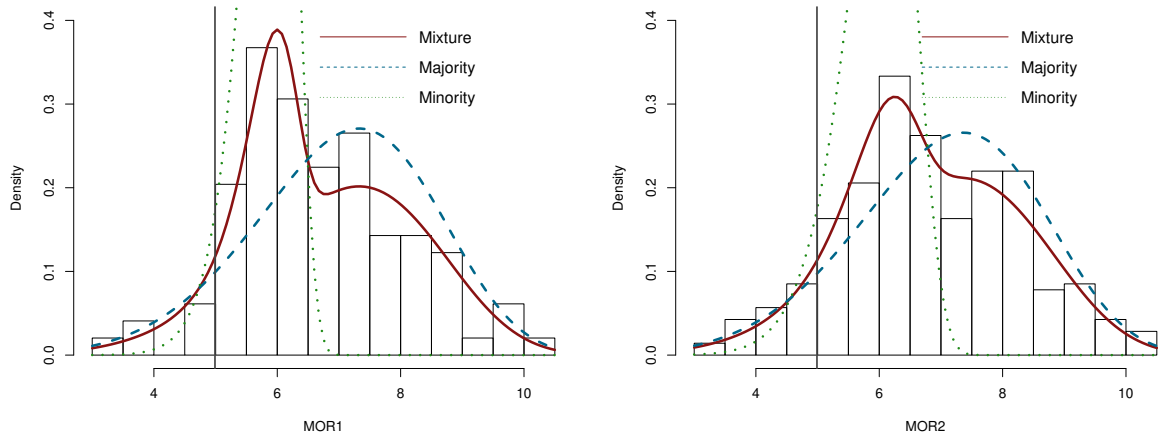


Figure 3.3.1: Weibull mixture models for MOR1 and MOR2.

Algorithm 3.1 EM algorithm for Weibull mixtures.

1. Choose a set of initial parameters $\underline{\theta}^{(r)}$, that is $\pi_1^{(r)}, \ldots, \pi_K^{(r)}, \alpha_1^{(r)}, \ldots, \alpha_K^{(r)}, \ldots, \eta_1^{(r)}, \ldots, \eta_K^{(r)}$, for the K mixing components or sub-populations.
2. E-step:
   (a) Determine the membership weights
   $$\begin{aligned}\gamma_{ik}^{(r+1)} &= E\left(z_{ik}|\boldsymbol{X}, \underline{\theta}^{(r)}\right). \\ &= \frac{\pi_k^{(r)} f\left(\boldsymbol{x}; \alpha_k^{(r)}, \eta_k^{(r)}\right)}{\sum_{j=1}^{2} \pi_j^{(r)} f\left(\boldsymbol{x}; \alpha_j^{(r)}, \eta_j^{(r)}\right)}.\end{aligned}$$
   (b) Determine the expected log-likelihood
   $$Q\left(\underline{\theta}, \underline{\theta}^{*(r)}\right) = \sum_{i=1}^{n}\sum_{k=1}^{2} \gamma_{ik}^{(r+1)} log\left(\left\{\pi_k^{(r)} f\left(x_i; \alpha_k^{(r)}, \eta_k^{(r)}\right)\right\}\right).$$
3. M-step:
   (a) Find $\underline{\theta}^{(r+1)}$ that maximises $Q\left(\underline{\theta}, \underline{\theta}^{(r)}\right)$.
   (b) Update the parameters $\underline{\theta}^{(r)} \leftarrow \underline{\theta}^{(r+1)}$.
4. Repeat steps 2 and 3 until the difference between successive log-likelihoods or parameter estimates becomes significantly small.

3.3.3 Weibull model selection

An important question to ask is whether the Weibull mixture fits the data better (significantly) compared to a single Weibull distribution. The statistical inference required here is whether the observed dataset comes from a homogeneous or heterogeneous population. Figure 3.3.1 motivates the presence of a Weibull distribution contained or nested within the mixture model. This section summarises the various statistical techniques one can make use of to justify the choice of considering a weibull mixture model. In particular, a summary of Section 4.2 in [16] is provided. Readers who are interested in a thorough investigation are referred to [16].

3.3.3.1 The bootstrap homogeneity test

To formulate the question asked above into a statistical nature the hypothesis of interest is given by:

$$\begin{aligned}H_0 &: \boldsymbol{X} = \{X_1, X_2, \ldots, X_n\} \textit{ is from a single Weibull distribution} \\ H_1 &: \boldsymbol{X} \textit{ is from a mixture of two Weibull distributions.}\end{aligned}$$

The most natural choice of test to evaluate the goodness-of-fit for the competing models would be the likelihood ratio test. Unfortunately, the Weibull mixture model violates certain regularity conditions necessary for the asymptotic properties of the likelihood ratio test [16, 15]. In order to circumvent this issue, [16] applies the bootstrap homogeneity test (in the context of Weibull mixtures and not Gaussian mixtures) by [18]. This test procedure involves bootstrapping the likelihood ratio test statistic. The purpose of the

Algorithm 3.2 Parametric bootstrap homogeneity test for estimating the correct number of sub populations K.

1. Obtain the MLE estimates $\left(\hat{\beta}, \hat{\eta}\right)$ and $\hat{\underline{\theta}} = \left(\hat{p}, \hat{\beta}_1, \hat{\eta}_1, \hat{\beta}_2, \hat{\eta}_2\right)$ for the single and mixture Weibull models on the given data set.
2. Calculate the log-likelihood ratio statistic $-2log\left(\lambda\right)$.
3. Simulate a 'bootstrap' data set of size $n = 300$ from a single Weibull$\left(\hat{\beta}, \hat{\eta}\right)$.
4. Fit the the single and mixture Weibull models on the bootstrapped data set and calculate the corresponding bootstrap log-likelihood ratio statistic, $-2log\left(\lambda^*\right)$.
5. Repeat steps 3 and 4, $B = 500$ times to generate the bootstrap sampling distribution of the likelihood ratio statistic.
6. Calculate the bootstrap $p - value$:

$$\frac{1}{B}\sum_{i=1}^{B} I\left\{-2log\left(\lambda^*\right) > -2log\left(\lambda\right)\right\}.$$

bootstrap homogeneity test is to estimate the distribution of the log-likelihood ratio test statistic, from which a $p - value$ can be used to motivate the choice of using a Weibull mixture model over a single univariate Weibull model. The bootstrap procedure is summarised in Algorithm 3.2.

Applying the same procedure, the $p - value$ for the homogeneity test on the MOR1 dataset is 0.034 and 0.015 for MOR2. Both the $p - values$ are smaller than 0.05, which indicate that it is highly unlikely that neither MOR1 nor MOR2 datasets belong to a single Weibull distribution, in comparison to a two component Weibull mixture. Although it is true that the data does not appear to come from a single Weibull distribution, it cannot be said with statistical certainty that the datasets belong to a Weibull mixture. The test does, however, partially support the choice of using a Weibull mixture over a single Weibull distribution.

3.3.3.2 A note on measures based on information criteria

The number of sub-populations or components in a mixture model can also be determined by using penalised likelihood criteria such as the Bayesian information criterion (BIC) or the Akaike information criterion (AIC), where,

$$AIC = 2p - 2l\left(\underline{\theta}\right)$$
$$BIC = log\left(n\right)p - 2l\left(\underline{\theta}\right).$$

The number of parameters in the model is denoted by p and the sample size by n. In general BIC is preferred over AIC given that it penalises the model complexity heavier than AIC. Lower values of BIC or AIC indicates a better model fit.

3.3.4 The censored Weibull mixture model

This subsection aims to provide a thorough revisit of Section 4.3 in [16]. The simulation studies presented in Section 3.4 show that the Weibull mixture is not flexible enough to accommodate data that is typically non-Weibull in nature. In order to improve on this potential flaw and ensure the accuracy of quantile estimates, the idea of subjective or artificial censoring can be applied to the Weibull mixture models. The approach

seems justifiable since the Weibull mixture is more flexible in comparison to the single Weibull distribution. The idea is to achieve the same (if not better) level of accuracy (smaller RMSEs) as the CW-MLE from Subsection 3.2.2. Furthermore, the amount of censoring can be reduced and the quantile estimate should be more efficient due to the increase in sample information usage. Following a similar approach as the CW-MLE in Subsection 3.2.2, the censoring threshold is denoted by C. The censored likelihood of the Weibull mixture with parameters $\underline{\theta} = (\pi, \beta_1, \eta_1, \beta_2, \eta_2)$ is:

$$L(\boldsymbol{X}, \underline{\theta}) = \prod_{i=1}^{n} \left[\pi f(X_i; \beta_1, \eta_1) + (1-\pi) f(X_i; \beta_2, \eta_2)\right]^{\delta_i} \times \left[\pi \{1 - F(C, \beta_1, \eta_1)\} + (1-\pi)\{1 - F(C, \beta_2, \eta_2)\}\right]^{1-\delta_i}, \tag{3.3.11}$$

As in the uncensored Weibull mixture, the maximum likelihood estimate of $\underline{\theta}$ for the censored Weibull mixture can be computed using the EM algorithm. The calculations of the parameter estimates for the censored Weibull mixture using the EM algorithm are provided below:

1. E-step:

   (a) The membership weights, γ_{ik}, are given as in (3.3.5):

$$\gamma_{ik}^{(r+1)} = E\left(z_{ik}|\boldsymbol{X}, \underline{\theta}^{(r)}\right) = \begin{cases} \frac{\pi_k^{(r)} f\left(\boldsymbol{x}; \beta_k^{(r)}, \eta_k^{(r)}\right)}{\sum_{j=1}^{2} \pi_j^{(r)} f\left(\boldsymbol{x}; \beta_j^{(r)}, \eta_j^{(r)}\right)} & X_i \leq C \\ \frac{\pi_k^{(r)}\left(1 - F\left(C; \beta_k^{(r)}, \eta_k^{(r)}\right)\right)}{\sum_{j=1}^{2} \pi_j^{(r)}\left(1 - F\left(C; \beta_j^{(r)}, \eta_j^{(r)}\right)\right)} & X_i > C. \end{cases} \tag{3.3.12}$$

   (b) The expected log-likelihood, $Q_C\left(\underline{\theta}, \underline{\theta}^{*(r)}\right)$, is given as in (3.3.6)

$$Q_C\left(\underline{\theta}, \underline{\theta}^{*(r)}\right) = \sum_{i=1}^{m}\sum_{k=1}^{2} \gamma_{ik}^{(r+1)} log\left(\left\{\pi_k^{(r)} f\left(x_i; \beta_k^{(r)}, \eta_k^{(r)}\right)\right\}\right) + \sum_{i=m}^{n}\sum_{k=1}^{2} \gamma_{ik}^{(r+1)} log\left(\left\{\pi_k^{(r)}\left(1 - F\left(C; \beta_k^{(r)}, \eta_k^{(r)}\right)\right)\right\}\right),$$

   where the lower script C is used to differentiate between the uncensored case.

2. M-step: The parameter estimates that maximise $Q_C\left(\underline{\theta}, \underline{\theta}^{(r)}\right)$ are given by:

   (a)

$$\pi_k^{(r+1)} = \frac{1}{n}\sum_{i=1}^{n} \gamma_{ik}^{(r+1)}.$$

(b)

$$\begin{aligned}
\frac{Q_C\left(\underline{\theta},\underline{\theta}^{(r)}\right)}{\partial\beta_k^{(r)}} &= \frac{\partial\sum_{i=1}^{m}\sum_{k=1}^{2}\gamma_{ik}^{(r+1)}log\left(\left\{\pi_k^{(r)}\left(\frac{\beta_k^{(r+1)}}{\eta_k^{(r)}}\right)\left(\frac{X_i}{\eta_k^{(r)}}\right)^{\beta_k^{(r)}-1}exp\left(-\left(\frac{X_i}{\eta_k^{(r)}}\right)^{\beta_k^{(r)}}\right)\right\}\right)}{\partial\beta_k^{(r)}} \\
&+ \frac{\partial\sum_{i=m}^{n}\sum_{k=1}^{2}\gamma_{ik}^{(r+1)}log\left(\left\{\pi_k^{(r)}\left(exp\left(-\left(\frac{C}{\eta_k^{(r)}}\right)^{\beta_k^{(r)}}\right)\right)\right\}\right)}{\partial\alpha_k^{(r)}} \\
&= \frac{\partial\sum_{i=1}^{m}\sum_{k=1}^{2}\gamma_{ik}^{(r+1)}log\left(\left\{\pi_k^{(r)}\beta_k^{(r)}X_i^{\beta_k^{(r)}-1}\zeta_k^{(r)}exp\left(-\left(X_i^{\beta_k^{(r)}}\zeta_k^{(r)}\right)\right)\right\}\right)}{\partial\beta_k^{(r)}} \\
&+ \frac{\partial\left(n-m\right)\sum_{k=1}^{2}\gamma_{nk}^{(r+1)}log\left(\left\{\pi_k^{(r)}\left(exp\left(-\left(C^{\beta_k^{(r)}}\zeta_k^{(r)}\right)\right)\right)\right\}\right)}{\partial\beta_k^{(r)}} \\
&= \sum_{i=1}^{m}\gamma_{ik}^{(r+1)}\left[\frac{1}{\beta_k^{(r)}}+log\left(x_i\right)-\zeta_k^{(r)}x_i^{\beta_k^{(r)}}log\left(x_i\right)\right] \\
&- \left(n-m\right)\gamma_{nk}^{(r+1)}\zeta_k^{(r)}C^{\beta_k^{(r)}}log\left(C\right)
\end{aligned}$$

$$\therefore\ \frac{1}{\beta_k^{(r+1)}} = \frac{\zeta_k^{(r)}}{\sum_{i=1}^{m}\gamma_{ik}^{(r+1)}}\left[\left(n-m\right)\gamma_{nk}^{(r+1)}C^{\beta_k^{(r)}}log\left(C\right)+\sum_{i=1}^{m}x_i^{\beta_k^{(r)}}log\left(x_i\right)\right]-\frac{\sum_{i=1}^{m}\gamma_{ik}^{(r+1)}log\left(x_i\right)}{\sum_{i=1}^{m}\gamma_{ik}^{(r+1)}}, \tag{3.3.13}$$

where the sample size is n and it is assumed that $X_1, X_2, \ldots, X_m \leq C$ and $X_{m+1}, X_{m+2}, \ldots, X_n \geq C$. It should also be clear that $\gamma_{m+1}^{(r+1)} = \gamma_{m+2}^{(r+1)} = \ldots = \gamma_n^{(r+1)}$ based on the definition of the membership weights given by (3.3.12).

$$\begin{aligned}
\frac{\partial Q\left(\underline{\theta},\underline{\theta}^{(r)}\right)}{\partial\zeta_k^{(r)}} &= \frac{\partial\sum_{i=1}^{m}\sum_{k=1}^{2}\gamma_{ik}^{(r+1)}log\left(\left\{\pi_k^{(r)}\beta_k^{(r)}X_i^{\beta_k^{(r)}-1}\zeta_k^{(r)}exp\left(-\left(X_i^{\beta_k^{(r)}}\zeta_k^{(r)}\right)\right)\right\}\right)}{\partial\zeta_k^{(r)}} \\
&+ \frac{\partial\left(n-m\right)\sum_{k=1}^{2}\gamma_{nk}^{(r+1)}log\left(\left\{\pi_k^{(r)}\left(exp\left(-\left(C^{\beta_k^{(r)}}\zeta_k^{(r)}\right)\right)\right)\right\}\right)}{\partial\zeta_k^{(r)}}
\end{aligned}$$

(c)

$$\therefore\ \frac{1}{\zeta_k^{(r+1)}} = \frac{\sum_{i=1}^{m}\gamma_{ik}^{(r+1)}X_i^{\beta_k^{(r+1)}}+\left(n-m\right)\gamma_{nk}^{(r+1)}C^{\beta_k^{(r+1)}}}{\sum_{i=1}^{n}\gamma_{ik}^{(r+1)}} \tag{3.3.14}$$

$$\therefore\ \eta_k^{(r+1)} = \left(\frac{\sum_{i=1}^{m}\gamma_{ik}^{(r+1)}X_i^{\beta_k^{(r+1)}}+\left(n-m\right)\gamma_{nk}^{(r+1)}C^{\beta_k^{(r+1)}}}{\sum_{i=1}^{m}\gamma_{ik}^{(r+1)}}\right)^{\frac{1}{\beta_k^{(r+1)}}}. \tag{3.3.15}$$

Substituting (3.3.14) into (3.3.13),

$$\frac{1}{\beta_k^{(r+1)}} = \frac{(n-m)\,\gamma_{nk}^{(r+1)} C^{\beta_k^{(r)}} log\,(C) + \sum_{i=1}^{m} x_i^{\beta_k^{(r)}} log\,(x_i)}{\sum_{i=1}^{m} \gamma_{ik}^{(r+1)} x_i^{\beta_k^{(r)}} + (n-m)\,\gamma_{nk}^{(r+1)} C^{\beta_k^{(r)}}} - \frac{\sum_{i=1}^{m} \gamma_{ik}^{(r+1)} log\,(X_i)}{\sum_{i=1}^{m} \gamma_{ik}^{(r+1)}}. \tag{3.3.16}$$

3.3.4.1 A note on selecting the correct censoring threshold

The choice of the correct censoring threshold in the mixture setting is slightly more complex. This approach has not been studied in literature and as mentioned in [16], there is no industry standard that can be used as a viable guideline. A significant issue with the threshold in the mixture model is that it cannot be chosen such that it excludes the majority of a potential modality point in the data. In other words, if the censoring threshold is too low (too far left) that it only includes data from one of the two sub-populations (mixing components) in the estimation procedure. In this case, the EM algorithm will struggle with convergence.

In order to be consistent with [16], the censoring threshold is only considered for C =70^{th} empirical percentile of the data. The censoring threshold cannot be smaller than this because the EM algorithm fails to converge. The failure of convergence results in questionable reliability of quantile estimates and would not be suitable for practice. However, if C is too large (larger than the third quartile), the parameter estimates of the censored Weibull mixture are very close to the uncensored Weibull mixture. It is clear that there is no statistical justification or recommendation of a suitable threshold which paves the way for future work.

From Table 3.3.1, it is true that the parameters of the censored Weibull mixtures are quite similar to the uncensored Weibull mixture models, moreover the censored Weibull mixture approximates and fits the left tail of the real data sets better than the uncensored mixture, see Table 3.3.2 for the truncated Kolmogorov-Smirnov statistics.

3.3.5 Goodness-of-fit for the censored and uncensored Weibull mixture models

In Table 3.3.2, the definition of $D^{TRUNC}\left(\widetilde{F}_n - \hat{F}\,(x)\right)$ remains unchanged, where the goodness-of-fit is still measured up to the 10^{th} empirical percentile. Here, $\widetilde{F}_n$ is the CDF estimate obtained by the uncensored and censored Weibull mixture and $\hat{F}$ the ECDF. Additionally, the $D^{TRUNC}\left(\widetilde{F}_n - \hat{F}\,(x)\right)$ for the CW-MLE from Subsection 3.2.3 is included for reference.

	Model	π	Majority Population		Minority Population	
MOR1	Weibull	0.7674	$\alpha_1 = 5.962$	$\eta_1 = 7.476$	$\alpha_2 = 16.495$	$\eta_2 = 5.949$
MOR2	Weibull	0.7754	$\alpha_1 = 5.327$	$\eta_1 = 7.696$	$\alpha_2 = 11.673$	$\eta_2 = 6.193$

Table 3.3.1: Censored Weibull mixture models for MOR1 and MOR2 using the 70^{th} empirical percentile as the censoring threshold.

From Table 3.3.2, it is clear that the censored Weibull mixture approximates the left tail of the datasets better than the uncensored Weibull mixture and the CW-MLE. The censored Weibull mixture censors much fewer data (30%) compared to the CW-MLE.

	Censored Weibull MLE	Censored Weibull mixture	Uncensored Weibull mixture
MOR1	0.05039	0.0229	0.0778
MOR2	0.0175	0.0124	0.0262

Table 3.3.2: Truncated goodness-of-fit $D^{TRUNC}\left(\widetilde{F}_n - \hat{F}\,(x)\right)$ up to the 10^{th} empirical percentile for the CW-MLE, censored Weibull mixture and uncensored Weibull mixture for the MOR1 and MOR2 datasets.

3.4 Simulation comparison

This section provides a thorough re-work of the simulation results presented in [16]. Furthermore, the section will be divided into two parts; namely a simulation comparison between the proposed univariate models, followed by the comparison of the mixture models. The top performing univariate model will be included in the mixture model simulation study. This design layout is simply to aid the readability and not overwhelm the reader with the large volume of notation in one single sitting.

3.4.1 Model settings used to imitate MOR1 and MOR2

The RMSE is used to compare the performance of the quantile estimates obtained by the CW-MLE, O-MLE, KDE, EMP, the uncensored Weibull mixture (MIX) and the censored Weibull mixture with the threshold chosen to be the 70^{th} empirical percentile (CMIX7). In order to make use of the RMSE measure, the value of the 'true' quantile has to be known. This is attainable if the true underlying distribution of the real dataset is known. Therefore, for each model (true known parametric PDF) used to imitate the MOR1 and MOR2 datasets, $N = 10000$ replicates of sample size $n = 300$ are simulated in order to evaluate the performance of the four quantile estimates using the RMSE. The analysis of the variance and bias of the four estimates is studied in Subsection 3.4.2. The choice of models used to imitate the real MOR datsets is discussed below.

Model settings

The models from which data will be simulated to imitate the MOR1 and MOR2 datasets are classified into three categories, namely parametric, mixture of parametric and non-parametric models.

Parametric models

The parametric models used to simulate the imitating datasets are the models obtained from the censored MLE in Figure 3.2.1. The parameter estimates for the censored models are summarised in Table 3.4.1 below.

Model	MOR1		MOR2	
Weibull	$\hat{\alpha} = 6.823$	$\hat{\eta} = 7.172$	$\hat{\alpha} = 7.378$	$\hat{\eta} = 6.739$
Log-normal	$\hat{\mu} = 2.074$	$\hat{\sigma} = 0.329$	$\hat{\mu} = 1.971$	$\hat{\sigma} = 0.296$
Gamma	$\hat{\theta} = 12.96$	$\hat{\kappa} = 0.599$	$\hat{\theta} = 16.168$	$\hat{\kappa} = 0.440$
Minimum Gumbel	$\hat{a} = 6.623$	$\hat{b} = 0.651$	$\hat{a} = 6.319$	$\hat{b} = 0.601$

Table 3.4.1: Parameter estimates for censored parametric models.

Mixture models

In the simulations provided below, mixture models with only two sub-populations are considered, namely a two-component normal, log-normal and Weibull mixture model. Given that the minimum Gumbel is used to model the distribution of the minimum of a number of samples, a mixture of minimum Gumbels did not provide sufficient flexibility to capture non-tail behaviour and therefore is not considered.

The parameters of the mixture models are summarised in Table 3.4.2 and can be estimated using the EM algorithm.

	Model	π	Majority Population		Minority Population	
MOR1	Normal	0.5631	$\hat{\mu}_1 = 5.956$	$\hat{\sigma}_1 = 0.963$	$\hat{\mu}_2 = 7.652$	$\hat{\sigma}_2 = 1.285$
	Log-normal	0.9758	$\hat{\mu}_2 = 1.897$	$\hat{\sigma}_2 = 0.189$	$\hat{\mu}_1 = 1.246$	$\hat{\sigma}_1 = 0.103$
	Weibull	0.7446	$\hat{\beta}_1 = 5.495$	$\hat{\eta}_1 = 7.599$	$\hat{\beta}_2 = 15.805$	$\hat{\eta}_2 = 5.983$
MOR2	Normal	0.5408	$\hat{\mu}_1 = 5.934$	$\hat{\sigma}_1 = 1.059$	$\hat{\mu}_2 = 7.834$	$\hat{\sigma}_2 = 1.098$
	Log-normal	0.6651	$\hat{\mu}_1 = 1.980$	$\hat{\sigma}_1 = 0.166$	$\hat{\mu}_2 = 1.741$	$\hat{\sigma}_2 = 0.226$
	Weibull	0.7943	$\hat{\beta}_1 = 5.425$	$\hat{\eta}_1 = 7.646$	$\hat{\beta}_2 = 11.992$	$\hat{\eta}_2 = 6.173$

Table 3.4.2: Mixture models' parameter estimates.

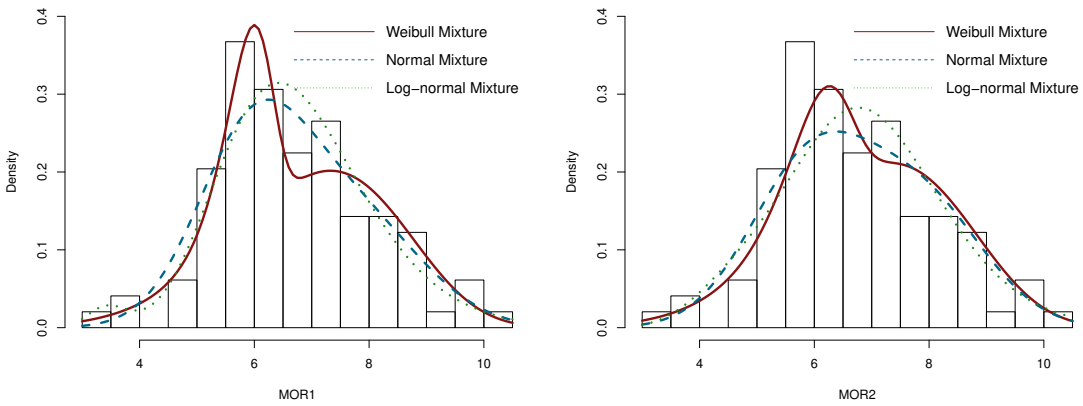


Figure 3.4.1: Mixture models for MOR1 and MOR2.

Non-parametric models

As discussed in Section 2.3, the investigator can never be certain about the validity of the model assumptions implemented on the real datasets. In other the words, the chosen model may not be representative of the of the real data, even if only the lower tail is considered.

To circumvent these issues, the non-parametric KDe can be used to approximate the real dataset (fit a KDe to the dataset) from which the quantile estimates can be obtained. The fitted KDe curves are shown in Figure 2.3.3.

3.4.2 Simulation comparison of the censored, parametric, non-parametric and empirical quantiles

In this subsection, the simulation study is presented to compare the 5^{th} percentile estimates obtained by the censored Weibull (CW-MLE), ordinary Weibull MLE (O-MLE), KDE and the empirical quantile estimate (EMP, Type 9 definition) under a variety of models. In other words, a dataset of size $n = 300$ (size of the MOR2 dataset) is generated from several parametric PDFs (discussed above) in the attempt to imitate the MOR datasets. The CW-MLE, O-MLE, KDE and EMP are fitted to these generated datasets in order to compute and compare the performance of the 5^{th} percentile estimates. The methodology is discussed in detail below. Furthermore, using the industrial standard [1], the censoring threshold C in the CW-MLE is chosen to be the 10^{th} empirical percentile. The procedure is summarised in Algorithm 3.3.

3.4.2.1 RMSE of quantile estimates

The calculation and results of the RMSE for the four quantile estimating techniques (CW-MLE, O-MLE, KDE, EMP) are summarised in Algorithm 3.3 and Tables 3.4.3 and 3.4.4, for the models imitating MOR1 and MOR2 respectively. For ease of readability, the RMSE values are super-imposed with typical heatmap colours, that is, the greener the cell the smaller the RMSE value.

It should be clear that the CW-MLE has the smallest RMSE among the four estimates. The exceptions being the Weibull O-MLE and Minimum Gumbel KDE. However, the performance of the CW-MLE is overall much better in every other model considered. For example, the difference between the RMSE of the Weibull O-MLE and CW-MLE is dwarfed by the difference between the Log-normal and Gamma O-MLE and CW-MLE respectively. In particular, the RMSE of the O-MLE is about 6 times larger than the CW-MLE for data generated from a log-normal distribution and 4 times larger for data generated from a gamma distribution, for MOR1 and MOR2.

Algorithm 3.3 Simulation comparison procedure.

1. Generate a sample of size $n = 300$ from the following models (parameterised by Tables 3.4.1 and 3.4.2) used to imitate the real MOR1 and MOR2 datasets:
   - Weibull
   - Log-normal
   - Gamma
   - Minimum Gumbel
   - two-component normal mixture
   - two-component log-normal mixture
   - two-component Weibull mixture
2. For each model generated sample, fit the CW-MLE, O-MLE, KDE and EMP to the data.
3. Obtain the 5^{th} percentile estimates for the four estimation techniques in Step 2.
4. Repeat Steps 1 - 3 for $N = 10000$ iterations in order to compute the RMSEs for each estimation technique.

Model	O-MLE	CW-MLE	KDE	EMP
Weibull	0.111	0.150	0.171	0.173
Log-normal	1.094	0.166	0.307	0.185
Gamma	0.678	0.163	0.237	0.181
Minimum Gumbel	0.166	0.155	0.146	0.155
Normal Mixture	0.531	0.112	0.164	0.131
Log-normal Mixture	0.571	0.161	0.219	0.195
Weibull Mixture	0.506	0.164	0.198	0.188

Table 3.4.3: RMSE of the 5^{th} quantile estimates for the O-MLE, CW-MLE, KDE and EMP estimation techniques for various models imitating MOR1.

Model	O-MLE	CW-MLE	KDE	EMP
Weibull	0.100	0.135	0.152	0.156
Log-normal	0.944	0.142	0.239	0.157
Gamma	0.626	0.139	0.196	0.157
Minimum Gumbel	0.165	0.154	0.147	0.154
Normal Mixture	0.369	0.126	0.169	0.143
Log-normal Mixture	0.385	0.137	0.158	0.153
Weibull Mixture	0.349	0.166	0.199	0.189

Table 3.4.4: RMSE of the 5^{th} quantile estimates for the O-MLE, CW-MLE, KDE and EMP estimation techniques for various models imitating MOR2.

Model	O-MLE		CW-MLE		KDE		EMP	
Weibull	0.49	[10.06]	0.36	[13.52]	5.93	[13.96]	0.12	[15.61]
Log-normal	92.92	[16.69]	4.59	[13.47]	19.72	[13.53]	0.11	[15.71]
Gamma	61.02	[13.84]	3.46	[13.41]	14.25	[13.54]	0.17	[15.65]
Minimum Gumbel	13.99	[8.85]	4.32	[14.79]	3.89	[14.19]	0.35	[15.39]
Normal Mixture	35.12	[11.32]	1.75	[12.48]	11.37	[12.51]	0.12	[14.31]
Log-normal Mixture	36.29	[12.85]	4.82	[12.90]	8.82	[13.21]	0.19	[15.37]
Weibull Mixture	33.04	[11.29]	2.34	[16.48]	11.05	[16.66]	0.35	[18.98]

Table 3.4.5: Bias and [standard error] (x100) of the O-MLE, CW-MLE, KDE and EMP quantile estimates for the models imitating the MOR2 data set. .

Bias and standard error of the quantile estimates

In order to get a better understanding of the advantages and limitations of the CW-MLE, as well as the bias and standard error of the quantile estimates (summarised in Table 3.4.5) are are revisited studied [16]. The biases and variances for both datasets are very similar so only the models imitating MOR2 have been provided.

From Table 3.4.5 it is clear that the bias of the CW-MLE estimation technique is considerably smaller than the O-MLE for non-Weibull models. Additionally, the standard errors of the CW-MLE estimates are smaller than the EMP estimates. In summary, if the performance of the quantile estimates is ranked according to bias then it follows that EMP $<$ CW-MLE $<$ KDE $<$ O-MLE. [16] achieved the same results. The rank according to standard error is O-MLE $<$ CW-MLE $<$ KDE $<$ EMP, again as documented in [16] as well. The KDE does not approximate the left tail of the data as well as the CW-MLE, in other words, the bias of the KDE is, in general worse than that of the CW-MLE. The KDE does, however, perform rather well on approximating the Minimum Gumbel. The censored Weibull is therefore neither the most accurate nor the most efficient, however, it is the best overall performer.

Upon further investigation, the O-MLE and CW-MLE for data generated from a Weibull distribution, are nearly unbiased, however, the O-MLE is more efficient (smaller standard errors). This should not be surprising since the O-MLE makes use of more information (no censoring). This warrants further investigation.

In order to fully investigate the comparative performance between the CW-MLE and O-MLE for Weibull generated samples, statistical and mathematical reasoning is required to understand the behaviour of the subjective (artificial) censoring methodology. A comparison of the *score function* and asymptotic behavior of the CW-MLE and O-MLE is thoroughly presented in Section 3.5 of [16].

3.4.3 Simulation comparison of the mixture and censored mixture model

In Subsection 3.4.2, the CW-MLE produced the best quantile estimates under most models considered in the simulations. In this subsection, the quantile estimates obtained from the CW-MLE, with the threshold chosen

to be the 10^{th} empirical percentile, are compared to the quantile estimates obtained from the uncensored Weibull mixture (MIX) and the censored Weibull mixture with the threshold chosen to be the 70^{th} empirical percentile (CMIX7). The 5^{th} percentile is still the quantile of interest. The results and conclusions from the models imitating the MOR1 and MOR2 datasets are comparable and therefore only the results for MOR2 are provided. The simulation procedure settings remain the same as in Subsection 3.4.1. That is, the performance of the quantile estimates are obtained using $N = 10000$ replicates from the seven parametric models (with a sample size of $n = 300$) used to imitate the MOR2 dataset, from which the RMSEs are calculated.

3.4.3.1 RMSE, bias and standard error of the quantile estimates

The RMSE of the three quantile estimate techniques for the seven parametric models are summarised in Table 3.4.6. From the background colour of the results provided, the CMIX7 is the best performer. The MIX has the smallest RMSE value when the data is generated from a Weibull distribution and Weibull mixture model. The CMIX7 is the best performer under the Log-normal, Gamma, Minimum Gumbel and Normal mixture model. Taking into consideration the joint performance of the mixture models (MIX and CMIX7), they perform better than the CW-MLE in terms of RMSE, however, neither mixture model performs uniformly better than the CW-MLE.

Furthermore, from Table 3.4.7, the MIX has a large bias under the log-normal and gamma models, which is indicative of its failure to approximate the left tail for those models.

Interestingly, the CMIX7 is generally less biased compared to the other two quantile estimates. The quantile estimates from the CW-MLE and CMIX7 have similar standard errors. In other words, in the presence of similar RMSEs, the CMIX7 provides better quantile estimates compared to the CW-MLE.

Model	CW-MLE	MIX	CMIX7
Weibull	0.138	0.128	0.131
Log-normal	0.139	0.261	0.128
Gamma	0.138	0.191	0.132
Minimum Gumbel	0.155	0.140	0.138
Normal Mixture	0.127	0.133	0.122
Log-normal Mixture	0.137	0.146	0.139
Weibull Mixture	0.163	0.158	0.161

Table 3.4.6: RMSE of the 5^{th} qauntile estimates of the Weibull CW-MLE, MIX and MIX7 for various models imitating MOR2.

Model	CW-MLE		MIX		CMIX7	
Weibull	0.36	[13.52]	1.01	[12.67]	1.33	[13.36]
Log-normal	4.59	[13.47]	13.11	[13.74]	2.29	[13.11]
Gamma	3.46	[13.41]	22.28	[14.08]	1.41	[12.82]
Minimum Gumbel	4.32	[14.79]	0.40	[14.27]	0.18	[13.82]
Normal Mixture	1.75	[12.48]	2.47	[12.95]	1.03	[12.73]
Log-normal Mixture	4.82	[12.90]	3.60	[12.97]	5.6	[12.98]
Weibull Mixture	2.34	[16.48]	0.54	[15.79]	0.27	[15.59]

Table 3.4.7: Bias and [standard error] (x100) of the Weibull CW-MLE, MIX and MIX7 quantile estimates for the models imitating the MOR2 dataset. .

Interestingly, Table 3.4.7 displays patterns which contradict the intuition mentioned earlier, namely, the more data the estimation procedure uses the more efficient the quantile estimate should be. The standard errors of the quantile estimates for the MIX are generally larger than the standard errors of the CW-MLE and CMIX7 under the log-normal, gamma and the normal mixture. A possible rationale could be that the

MIX does not approximate the true underlying model well enough, i.e. it is not flexible enough. The next chapter will focus on selecting an optimal threshold for the censored CW-MLE.

3.5 Chapter summary

Chapter 3 addresses the challenges of parametric and non-parametric quantile estimation presented in Chapter 2. In particular, the idea of artificial censoring in conjunction with a censored-adjusted semi-parametric modeling approach is used (via censored MLE) to focus on the estimation of the lower tail of a dataset. The Weibull distribution is used as the parametric lower tail. The CW-MLE approach proves to perform the best among the O-MLE, EMP and KDE quantile estimation techniques. A simulation study is conducted where it is shown that for all parametric PDFs used to imitate the MOR datasets, the CW-MLE provides almost unbiased estimates for lower quantiles and has the same (if not better) performance compared to non-parametric estimates. The CW-MLE approach censors 90% of the information available in data and is therefore not completely efficient in terms of information usage. Section 3.3 addresses the issue of inefficient data usage of the CW-MLE quantile estimation technique by proposing censored and uncensored Weibull mixture models. The mathematical derivation of both mixture models, as well as the expressions required for parameter estimation in the EM algorithm, are derived. Finally, a simulation comparison is provided to evaluate the performance of the quantile estimates obtained from the censored and uncensored Weibull mixture models. The uncensored Weibull mixture, MIX, performs adequately, however it struggles to approximate the left tails of several of the parametric models considered. The censored Weibull mixture model, CMIX7, performs the best and outperforms the CW-MLE. The choice of threshold in the CMIX7 is extremely experimental and requires further investigation. Moreover, a thorough threshold selection study is required to select the optimal threshold for the CW-MLE for a fair comparison of the CMIX7 technique. This threshold selection study for the CW-MLE is proposed in Chapter 4.

Chapter 4

Threshold Selection Techniques in the Censored Weibull Model

Chapter highlights

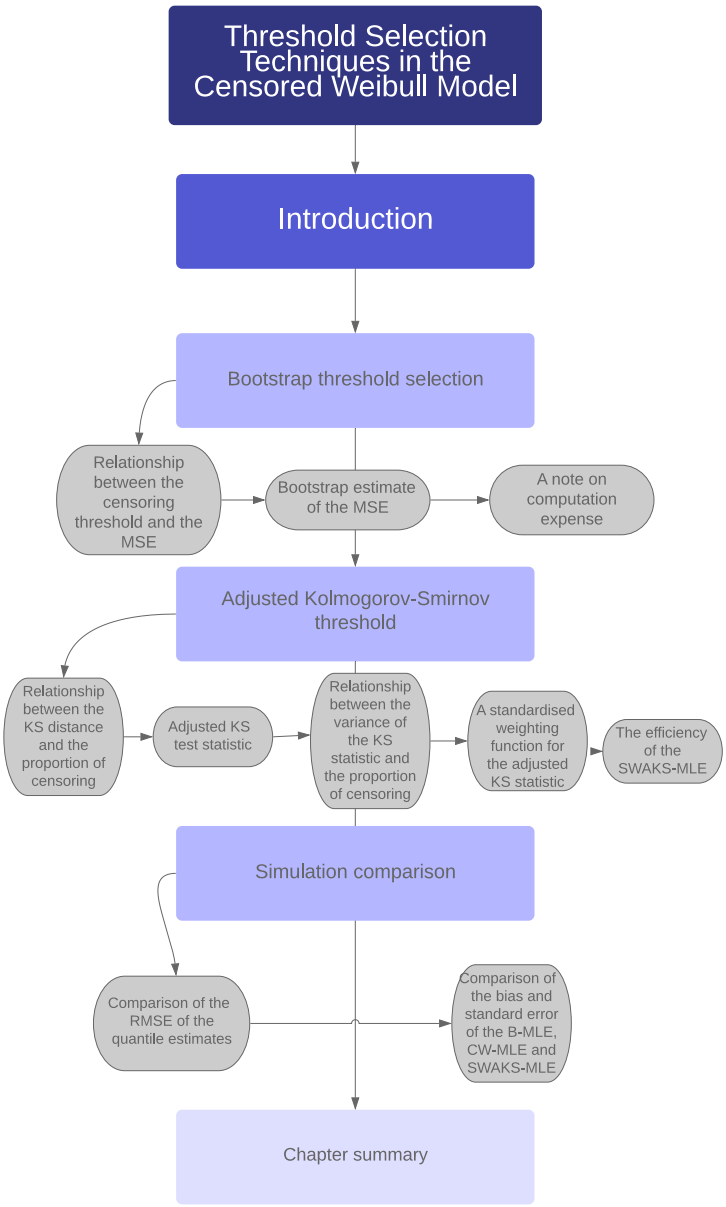
The highlights for Chapter 4 are:

- Introduction to the bootstrap threshold censored Weibull MLE (B-MLE).
- Estimating the MSE using a bootstrap approach.
- Introduction to the standardised-weighted adjusted Kolmogorov-Smirnov test statistic (SWAKS).
- Simulation comparative study of the threshold selection techniques.

4.1 Introduction

In Chapter 3 the censoring threshold C in the CW-MLE is fixed at the 10^{th} empirical percentile based on industrial standards [16, 1]. The first half of this chapter provides a rigorous re-work of Chapter 5 in [16]. That is, the relationship between the censoring threshold and MSE of the quantile estimate obtained from the CW-MLE is initially studied. To obtain an optimal threshold, the MSE is estimated using a bootstrap approach in Section 4.2 and the best threshold is chosen such that the estimated MSE is a minimum. According to simulations completed in [16], the *bootstrap threshold censored Weibull MLE* (B-MLE) performs better than the original CW-MLE. In the remainder of the chapter, namely Section 4.3, a new threshold selection technique is proposed that makes use of an *adjusted truncated Kolmogorov-Smirnov test*. Simulation studies clearly indicate that the *adjusted truncated Kolmogorov-Smirnov censored Weibull MLE* (SWAKS-MLE) outperforms the B-MLE. Finally, a simulation study to compare the bootstrap and SWAKS quantile estimation techniques is provided in Section 4.4.

Chapter outline



4.2 Bootstrap threshold selection

This section introduces the methodology and motivation behind the B-MLE proposed by [16].

4.2.1 Relationship between the censoring threshold and the MSE

As discussed in Chapter 3, the censoring threshold in the CW-MLE is fixed at the 10^{th} empirical percentile. If the censoring threshold C is increased, more data will be used in the estimation procedure, so in terms of information usage the quantile estimates should be more efficient. However, an increase in C might impact the goodness-of-fit of the Weibull distribution on the left tail, if the data is non-Weibull in nature, leading to

a biased quantile estimate. Recall that the MSE is a function of the bias and the variance and the relationship of these components can be studied using Monte Carlo methods, given that the underlying distribution is known (this is why the seven parametric models used to imitate the MOR datasets were introduces in Chapter 2).

Figure 4.2.1, summarises the relationship between the MSE and censoring the threshold (chosen as the empirical percentiles). The theoretical MSE values are evaluated as follows: a sample size of $n = 300$ is generated from the seven parametric models (model settings in Chapter 3) imitating the real MOR2 dataset, from which the MSEs are obtained by fitting the CW-MLE to each sample. This process is repeated $N = 10000$ times for a range of censoring threshold candidates. The threshold candidates range from the 10^{th} to the 100^{th} (no censoring present) empirical percentile in steps of 5%. The curves for MOR1 are very similar and have therefore been omitted.

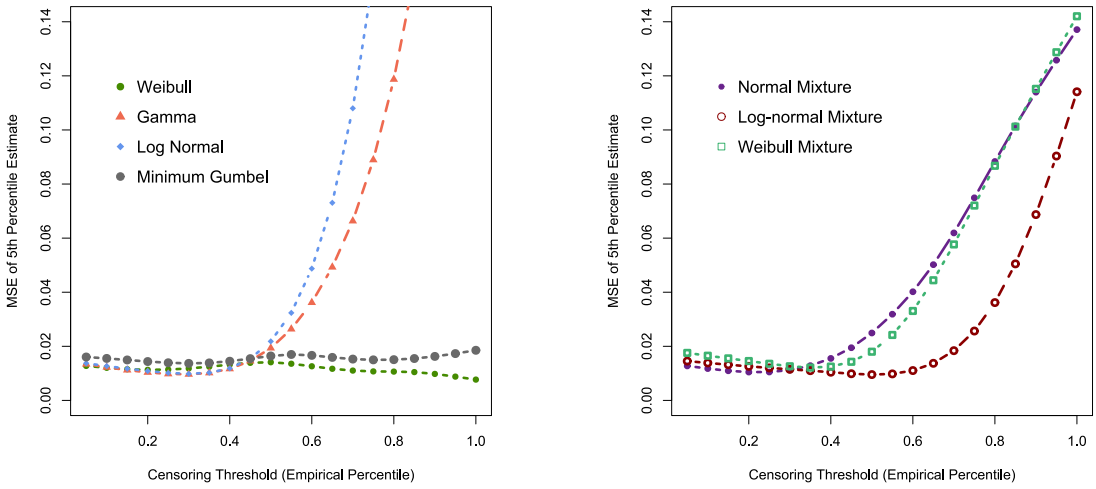


Figure 4.2.1: Relationship between the censoring threshold and MSE for MOR2 given that the underlying model is known.

Upon closer inspection it is evident that under the Weibull model, the MSE decreases as the threshold increases. In other words, it is true that the CW-MLE is almost unbiased (see Chapter 3) under all thresholds. More importantly, the variance of the quantile estimate decreases as the threshold increases. This should not be surprising since the CW-MLE approaches the O-MLE as the amount of censoring decreases (i.e. the correct model is fitted). For the remainder of the parametric models, the MSE initially decreases and then increases quite drastically for larger thresholds. Clearly, for smaller thresholds, the bias increases very little compared to the decrease in variance, leading to a decrease in MSE values. However, when the threshold becomes sufficiently large, for example, larger than the median, then the increase in bias completely overshadows the continuous marginal decrease in variance, leading to an increase in MSE. This is indicative that as the censoring threshold increases, the CW-MLE does not fit the data adequately. The optimal censoring threshold (the threshold associated with the smallest MSE) for most of the parametric models considered is arguably between the 30^{th} and 50^{th} percentile. These results, unsurprisingly, agree with those of [16].

4.2.2 Bootstrap estimate of MSE

In the simulations above, the relationship between the MSE and the censoring threshold is easy to identify and study since the true parametric PDF is known. However, in practice, this will hardly (if ever) be the case. The ECDF of the real dataset is ,however, a consistent estimator of the true underlying distribution

[20]. The MSE can be estimated by making use of the bootstrapping technique. Recall that the MSE under the true model G is given by (2.2.1)

$$E_G\left(\widetilde{q}_n\left(C\right)-q\right)^2$$

which can be estimated by the bootstrap MSE

$$E_{G_n}\left(\widetilde{q}_n^*\left(C\right)-\hat{q}_n\right)^2.$$

Where G_n is the ECDF of a sample from the true model G and $\hat{q}_n$ the empirical quantile estimate (Type 9 definition in R). $\widetilde{q}_n\left(C\right)$ is the quantile estimate obtained from the CW-MLE with threshold C and $\widetilde{q}_n^*\left(C\right)$ is the CW-MLE quantile estimate of the bootstrap sample. Take note that G can be completely unspecified.

For $E_{G_n}\left(\widetilde{q}_n^*\left(C\right)-\hat{q}_n\right)^2$ to be a consistent estimator of $E_G\left(\widetilde{q}_n\left(C\right)-q\right)^2$, several regularity conditions need to hold. These conditions along with a work-in-progress proof can be found in Chapter 5 of [16]. For completeness the conditions are provided below,

$$E_{G_n}\left(\widetilde{q}_n^*\left(C\right)-\hat{q}_n\right)^2 \xrightarrow{p} E_G\left(\widetilde{q}_n\left(C\right)-q\right)^2,\ n\to\infty. \tag{4.2.1}$$

Furthermore,

$$\lim_{n\to\infty} Pr\left\{E_{G_n}\left(\widetilde{q}_n^*\left(C_1\right)-\hat{q}_n\right)^2 > E_{G_n}\left(\widetilde{q}_n^*\left(C_2\right)-\hat{q}_n\right)^2\right\}=1,$$

when

$$E_G\left(\widetilde{q}_n\left(C_1\right)-q\right)^2 > E_G\left(\widetilde{q}_n\left(C_2\right)-q\right)^2,$$

for any C_1 and C_2. This result implies that the best threshold should be chosen asymptotically from a finite set of thresholds, such that the chosen threshold is associated with the smallest MSE of the desired quantile estimate. The approach here would be that the probability of choosing the best threshold from a finite sample with a relatively large sample size, should be greater than the probability of choosing the incorrect threshold.

The next section will discuss the computation of the bootstrap estimate of the MSE, followed by simulations on the consistency of the bootstrap MSE. This work is not novel and is simply a re-work of the concepts presented in [16].

Computing the bootstrap MSE

In order to compute $E_{G_n}\left(\widetilde{q}_n^*\left(C_1\right)-\hat{q}_n\right)^2$ for an *i.i.d.* sample $\boldsymbol{X}=\{X_1, X_2, \ldots, X_n\}$, it is first necessary to obtain the Type 9 empirical quantile (see Chapter 3) as $\hat{q}_n$ from $\boldsymbol{X}$. Next, re-sampling with replacement from $\boldsymbol{X}$ is performed in order to obtain the bootstrap sample $\boldsymbol{X}^*=\left\{X_1^*, X_2^*, \ldots, X_n^*\right\}$. Next, Type II subjective censoring (see Chapter 2) is applied to the bootstrapped sample, i.e., calculate the CW-MLE on the smallest r observations. Where r is the number of observations smaller than or equal to the censoring threshold C obtained using the original sample $\boldsymbol{X}$. The quantile estimate of the bootstrap sample is obtained as $\widetilde{q}_{\boldsymbol{X}^*}^*$, due to the parameters being estimated from the CW-MLE. The calculation of $\widetilde{q}_{\boldsymbol{X}_i^*}^*$ will be repeated for B (e.g. 5000) times, where the sub-script i indicates the i-th bootstrap sample. Finally $E_{G_n}\left(\widetilde{q}_n^*\left(C_1\right)-\hat{q}_n\right)^2$ is calculated as follows

$$E_{G_n}\left(\widetilde{q}_n^*\left(C_1\right)-\hat{q}_n\right)^2=\frac{1}{B}\sum_{i=1}^{B}\left(\widetilde{q}_{\boldsymbol{X}_i^*}^*-\hat{q}_n\right).$$

For ease of readability, the procedure described above is summarised in Algorithm 4.1. This procedure is repeated for a range of thresholds C. The best threshold is chosen such that it is associated with the smallest bootstrap MSE.

In the study proposed by [16], only thresholds between the 10^{th} empirical percentile and the median are considered (in steps of 10%) when calculating the lower quantile estimates. This is purely for computational simplicity as the above procedure is rather computationally intensive (discussed in Subsection 4.2.3).

Algorithm 4.1 Bootstrap MSE estimation.

1. Calculate the Type 9 empirical quantile $\hat{q}_n$ from the original sample $\boldsymbol{X}$.
2. Re-sample from $\boldsymbol{X}$ with replacement to obtain the bootstrap sample $\boldsymbol{X}^*$.
3. Apply Type II subjective censoring to $\boldsymbol{X}^*$ and calculate the CW-MLE on the smallest r observations.
4. Obtain the quantile estimate of the bootstrap sample $\widetilde{q}^*_{\boldsymbol{X}^*}$.
5. Repeat Steps $2-4$, for $B = 5000$ repetitions.
6. Calculate $E_{G_n}\left(\widetilde{q}^*_n(C) - \hat{q}_n\right)^2 = \frac{1}{B}\sum_{i=1}^{B}\left(\widetilde{q}^*_{\boldsymbol{X}^*_i} - \hat{q}_n\right)^2$.
7. Repeat Steps $5-6$ for a range of threshold candidates C for $\boldsymbol{X}$.
8. Select the threshold with the smallest bootstrap MSE.

Discussion on the type of censoring and number of bootstrap replicates

Type of censoring

The censoring threshold for the CW-MLE in Chapter 3 is fixed at the 10^{th} empirical percentile. In other words, 90% of the available data is censored in order to focus on the fit of the left tail. In this way, the censoring could be treated as either Type I censoring, where the threshold is chosen to be the sample 10^{th} empirical percentile, or Type II censoring where only 10% of the smallest observations are considered.

The choice of Type I or Type II censoring will affect the quantile estimate in the bootstrap approach and therefore the MSE estimate in the bootstrap sample. The reason for this is that the censoring threshold is selected from the original sample $\boldsymbol{X}$ and the smallest r observations from the bootstrap sample $\boldsymbol{X}^*$.

According to [16], the performance of the RMSE of the B-MLE quantile estimates for the Type II censoring is marginally better. Therefore, the Type II censoring will be reported for this study. The results for Type I should be requested from the original author.

Number of bootstrap replicates

The number of replicates B will have an impact on the precision of the MSE estimate. In other words, the larger the value of B, the better the precision of the MSE estimate will be, at the expense of computation time. The rationale for this value is explained in detail in [16].

Evaluating the consistency of the bootstrap MSE estimate using simulation

To ensure that the B-MLE performs as desired (asymptotically), is equivalent to showing that (4.2.1) is satisfied. This task is rather involved and time-consuming. For the sake of brevity, interested readers are referred to [16] for a work-in-progress solution.

The approach provided below evaluates the consistency of the bootstrap MSE by means of a simulation study, re-worked from Chapter 5 in [16]. The bootstrap MSE is a consistent estimator of the true MSE if the distribution of the difference between the true MSE and bootstrap MSE becomes degenerate as the sample size increases indefinitely (convergence in probability [2]). The simulation procedure settings remain the same as in Chapter 3. That is, using $N = 10000$ replicates from the seven parametric models used to imitate the MOR datasets, from which the MSEs are calculated. The only difference here, is that the sample sizes are varied ($n = 300,\ 500,\ 1000,\ 2000,\ 3000,\ 5000$).

For each repetition of each sample size, the quantile estimate $\widetilde{q}^i_n$ is calculated from the CW-MLE, using the 10^{th} empirical percentile as the threshold. The superscript indicates the i^{th} replicate and the subscript

the sample size n. The population MSE is estimated using these quantile estimates, for sample size n, as M_n (see Chapter 2):

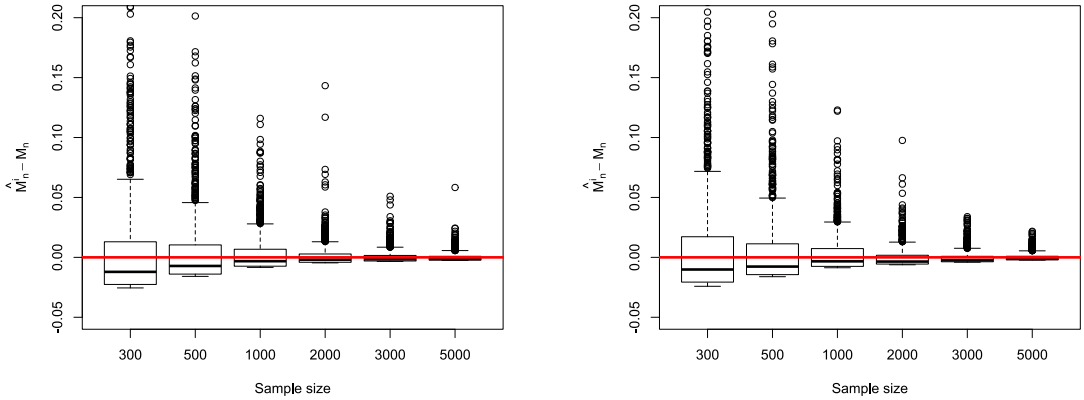
$$M_n = \frac{1}{N}\sum_{i=1}^{N}\left(\widetilde{q}_n^i - q\right)^2. \tag{4.2.2}$$

Where q is the true population quantile from the model used to imitate the real MOR data. Additionally, for each repetition, the bootstrap procedure introduced earlier is also applied to obtain the MSE estimate $\widehat{M}_n^i$, with $B = 5000$ repetitions. Finally, the Monte Carlo estimate of the distribution of the difference between the population MSE and bootstrap MSE, $\widehat{M}_n^i - M_n$, is obtained. For convenience, the procedure is provided in Algorithm 4.2.

Algorithm 4.2 Obtaining the Monte Carlo estimate of the distribution of the difference between the population MSE and bootstrap MSE.

1. Simulate data from a model that replicates the MOR dataset.
   - Population MSE estimate components:
   (a) Fit the censored Weibull MLE to data obtained in (1), using the 10^{th} empirical percentile as the threshold C.
   (b) Estimate the desired quantile using the MLE estimates obtained in (a).
   - Bootstrap MSE estimate components:
   (a) Obtain the Type IX empirical quantile estimate from the data obtained in (1) as $\hat{q}_n$.
   (b) Re-sample from the data obtained in (1) (with replacement), to obtain $\boldsymbol{X}^*$.
   (c) Fit the censored Weibull MLE to $\boldsymbol{X}^*$, using the 10^{th} empirical percentile as the threshold C. Note C is obtained from $\boldsymbol{X}^*$ (Type II likelihood).
   (d) Estimate the desired quantile using the MLE estimates obtained in (c).
   (e) Repeat steps (b) - (d) B times to obtain
   $$\widehat{M}_n^i = \frac{1}{B}\sum_{i=1}^{B}\left(\widetilde{q}^*_{\boldsymbol{X}_i^*} - \hat{q}_n\right)^2.$$
2. Repeat step(s) 1, N times in order to obtain the population Monte Carlo MSE estimate as in (4.2.2).
3. Obtain the Monte Carlo distribution of the difference between the population MSE and bootstrap MSE, $\widehat{M}_n^i - M_n$.
4. Repeat Steps $1-3$ for sample sizes, n, given by $(300,\ 500,\ 1000,\ 2000,\ 3000)$.

The simulation study outlined above is repeated for all seven parametric models introduced in Chapters 2 and 3, in particular, see Step 1 of Algorithm 3.3. The distribution of $\widehat{M}_n^i - M_n$, for all values of n, exhibits the same behaviour for all seven parametric models considered. For simplicity, only the results for datasets generated from a Weibull and Weibull mixture distribution, imitating the MOR2 dataset, are presented. The box-plot of the Monte Carlo distribution of $\widehat{M}_n^i - M_n$ for various sample sizes is provided in Figure 4.2.2.



(a) Weibull

(b) Weibull mixture

Figure 4.2.2: Distribution of $\widehat{M}_n^i - M_n$ for different sample sizes.

From Figure 4.2.2, it is clear that the distribution of $\widehat{M}_n^i - M_n$ is concentrated around zero. Furthermore, the right tail of said distribution becomes shorter as the sample size n increases. That is, the difference between the true MSE and bootstrap MSE becomes degenerate as the sample size increases.

Model	Statistic	300	500	1000	2000	3000	5000
Weibull	mean	0.0909	0.0807	0.0477	0.0419	0.0358	0.0306
	standard error	0.2539	0.1479	0.1285	0.0862	0.0739	0.0655
Weibull mixture	mean	0.1175	0.0993	0.0868	0.0320	0.0237	0.0065
	standard error	0.3983	0.2154	0.1882	0.1538	0.1245	0.1071

Table 4.2.1: $\sqrt{n}$ times the standard error and mean of $\widehat{M}_n^i - M_n$ for different sample sizes.

To further study the convergence of $\widehat{M}_n^i - M_n$, the $\sqrt{n}$ times the standard error and the mean are provided in Table 4.2.1. Notice that the mean and standard error decrease as the sample sizes increases. These trends further support the idea of the consistency of the bootstrap MSE. These results are not novel and agree with those presented in [16]. For ease of readability, the performance of the B-MLE is presented in Section 4.4 in order to keep the comparisons of all techniques studied together.

4.2.3 A note on computation expense

The B-MLE method proposed by [16] is computationally expensive. Algorithm 4.3 summarises the quantile estimation procedure for the B-MLE technique. If the range of threshold candidates is selected to be $C \in [0.1, 1]$ in steps of 0.01, the total number of quantiles that need to be estimated for a single model that imitates the MOR dataset is

$$\begin{aligned} \textit{Total number of } 5^{th} \textit{ quantiles estimated} &= 10\,000 \times 5000 \times 91 \\ &= 4\,550\,000\,000. \end{aligned}$$

The total number of iterations and quantile estimates is extremely large and computationally expensive. The number of quantile estimates could be lowered, at the expense of the precision of the MSE. [16] only consider the 10^{th}, 20^{th}, 30^{th}, 40^{th} and 50^{th} percentiles as threshold candidates. Unfortunately, the limited selection of threshold candidates may not be sufficient to obtain the best performing threshold for each model that imitates the MOR datasets. Furthermore, applying the B-MLE in a more computationally demanding

Algorithm 4.3 Simulation procedure to evaluate the performance of the B-MLE quantile estimation technique.

1. Generate a sample $\boldsymbol{X}$ of size $n = 300$ from the following models (parameterised by Tables 3.4.1 and 3.4.2) used to imitate the real MOR1 and MOR2 datasets:
   - Weibull
   - Log-normal
   - Gamma
   - Minimum Gumbel
   - two-component normal mixture
   - two-component log-normal mixture
   - two-component Weibull mixture
2. Calculate the Type 9 empirical quantile $\hat{q}_n$ from the original sample $\boldsymbol{X}$.

   —**Bootstrap procedure**

   (a) Re-sample from $\boldsymbol{X}$ with replacement to obtain the bootstrap sample $\boldsymbol{X}^*$.

   (b) Apply Type II subjective censoring to $\boldsymbol{X}^*$ and calculate the CW-MLE on the smallest r observations.

   (c) Obtain the quantile estimate of the bootstrap sample $\widetilde{q}^*_{\boldsymbol{X}^*}$.

   (d) Repeat Steps (a)-(b), for $B = 5000$ times.

   (e) Calculate $E_{G_n}\left(\widetilde{q}^*_n(C) - \hat{q}_n\right)^2 = \frac{1}{B}\sum_{i=1}^{B}\left(\widetilde{q}^*_{\boldsymbol{X}^*_i} - \hat{q}_n\right)^2$.

   (f) Repeat Steps (d)-(e) for a range of threshold candidates C_i for $\boldsymbol{X}$.

   (g) Select the threshold C^*_i with the smallest bootstrap MSE.
3. Apply the CW-MLE technique to $\boldsymbol{X}$ to obtain the 5^{th} quantile estimate using C^*_i as the censoring threshold.
4. Repeat Steps 1 - 3 for $N = 10000$ iterations in order to compute the RMSEs for the B-MLE quantile estimation technique.

setting, such as the censored mixture model where the EM algorithm is required for parameter estimation, may become completely infeasible. Clearly, a new threshold selection technique is required that performs at least as well as the B-MLE, in addition to being vastly less computationally intensive.

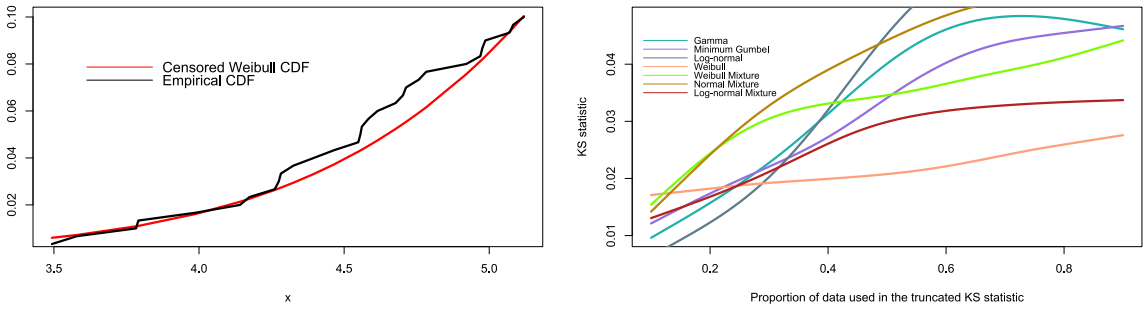
4.3 Adjusted Kolmogorov-Smirnov threshold

This section introduces the methodology and motivation behind the adjusted KS threshold selection technique for the CW-MLE. The objective here is still to find the optimal threshold for a given dataset, however, the computational intensity and methodology complexity should be vastly reduced. The idea is quite simple; select the threshold that minimises an adjusted KS (discussed later) distance between the ECDF and the fitted model CDF, in this case, the censored Weibull.

4.3.1 Relationship between the KS distance and the proportion of censoring

The nature of the KS statistic is such that the distances between the ECDF and the CDF of interest are usually smallest in the left and right tail of the curves. In other words, any indication of a lacklustre performance in goodness-of-fit will usually only be detected between the *extended interquartile range*[1]. Here the extended interquartile range refers to the support of the CDFs between the 10^{th} and 95^{th} percentile. This implies that the truncated KS test statistic (3.2.4), tends to be a minimum for smaller thresholds. Here $\widetilde{F}_n(x)$ is given by $\ddot{\ddot{F}}^i_{(r)}$, the CDF of the censored Weibull fitted to the data from model i (one of the seven parametric models) imitating the MOR dataset and $\hat{F}_{(r)}$ the ECDF. The subscript (r) indicates that the distances are only considered up to the r^{th} order statistic, i.e. the smallest r observations below the cut-off threshold C.

The truncated KS test statistic is an increasing function of the threshold. This should not be surprising because as mentioned earlier, an increase in the censoring threshold might impact the goodness-of-fit of the Weibull distribution on the left tail if the data is non-Weibull in nature. From Figure 4.3.1 it is clear that the distance between the fitted censored Weibull CDF and ECDF is smaller in left tails of the CDF curves. Moreover, according to Figure 4.2.1, the curve in Figure 4.3.1 for data generated from a Weibull distribution (pink line) should not have a positive slope. Clearly the standard truncated KS test statistic cannot be used as a threshold selection technique.



(a) CW-MLE CDF against ECDF for Gamma.

(b) $D^{TRUNC}\left(\ddot{\ddot{F}}^i_{(r)} - \hat{F}_{(r)}\right)$ for various thresholds.

Figure 4.3.1: Censored Weibull CDF against the ECDF for a Gamma model (a) and Kolmogorov-Smirnov (KS) test statistic $D^{TRUNC}\left(\ddot{\ddot{F}}^i_{(r)} - \hat{F}_{(r)}\right)$ for all models imitating MOR2 for various threshold candidates (b).

4.3.2 Adjusted KS test statistic

The distance between the ECDF and censored Weibull CDF fitted to the data from a model imitating the MOR dataset is extremely small between the left tails of the curves. If a threshold were to be chosen such that the KS distance is minimised, then the smallest available threshold (in this case the 10^{th} empirical percentile) would be almost surely selected. The boxplot in Figure 4.3.2 summarises the distribution of the selected threshold, for each parametric model imitating MOR2, using the truncated KS test statistic based on $N = 10000$ iterations. Clearly, all boxplots are concentrated at the 10^{th} empirical percentile. This is not ideal, after all, Figure 4.2.1 suggests that the ideal threshold for most of the models considered is arguably between the 30^{th} and 50^{th} percentile. The boxplot for MOR1 is very similar and is therefore not included here.

[1] The interquartile range here does not refer to the descriptive statistic, but instead to the support of the distribution.

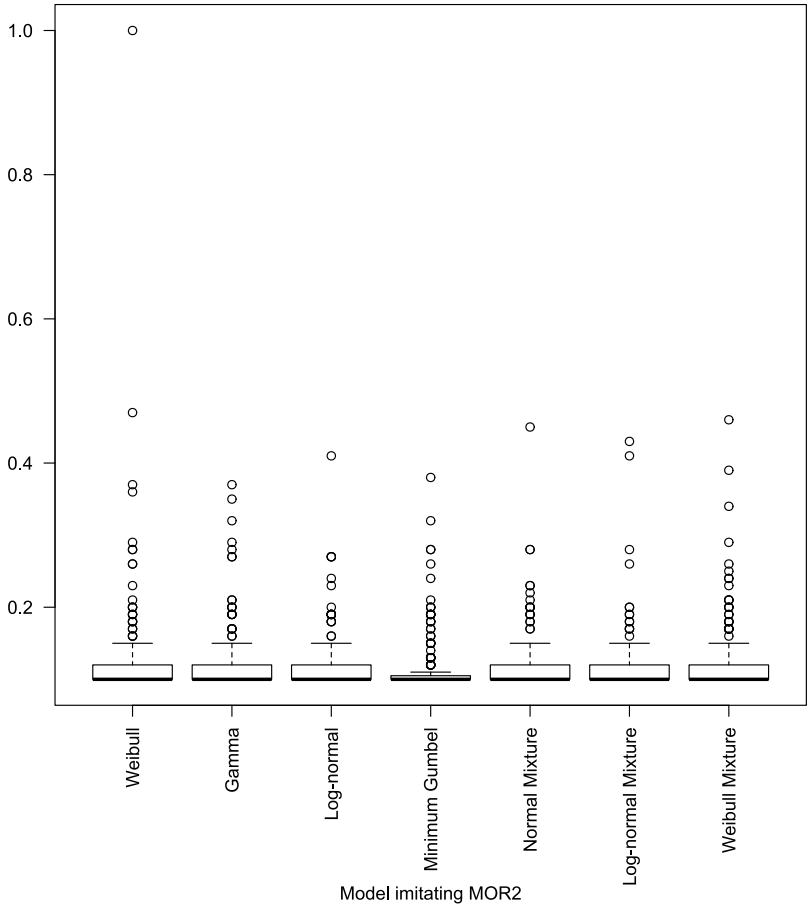


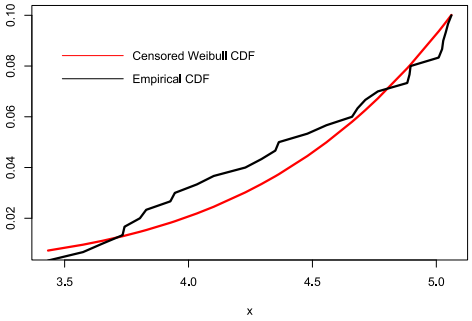
Figure 4.3.2: Boxplot of selected threshold using the truncated KS test statistic $D^{TRUNC}\left(\ddot{\ddot{F}}^{i}_{(r)} - \hat{F}_{(r)}\right)$.

In order to circumvent this issue, the truncated KS test statistic can be adjusted to

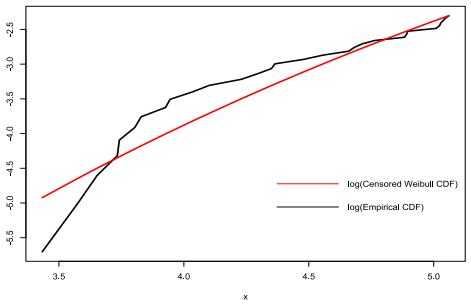
$$D^{AKS}\left(\ddot{\ddot{F}}_{(r)} - \hat{F}_{(r)}\right) = max\left\{\left|log\left(\ddot{\ddot{F}}_{(r)}\right) - log\left(\hat{F}_{(r)}\right)\right|\right\}. \tag{4.3.1}$$

The natural logarithm maps the differences between the fitted censored CDF and ECDF from the space [0, 1] to $(-\infty, 0]$. The mapping exaggerates differences in the left tail between the CDF curves.

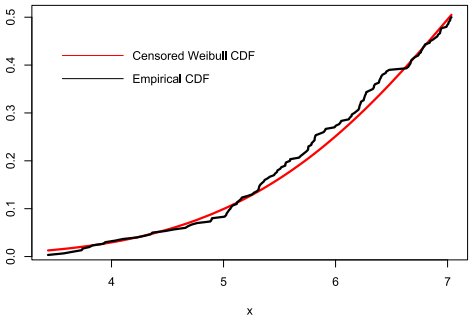
To illustrate this concept, a sample of $n = 300$ is generated from a gamma(16.168, 0.440) distribution (used to imitate the MOR2 dataset) and the corresponding $\ddot{\ddot{F}}_{(r)}$, $\hat{F}_{(r)}$, $log\left(\ddot{\ddot{F}}_{(r)}\right)$ and $log\left(\hat{F}_{(r)}\right)$ are plotted in Figure 4.3.3 for various thresholds. Figure 4.3.3 illustrates how the log transform exaggerates the distance in the left tails from $\left|\ddot{\ddot{F}}_{(r)} - \hat{F}_{(r)}\right|$ to $\left|log\left(\ddot{\ddot{F}}_{(r)}\right) - log\left(\hat{F}_{(r)}\right)\right|$. Figure 4.3.4 plots the distances between $\ddot{\ddot{F}}_{(r)}$ and $\hat{F}_{(r)}$, as well as $log\left(\ddot{\ddot{F}}_{(r)}\right)$ and $log\left(\hat{F}_{(r)}\right)$ respectively. From Figure 4.3.4, it should be clear, by looking at the scale on the $y-axes$, how the natural logarithmic transformation impacts the distances in the left tails between the curves. Furthermore, as the number of data points included in the estimation process increases (i.e. a larger censoring threshold), the log distances in the left tail increase substantially. That is, as the censoring threshold increases, the ECDF smooths out and the fitted censored Weibull CDF re-adjusts



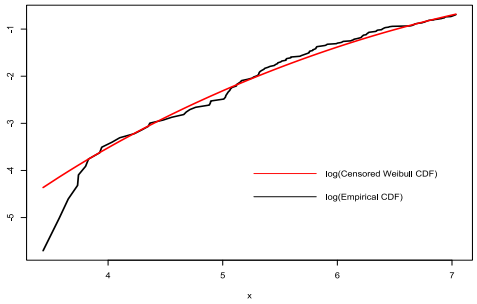
(a) $\ddot{\ddot{F}}_{(r)}$ and $\hat{F}_{(r)}$ for $p = 0.1$



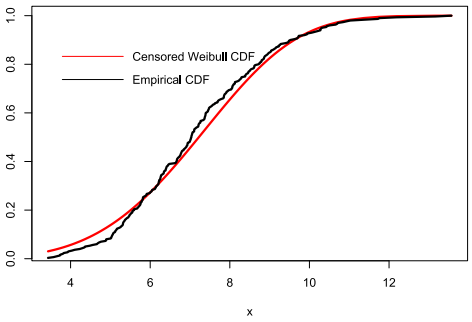
(b) $log\left(\ddot{\ddot{F}}_{(r)}\right)$ and $log\left(\hat{F}_{(r)}\right)$ for $p = 0.1$



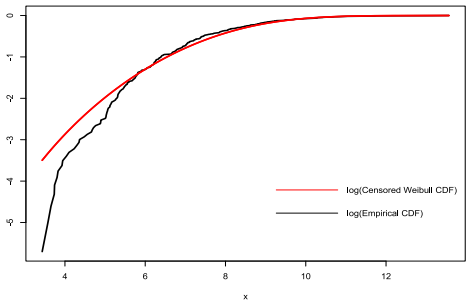
(c) $\ddot{\ddot{F}}_{(r)}$ and $\hat{F}_{(r)}$ for $p = 0.5$



(d) $log\left(\ddot{\ddot{F}}_{(r)}\right)$ and $log\left(\hat{F}_{(r)}\right)$ for $p = 0.5$

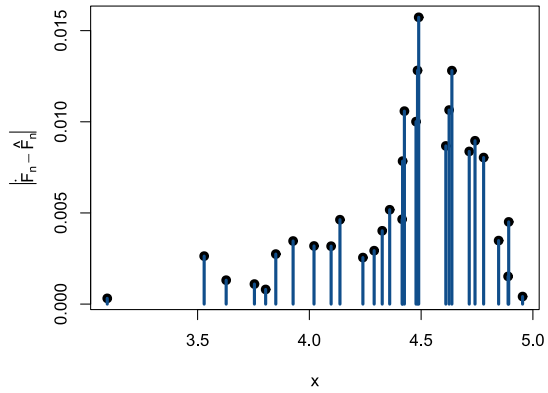


(e) $\ddot{\ddot{F}}_{(r)}$ and $\hat{F}_{(r)}$ for $p = 1$

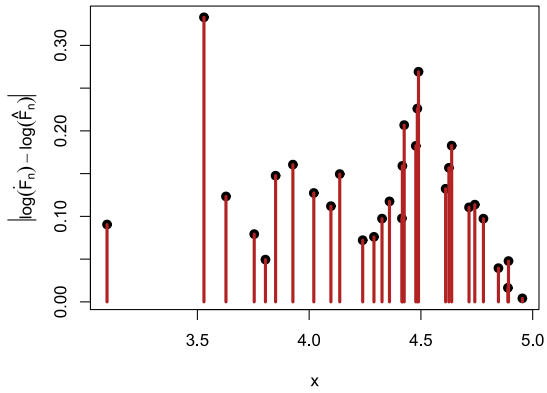


(f) $log\left(\ddot{\ddot{F}}_{(r)}\right)$ and $log\left(\hat{F}_{(r)}\right)$ for $p = 1$

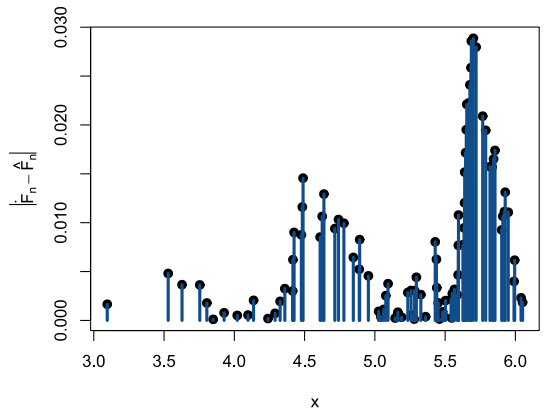
Figure 4.3.3: Comparison of $\ddot{\ddot{F}}_{(r)}$, $\hat{F}_{(r)}$, $log\left(\ddot{\ddot{F}}_{(r)}\right)$ and $log\left(\hat{F}_{(r)}\right)$ for a gamma(16.168, 0.440) model imitating MOR2 and various thresholds.



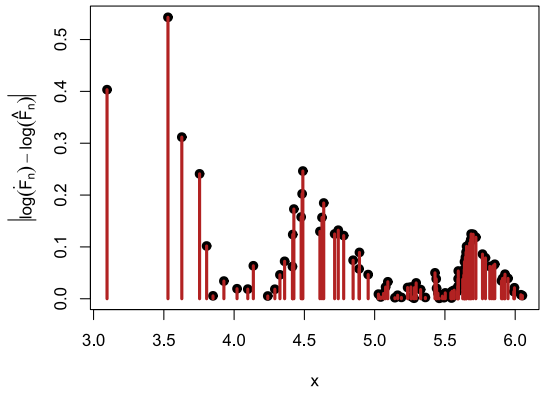
(a) $\left|\dddot{F}_n - \hat{F}_n\right|$ for $p = 0.1$



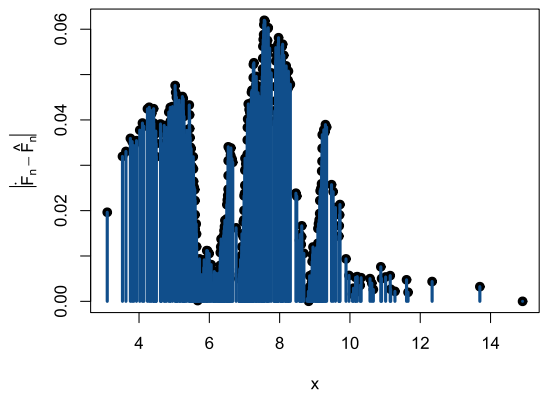
(b) $\left|log\left(\dddot{F}_n\right) - log\left(\hat{F}_n\right)\right|$ for $p = 0.1$



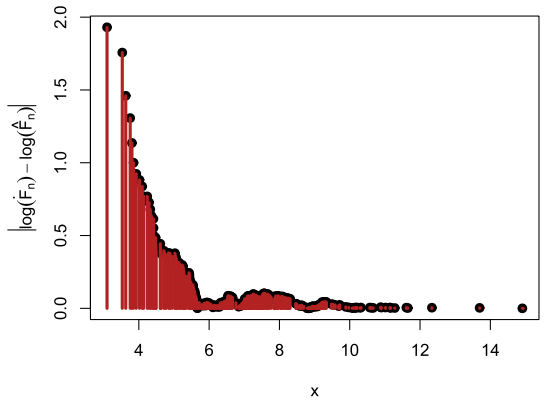
(c) $\left|\dddot{F}_n - \hat{F}_n\right|$ for $p = 0.3$



(d) $\left|log\left(\dddot{F}_n\right) - log\left(\hat{F}_n\right)\right|$ for $p = 0.3$



(e) $\left|\dddot{F}_n - \hat{F}_n\right|$ for $p = 1$



(f) $\left|log\left(\dddot{F}_n\right) - log\left(\hat{F}_n\right)\right|$ for $p = 1$

Figure 4.3.4: Comparison of $\left|\dddot{F}_{(r)} - \hat{F}_{(r)}\right|$ and $\left|log\left(\dddot{F}_{(r)}\right) - log\left(\hat{F}_{(r)}\right)\right|$ for a gamma(16.168, 0.440) model imitating MOR2 and various thresholds.

according to the updated parameters, such that the difference between the fitted CDF curves becomes extremely small in the left tail.

There is, however, a problem here; Figure 4.3.4 demonstrates that the maximum of the absolute difference between the fitted log censored Weibull CDF and log ECDF is an increasing function of the threshold. In other words, the smallest available threshold would still be selected in the majority of cases. The boxplot in Figure 4.3.5 summarises the distribution of the selected threshold, for each parametric model imitating MOR2, using the log-adjusted truncated KS test statistic based on $N = 10000$ iterations. Again, all boxplots are concentrated close to the 10^{th} empirical percentile so further adjustments to the test statistic are needed.

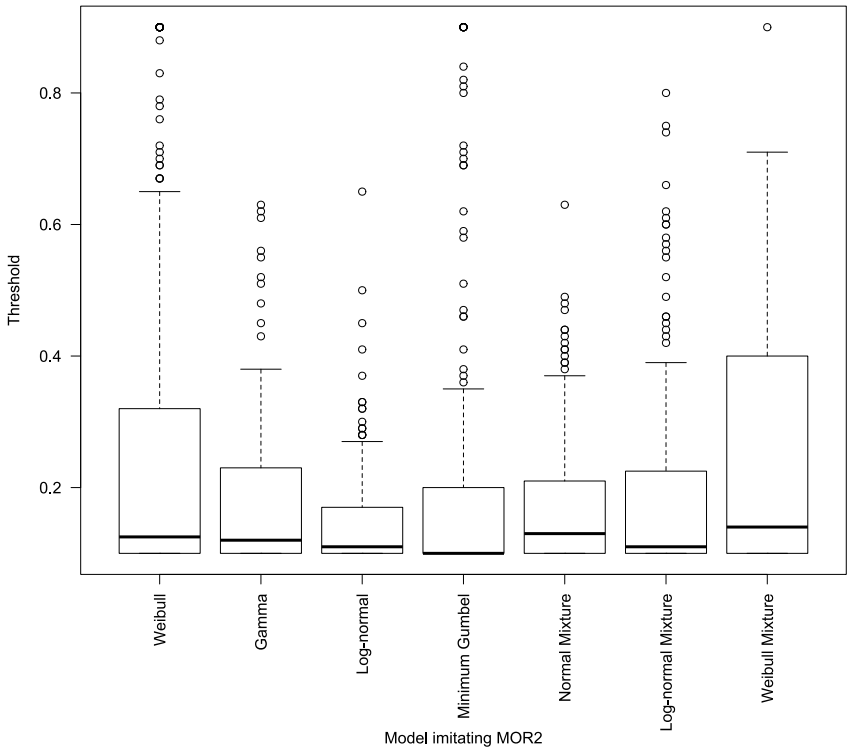


Figure 4.3.5: Boxplot of selected threshold using the log-adjusted truncated KS test statistic $D^{AKS}\left(\ddot{F}_{(r)} - \hat{F}_{(r)}\right)$.

In order to fully utilise the potential of the adjusted KS test statistic, a weighting mechanism is required such that the distance between the CDF curves in the left tail is not overly inflated and the extended interquartile range of the curves not under-provisioned. That is, the difference of logs run the risk of overshadowing the importance of any differences that may be present in the extended interquartile range of the CDF curves.

4.3.3 Relationship between the variance of the KS statistic and the proportion of censoring

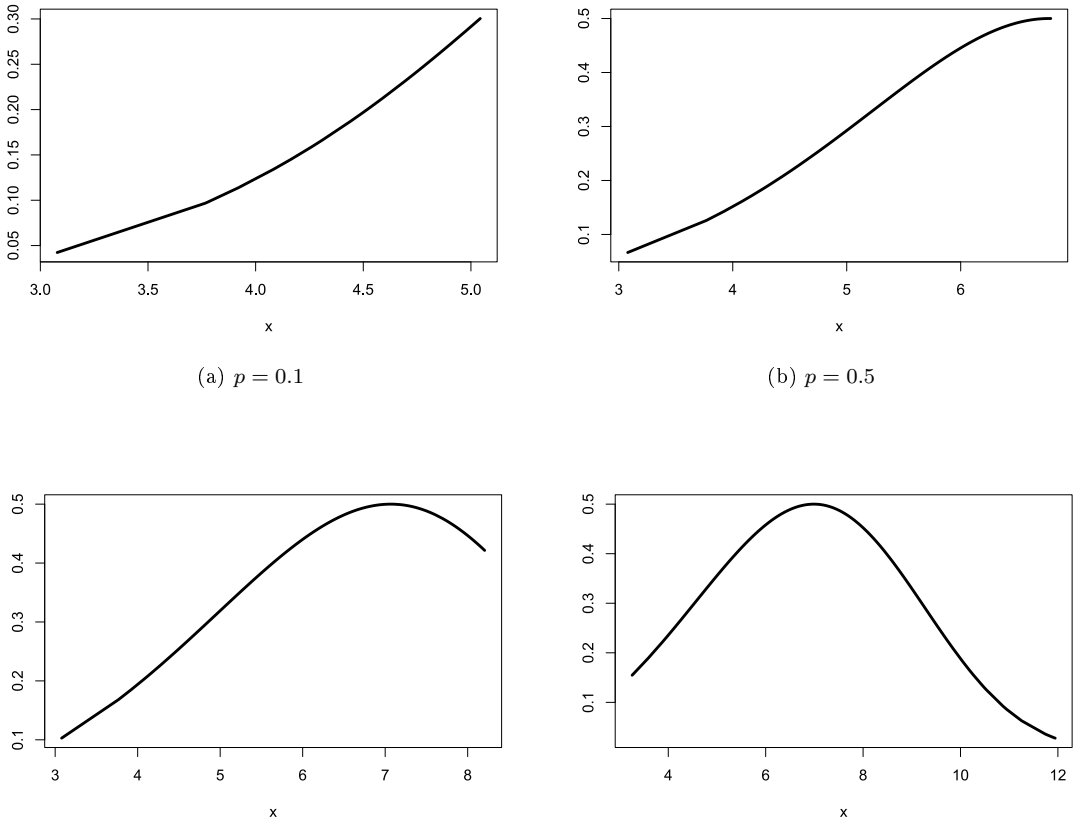
Selecting the correct censoring threshold based on either the truncated KS test statistic or adjusted KS test statistic is not sufficient. In order to use the adjusted truncated KS test statistic, a weighting mechanism is required.

The adjusted KS method inflates the distances between the CDF curves substantially more in the left tail compared to the remainder of the support. An appropriate weighting function should penalize distances in the left tail quite heavily to compensate for this. More importantly, the degree of penalisation of weighting function should decrease between the extended interquartile range of the curves, where the majority of the discrepancies between the curves would naturally occur given the fit is incorrect. The weighting function is therefore crucial to ensure that no artificial bias is introduced whilst compensating for the exaggeration introduced by the logarithmic transformation. The standard error of the fitted censored Weibull CDF at each observation meets the required criteria and is given by

$$Weight\ function = \sqrt{\ddot{F}_{(r)} \times \left(1 - \ddot{F}_{(r)}\right)}. \tag{4.3.2}$$

The right tail shrinkage is not of much concern because a poor model fit would be detected more towards the left or centre of the curves (based on the nature of the KS test and the behaviour of CDF curves). Furthermore, the logarithmic transform has very little effect on the right tail, i.e. shrinking distances already close to 0 would not have a significant effect on the threshold selection.

For data generated from a gamma(16.168, 0.440) distribution, the mechanics of the weighting function for various thresholds for the censored Weibull CDF is given by Figure 4.3.6. The weighting function for the other seven parametric models are similar and are therefore not included here.



(a) $p = 0.1$ (b) $p = 0.5$

(c) $p = 0.75$ (d) $p = 1$

Figure 4.3.6: Weighting function for the adjusted KS method.

The weight-adjusted truncated KS threshold selection measure is given by

$$D^{WAKS}\left(\dddot{F}_{(r)} - \hat{F}_{(r)}\right) = max\left\{\left|log\left(\dddot{F}_{(r)}\right) - log\left(\hat{F}_{(r)}\right)\right| * \sqrt{\dddot{F}_{(r)} \times \left(1 - \dddot{F}_{(r)}\right)}\right\}. \tag{4.3.3}$$

Observing the plots in Figure 4.3.8, the weight function seems to be performing as desired. Particularly, comparing the weighted distances $\left|log\left(\dddot{F}_{(r)}\right) - log\left(\hat{F}_{(r)}\right)\right| * \sqrt{\dddot{F}_{(r)} \times \left(1 - \dddot{F}_{(r)}\right)}$ in Figure 4.3.8 to $\left|log\left(\dddot{F}_{(r)}\right) - log\left(\hat{F}_{(r)}\right)\right|$ in Figure 4.3.4, the scale of weighted distances have been adjusted as desired. Furthermore, Figure 4.3.8 illustrates that the smallest weight adjusted KS test statistic occurs at the 30^{th} empirical percentile threshold (as recommended by Figure 4.2.1), compared to the 10^{th} empirical percentile threshold obtained without the weight penalisation. From the weight-adjusted log CDF curves in Figure 4.3.8, the weight-adjustment penalises the distances in the left tail and extended interquartile range adequately for smaller thresholds. For larger thresholds (less censoring) the logarithmic transformation appears to overpower the weight penalisation. From the boxplot in Figure 4.3.7, the performance of the new weight-adjusted KS statistic is noticeably better, however, the distribution of the thresholds selected is still too skewed towards the 10^{th} empirical percentile.

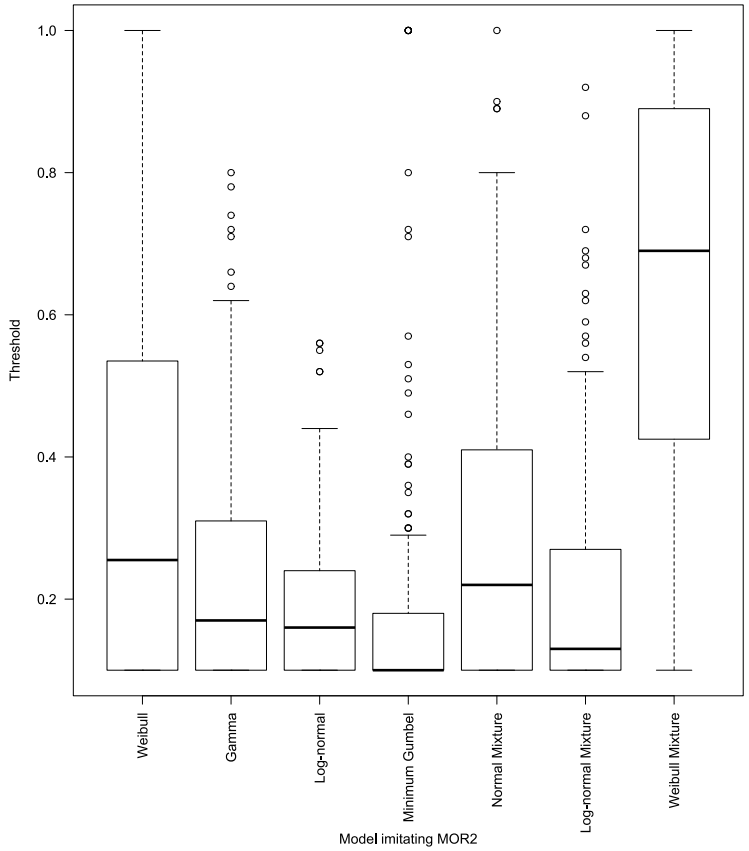
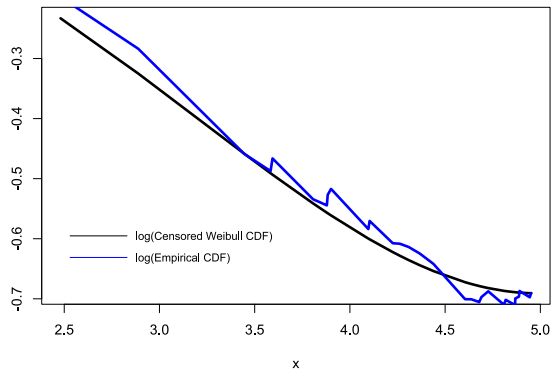
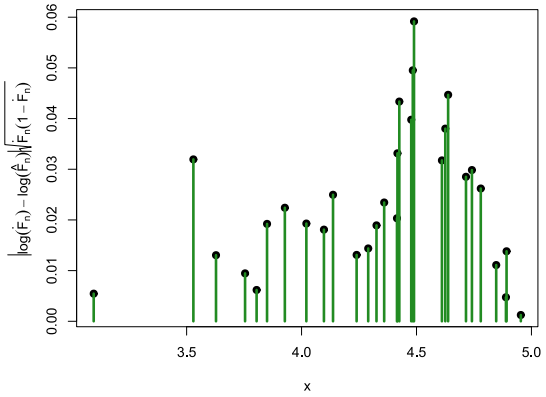


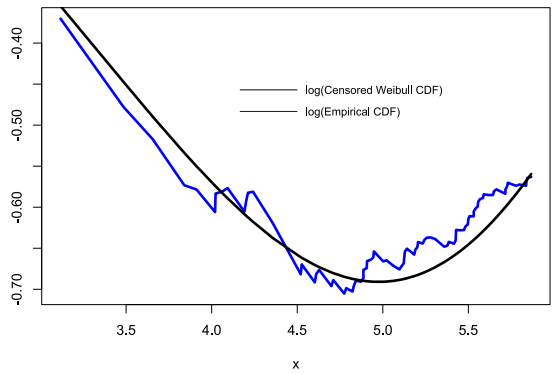
Figure 4.3.7: Boxplot of selected threshold using the weight-adjusted truncated KS test statistic $\left|log\left(\dddot{F}_n\right) - log\left(\hat{F}_n\right)\right| * \sqrt{\dddot{F}_n \times \left(1 - \dddot{F}_n\right)}$.



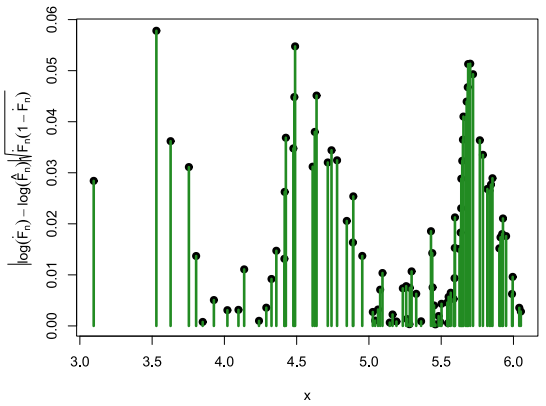
(a) Weight adjusted $log\left(\ddot{\bar{F}}_{(r)}\right)$ and $log\left(\hat{F}_{(r)}\right)$ for $p = 0.1$



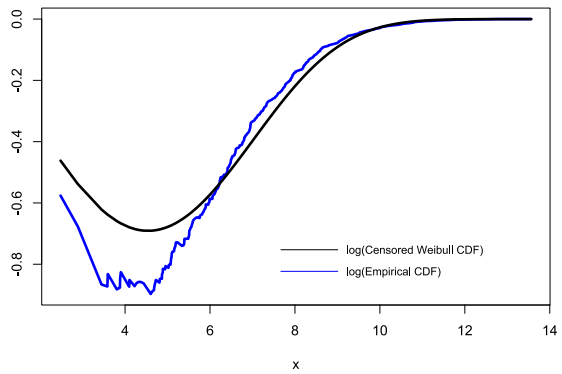
(b) $\left|log\left(\ddot{\bar{F}}_{(r)}\right) - log\left(\hat{F}_{(r)}\right)\right| * \sqrt{\ddot{\bar{F}}_{(r)} \times \left(1 - \ddot{\bar{F}}_{(r)}\right)}$ for $p = 0.1$



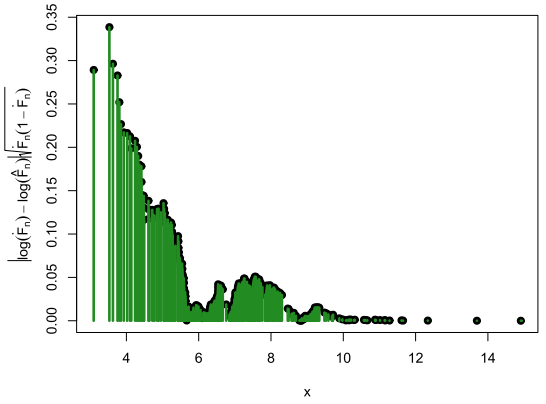
(c) Weight adjusted $log\left(\ddot{\bar{F}}_{(r)}\right)$ and $log\left(\hat{F}_{(r)}\right)$ for $p = 0.3$



(d) $\left|log\left(\ddot{\bar{F}}_{(r)}\right) - log\left(\hat{F}_{(r)}\right)\right| * \sqrt{\ddot{\bar{F}}_{(r)} \times \left(1 - \ddot{\bar{F}}_{(r)}\right)}$ for $p = 0.3$



(e) Weight adjusted $log\left(\ddot{\bar{F}}_{(r)}\right)$ and $log\left(\hat{F}_{(r)}\right)$ for $p = 1$



(f) $\left|log\left(\ddot{\bar{F}}_{(r)}\right) - log\left(\hat{F}_{(r)}\right)\right| * \sqrt{\ddot{\bar{F}}_{(r)} \times \left(1 - \ddot{\bar{F}}_{(r)}\right)}$ for $p = 1$

Figure 4.3.8: Weight-adjusted $log\left(\ddot{\bar{F}}_{(r)}\right)$ and $log\left(\hat{F}_{(r)}\right)$ and $\left|log\left(\ddot{\bar{F}}_{(r)}\right) - log\left(\hat{F}_{(r)}\right)\right| * \sqrt{\ddot{\bar{F}}_{(r)} \times \left(1 - \ddot{\bar{F}}_{(r)}\right)}$ for gamma(16.168, 0.440) model imitating MOR2 for various thresholds.

4.3.4 A standardised weighting function for the adjusted KS statistic

Recall that as the proportion of censoring decreases, the ECDF smooths out and the fitted censored Weibull CDF re-adjusts according to the updated parameters obtained in the censored maximum likelihood estimation process. This means that the distances between the fitted CDF and ECDF curves re-adjust accordingly (due to smoothing). Moreover, the distances become closer in the left tail and as a result, the log distances increase substantially. The scale of shrinkage induced by the weight function introduced above is not enough to compensate for the resulting exponential exaggeration of distances in the left tail. In other words, the more observations included in the censoring process the closer the CDF curves become in the left tail (not necessarily true for the rest of the support) and the larger the resulting log difference distances. The weight function, therefore, needs to penalise the log distances based on the number of observations used in the estimation process as well. The weight function needs to dynamically shrink the log distances in the same way the original weight function does, however, here the sample size needs to be taken into consideration so that the log-transformed distances are dealt with scalably, i.e. a standardised weight function. The ideal weight function is adjusted to

$$Standardised\ weight\ function = \sqrt{\frac{\dddot{F}_{(r)} \times \left(1 - \dddot{F}_{(r)}\right)}{r}}, \tag{4.3.4}$$

Figure 4.3.10 shows how the standardised weight function penalises the adjusted KS distances in proportion to the size r. Namely, the larger the value of r, the smaller the shrinkage multiplier becomes.

The final standardised-weighted adjusted KS (SWAKS) threshold selection measure is given by

$$D^{SWAKS}\left(\dddot{F}_{(r)} - \hat{F}_{(r)}\right) = max\left\{\left|log\left(\dddot{F}_{(r)}\right) - log\left(\hat{F}_{(r)}\right)\right| * \sqrt{\frac{\dddot{F}_{(r)} \times \left(1 - \dddot{F}_{(r)}\right)}{r}}\right\}. \tag{4.3.5}$$

From Tables 4.3.1 and 4.3.2, the SWAKS values are smallest when the threshold is chosen to be the 40^{th} empirical percentiles for a sample generated from a gamma(16.168, 0.440) and the 100^{th} percentile (no censoring) for a sample from a Weibull(7.378, 6.738). The smallest values for each threshold selection technique are bolded for ease of readability. This chosen threshold may change depending on the simulated sample. Only the SWAKS method selects the correct threshold for both the Gamma and Weibull models. Furthermore, from the boxplot in Figure 4.3.9, the performance of SWAKS threshold selection is outstanding. The median of each simulated model falls in the desired interval recommended by Figure 4.2.1.

Threshold p	Threshold selection technique for data generated from Gamma(16.168, 0.440)			
	$D^{\star}\left(\dddot{F}_{(r)} - \hat{F}_{(r)}\right)$	$D^{AKS}\left(\dddot{F}_{(r)} - \hat{F}_{(r)}\right)$	$D^{WAKS}\left(\dddot{F}_{(r)} - \hat{F}_{(r)}\right)$	$D^{SWAKS}\left(\dddot{F}_{(r)} - \hat{F}_{(r)}\right)$
0.1	**0.013**	**0.627**	**0.109**	0.023
0.2	0.018	0.676	0.137	0.014
0.3	0.026	0.976	0.175	0.013
0.4	0.027	0.885	0.193	**0.009**
0.5	0.026	0.913	0.155	0.011
0.6	0.027	1.064	0.152	0.012
0.7	0.042	1.222	0.169	0.016
0.8	0.050	1.327	0.194	0.017
0.9	0.062	1.588	0.224	0.022
1	0.073	1.834	0.266	0.028

Table 4.3.1: KS distances for various threshold selection techniques for a sample generated from a Gamma(16.168, 0.440).

Threshold p	Threshold selection technique for data generated from Weibull(7.378, 6.738)			
	$D^{\star}\left(\ddot{\ddot{F}}_{(r)} - \hat{F}_{(r)}\right)$	$D^{AKS}\left(\ddot{\ddot{F}}_{(r)} - \hat{F}_{(r)}\right)$	$D^{WAKS}\left(\ddot{\ddot{F}}_{(r)} - \hat{F}_{(r)}\right)$	$D^{SWAKS}\left(\ddot{\ddot{F}}_{(r)} - \hat{F}_{(r)}\right)$
0.1	**0.006**	**1.557**	**0.054**	0.010
0.2	0.024	2.575	0.065	0.008
0.3	0.030	3.010	0.076	0.008
0.4	0.031	2.673	0.076	0.006
0.5	0.030	2.532	0.073	0.005
0.6	0.033	2.314	0.076	0.005
0.7	0.033	2.413	0.073	0.005
0.8	0.032	2.528	0.071	0.004
0.9	0.034	2.432	0.074	0.004
1	0.034	2.571	0.065	**0.003**

Table 4.3.2: KS distances for various threshold selection techniques for a sample generated from a Weibull(7.378, 6.738).

The plots in Figure 4.3.10 are very similar to those in Figure 4.3.9. Here the difference in log distances is scaled in proportion to the size r in order to highlight any discrepancies that may be present between the extended interquartile range of the CDF curves. Only the plots for data generated from a Gamma distribution for MOR2 are given, however, the plots for the remainder of the parametric models for both MOR datasets are very similar and have therefore been omitted.

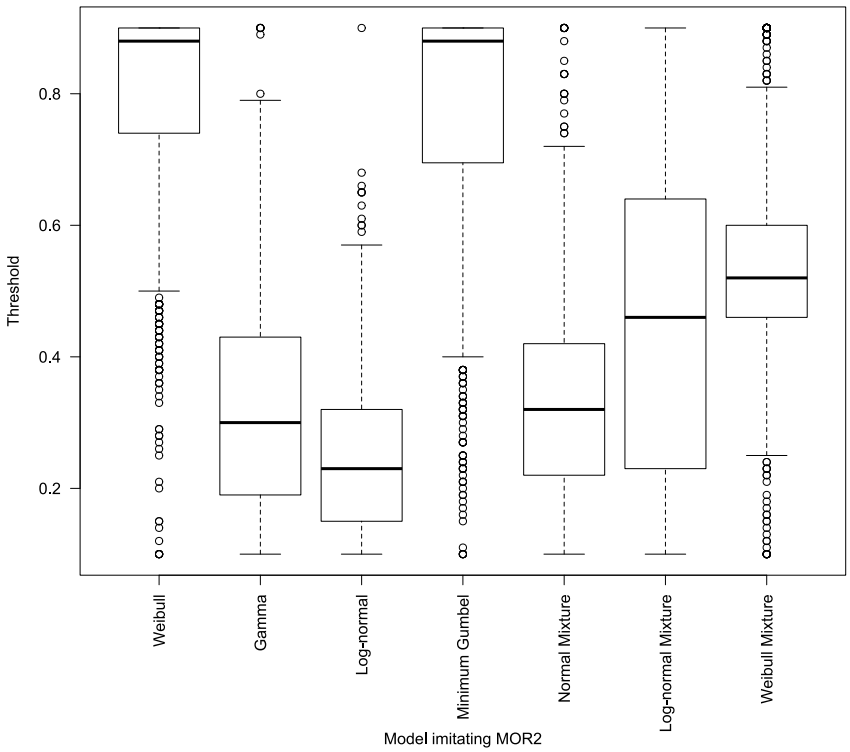
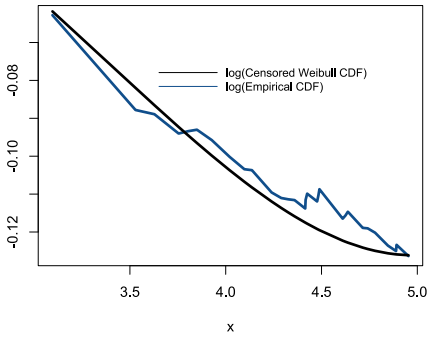
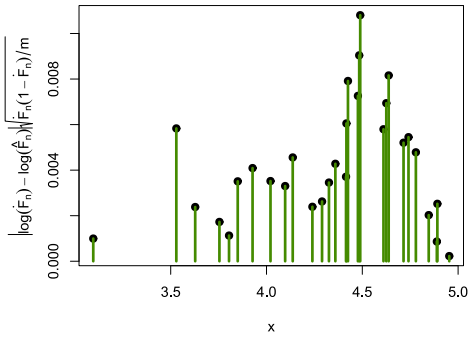


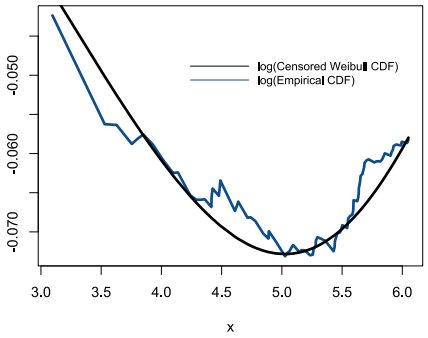
Figure 4.3.9: Boxplot of selected threshold using the standardised weight-adjusted truncated KS test statistic $\left|log\left(\ddot{\ddot{F}}_{(r)}\right) - log\left(\hat{F}_{(r)}\right)\right| * \sqrt{\frac{\ddot{\ddot{F}}_{(r)} \times \left(1 - \ddot{\ddot{F}}_{(r)}\right)}{r}}$.



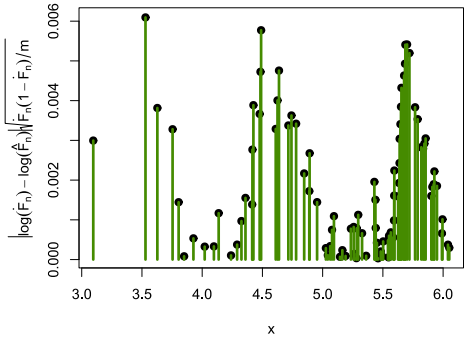
(a) Standardised weight adjusted $log\left(\ddot{\ddot{F}}_{(r)}\right)$ and $log\left(\hat{F}_{(r)}\right)$ for $p = 0.1$



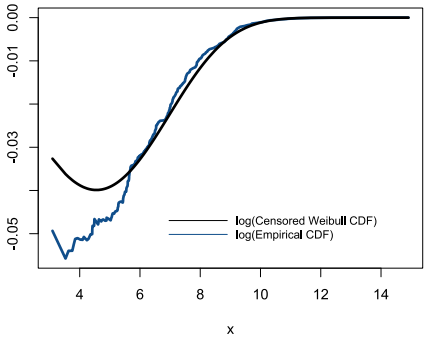
(b) $\left|log\left(\ddot{\ddot{F}}_{(r)}\right) - log\left(\hat{F}_{(r)}\right)\right| * \sqrt{\frac{\ddot{\ddot{F}}_{(r)} \times \left(1 - \ddot{\ddot{F}}_{(r)}\right)}{r}}$ for $p = 0.1$



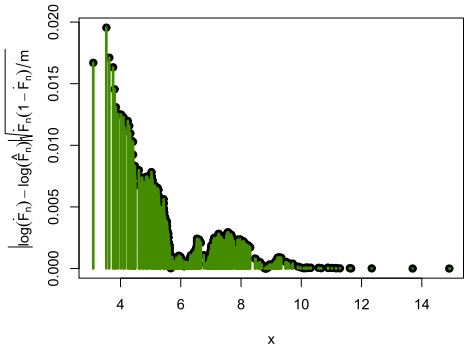
(c) Standardised weight adjusted $log\left(\ddot{\ddot{F}}_{(r)}\right)$ and $log\left(\hat{F}_{(r)}\right)$ for $p = 0.3$



(d) $\left|log\left(\ddot{\ddot{F}}_{(r)}\right) - log\left(\hat{F}_{(r)}\right)\right| * \sqrt{\frac{\ddot{\ddot{F}}_{(r)} \times \left(1 - \ddot{\ddot{F}}_{(r)}\right)}{r}}$ for $p = 0.3$



(e) Standardised weight adjusted $log\left(\ddot{\ddot{F}}_{(r)}\right)$ and $log\left(\hat{F}_{(r)}\right)$ for $p = 1$



(f) $\left|log\left(\ddot{\ddot{F}}_{(r)}\right) - log\left(\hat{F}_{(r)}\right)\right| * \sqrt{\frac{\ddot{\ddot{F}}_{(r)} \times \left(1 - \ddot{\ddot{F}}_{(r)}\right)}{r}}$ for $p = 1$

Figure 4.3.10: Standardised weight-adjusted $log\left(\ddot{\ddot{F}}_{(r)}\right)$ and $log\left(\hat{F}_{(r)}\right)$ and $\left|log\left(\ddot{\ddot{F}}_{(r)}\right) - log\left(\hat{F}_{(r)}\right)\right| * \sqrt{\frac{\ddot{\ddot{F}}_{(r)} \times \left(1 - \ddot{\ddot{F}}_{(r)}\right)}{r}}$ for gamma(16.168, 0.440) model imitating MOR2 for various thresholds.

4.3.5 The efficiency of the SWAKS-MLE algorithm

For ease of readability, the algorithm for the SWAKS-MLE is provided in Algorithm 4.4. If the range of threshold candidates is selected to be $C \in [0.1, 1]$ in steps of 0.01, the total number of quantiles that need to be estimated for a single model that imitates the MOR dataset is

$$\begin{aligned} \textit{Total number of } 5^{th} \textit{ quantiles estimated} &= 10\,000 \times 91 \\ &= 910\,000. \end{aligned}$$

The SWAKS-MLE requires 0.0002 the number of quantile estimates required by the B-MLE approach. However, since [16] only consider the 10^{th}, 20^{th}, 30^{th}, 40^{th} and 50^{th} percentiles as threshold candidates, this study will only consider threshold candidates for $C \in [0.1, 0.5]$ in steps of 0.01 for the SWAKS MLE to ensure the results are directly comparable.

Algorithm 4.4 SWAKS-MLE algorithm

1. Generate a sample $\boldsymbol{X}$ of size $n = 300$ from the following models (parameterised by Tables 3.4.1 and 3.4.2) used to imitate the real MOR1 and MOR2 datasets:
2. Select a censoring threshold $C \in [0.1, 0.5]$.
3. Fit the CW-MLE to the smallest r observations.
4. Calculate $D^{SWAKS}\left(\ddot{\hat{F}}_{(r)} - \hat{F}_{(r)}\right)$.
5. Repeat Steps 2 - 4 for a variety of threshold candidates.
6. Select the threshold with the smallest (SWAKS) test statistic value.
7. Obtain the 5^{th} quantile using the threshold obtained in Step 6.
8. Repeat Steps 1 - 7 for $N = 10000$ iterations.
9. Obtain the MSE and RMSE using the estimates obtained in Step 8.

4.4 Simulation comparison

In the following simulations, the 5^{th} percentile estimate achieved by the SWAKS-MLE is compared to the B-MLE, the CW-MLE from Chapter 3, as well as the MIX and MIX7. These quantile estimate techniques were the best performing in the aforementioned chapter.

The models from which the data are simulated are the same models introduced in Chapter 4: seven models imitating the MOR1 and MOR2 datasets. The sample size for each model is set to $n = 300$. The RMSE for the quantile estimates are summarised in Tables 4.4.1 and 4.4.2. As in previous chapters, darker backgrounds indicate larger values of RMSE. The simulation process of the B-MLE is exceptionally computationally expensive, therefore the results obtained by the B-MLE in [16] are quoted here.

In the remainder of this section, the simulation results in Tables 4.4.1 and 4.4.2 are analysed to compare the five quantile estimates. Furthermore, in order to understand the advantages and limitations of the SWAKS-MLE, the bias and standard error of the B-MLE, CW-MLE and SWAKS-MLE are compared.

Model	CW-MLE	MIX	CMIX7	B-MLE	SAWKS-MLE
Weibull	0.150	0.148	0.152	0.147	0.141
Log-normal	0.163	0.257	0.159	0.174	0.158
Gamma	0.163	0.217	0.159	0.166	0.168
Minimum Gumbel	0.175	0.153	0.147	0.143	0.139
Normal Mixture	0.112	0.141	0.110	0.120	0.141
Log-normal Mixture	0.169	0.197	0.155	0.156	0.140
Weibull Mixture	0.163	0.156	0.163	0.149	0.146

Table 4.4.1: RMSE of the 5^{th} quantile estimates from a Weibull CW-MLE, MIX, MIX7, B-MLE and SAWKS-MLE for various models imitating MOR1.

Model	CW-MLE	MIX	CMIX7	B-MLE	SAWKS-MLE
Weibull	0.138	0.128	0.131	0.131	0.127
Log-normal	0.139	0.261	0.128	0.148	0.137
Gamma	0.138	0.191	0.132	0.142	0.138
Minimum Gumbel	0.155	0.140	0.138	0.134	0.129
Normal Mixture	0.127	0.133	0.122	0.132	0.127
Log-normal Mixture	0.137	0.146	0.139	0.134	0.126
Weibull Mixture	0.163	0.158	0.161	0.156	0.151

Table 4.4.2: RMSE of the 5^{th} quantile estimates from a Weibull CW-MLE, MIX, MIX7, B-MLE and SAWKS-MLE for various models imitating MOR2.

4.4.1 Comparison of the RMSE of the quantile estimates

From the background shades in Tables 4.4.1 and 4.4.2, it is easy to see that either the CMIX7 or the SWAKS-MLE is the best quantile estimate in all seven parametric models considered. For the models imitating MOR1, MIX7 has the smallest RMSE in 2 models while SWAKS-MLE has the smallest in 5 models. Among the models imitating MOR2, the SWAKS-MLE is the best in 4 models while CMIX7 is the best in 3 models. Clearly, the SWAKS-MLE achieves the largest number of pole positions in all the parametric models considered. The SWAKS-MLE outperforms the B-MLE in all models except for the Normal mixture and Gamma models imitating the MOR1 dataset.

To summarise the performance of these five quantiles estimates, their average ranks for each parametric model simulated from is compared. For example, in the Log-normal model imitating the MOR1 dataset, the ranks are: CW-MLE (4) ,MIX (5), CMIX7 (2), B-MLE (3) and SWAKS-MLE (1) . The ranks of each quantile are averaged across all seven parametric models for both the MOR1 and MOR2 datasets. The results are given by:

	CW-MLE	MIX	MIX7	B-MLE	SWAKS-MLE
Averaged rank	3.428	4.071	2.5	2.857	1.785

The SWAKS-MLE has the smallest averaged rank among all quantile estimates and the CMIX7 is in the second place. Moreover, since the lumber strength data is commonly modelled by the Weibull distribution, the models that are closely related to said distribution, such as the Weibull mixture and Minimum Gumbel, should be assigned slightly more weight in the averaged rank calculation, [16]. The SWAKS-MLE performs better with those models compared to the B-MLE and CMIX7.
The SWAKS-MLE is better than the fixed threshold CW-MLE except for the Gamma and Log-normal mixture model. It may, therefore, be worthwhile to compare the bias and standard errors of the B-MLE, CW-MLE and SWAKS-MLE.

4.4.2 Comparison of the bias and standard error of the B-MLE, CW-MLE and SWAKS-MLE

The bias and standard error (SE) of the quantile estimates from the B-MLE, CW-MLE and SWAKS-MLE for the models imitating MOR2 are summarised in Table 4.4.3. The results for the models imitating the MOR1 dataset are similar and are thus excluded. As in previous tables, the greener background shade corresponds to a smaller bias and standard error. The B-MLE reduces the bias the most followed by the SWAKS-MLE. The SWAKS-MLE has the smallest standard error across all parametric models considered, followed by the CW-MLE.

Parametric	Bias			SE		
Model	CW-MLE	B-MLE	SWAKS-MLE	CW-MLE	B-MLE	SWAKS-MLE
Weibull	0.36	0.95	0.71	13.5	13.1	12.7
Log-normal	4.59	0.43	2.27	13.5	14.8	13.4
Gamma	3.46	0.5	3.75	13.4	14.2	13.4
Minimum Gumbel	4.32	0.24	0.54	14.8	13.4	12.9
Normal Mixture	1.75	1.94	1.04	12.5	13.2	12.6
Log-normal Mixture	4.82	0.97	1.93	12.9	13.3	12.4
Weibull Mixture	2.34	2.12	1.35	16.5	15.6	15.1

Table 4.4.3: Bias and standard error (SE) (x100) of the CW-MLE, B-MLE and SWAKS-MLE quantile estimates for the models imitating the MOR2 data set. .

This is rather impressive, considering the SWAKS-MLE is a data-driven procedure. Clearly from the results discussed, the SWAKS-MLE outperforms the B-MLE in terms of accuracy (reduces the RMSE and standard errors of the quantile estimates) and computational intensity in all models considered.

4.5 Chapter summary

Chapter 4 revisits the B-MLE proposed by [16] and proposes a new threshold selection technique, namely the SWAKS-MLE. These two methodologies can be used to select an optimal threshold for the CW-MLE quantile estimation technique. The B-MLE performs better than the fixed threshold CW-MLE from Chapter 3, however, it is computationally intensive. The newly proposed SWAKS-MLE makes use of a standardised-weighted log-adjusted truncated Kolmogorov-Smirnov test to select the optimal threshold. The SWAKS-MLE outperforms all the quantile estimation techniques used in this study and should be the preferred choice for lower quantile estimation.

Chapter 5

Future Work and Conclusions

This dissertation focuses on statistical methods for lower quantile estimation of modulus of rupture wood datasets, initially studied in [16]. In particular, Chapters 2, 3, 4 and 5 of [16] were thoroughly re-worked. More so, the main contribution here was the data-driven threshold selection technique (SWAKS-MLE). The study of lower quantile estimation of wood strength data is a crucial problem for the safety and reliability of wood built structures. Several non-parametric quantile estimation techniques are available and intuitive to use, such as the empirical quantile and KDE quantile estimates. These estimates may be unbiased but suffer from large standard errors. Fully parametric quantile estimates do not have this problem, however, they may be extremely bias under misspecified distributions. To circumvent the flaws of both the non-parametric and parametric quantile estimates, the wood engineering industrial standard [1] apply a semi-parametric censored Weibull MLE approach.

The fundamental idea of the censored Weibull MLE approach is to fit the left tail of the data, rather than the entire range in order to sufficiently obtain lower quantile estimates. Thus if there is a desire to estimate the lower quantile, all observations to the right of the threshold are artificially censored. This allows the investigator to focus on the fit of a parametric PDF to the left tail and artificially censor the rest of the observations. From the simulation and theoretical studies presented, the censored Weibull MLE approach addresses the issues of parametric and non-parametric quantile estimation, that is, a smaller MSE of the quantile estimates is generally achieved.

The censored Weibull MLE is slightly more biased compared to the empirical quantile estimate, however, it is noticeably more efficient in terms of the standard error of the quantile estimates. The censored Weibull MLE is less efficient than the fully parametric approach under the correct model (i.e. Weibull model), however, remarkably less biased under misspecified models.

The censored Weibull MLE approach does have its own shortcomings that are addressed in this dissertation. The first potential successor to the censored Weibull MLE is to consider a more complex model, to make use of more sample data in the estimation procedure. This being said, the two-component Weibull and censored Weibull mixture are investigated. The uncensored Weibull mixture performs adequately, however, it struggles to approximate the left tail of several of the parametric models considered to imitate the real dataset. The censored Weibull mixture model performs the best and reduces the bias of the censored Weibull MLE, however, the choice of threshold remains to be investigated.

Another approach to improving the censored Weibull MLE is to select the optimum threshold for the dataset. This idea leads to the bootstrap censored Weibull MLE proposed by [16]. The approach makes use of the bootstrap to obtain an MSE estimate based on the bootstrap samples. Furthermore, the threshold with the smallest bootstrap MSE is then used in the censored Weibull MLE to obtain the quantiles estimates of interest. The B-MLE approach increases the standard errors of the quantile estimates, however, this is not uncommon of a data-driven process and the bias is significantly reduced. The B-MLE offers improvements over the fixed threshold censored Weibull MLE, however, the computational intensity of this approach is rather staggering.

An alternative improvement to the censored Weibull MLE is to select the threshold that minimises a

standardised-weighted log-adjusted Kolmogorov-Smirnov test statistic between the ECDF of the dataset and the censored Weibull CDF. The idea here is to select the threshold for the dataset that offers the best fit. The SWAKS censored Weibull MLE (SWAKS-MLE) performs better than all other methods presented in this dissertation and is vastly less computationally intensive compared to the B-MLE. Furthermore, the SWAKS-MLE does not enjoy the benefits of reduced bias compared to the B-MLE approach, however, it is very much on par and comparable to the censored Weibull MLE approach. Moreover, the SWAKS-MLE offers vast improvements over all other methods in terms of the standard errors of the quantile estimates, which is rather impressive considering this is a data-driven approach.

As for future research, the asymptotic behaviour and consistency of the SWAKS-MLE threshold selection from a mathematical point of view needs to be completed. This will provide insight as to whether the performance of the SWAKS-MLE will be affected by larger sample sizes (how the log transformation behaves asymptotically). Moreover, an adjustment to the SWAKS-MLE in order to estimate upper quantile estimates remains to be investigated. This will allow a direct comparison with the B-MLE and peaks-over-threshold threshold comparison in [17]. Finally, the SWAKS technique could potentially be applied in the censored Weibull mixture setting to obtain the optimum threshold and further improvements on the lower quantile estimates.

Bibliography

[1] ASTM. *Standard specification for computing reference resistance of wood-based materials and structural connections for load and resistance factor design D5457*. American Society for Testing Materials. Philadephia Pa, (2004).

[2] L. J. Bain and M. Engelhardt. *Introduction to Probability and Mathematical Statistics*. The Duxbury advanced series in statistics and decision sciences. Brooks/Cole, 2nd edition, (1987).

[3] L. R. Burden and J. D. Faires. *Numerical Analysis*. CENCAGE Learning, 9th edition, (2010).

[4] Y. Cheng. *Wood property relationships and survival models in reliability*. PhD thesis, University of British Columbia, (2010).

[5] S. Coles, J. Bawa, L. Trenner, and P. Dorazio. *An Introduction to Statistical Modeling of Extreme Values*, volume 208. Springer, (2001).

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, (1977).

[7] Beirlant et al. *Statistics of Extremes: Theory and Applications*. John Wiley & Sons, (2004).

[8] R. A. Fisher and L. H. C. Tippett. Limiting forms of the frequency distribution of the largest or smallest member of a sample. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 24 of number 2, pages 180–190. Cambridge University Press, (1928).

[9] M. Fréchet. Sur la loi de probabilité de l'écart maximum. *Ann. Soc. Math. Polon.*, 6:93–116, (1927).

[10] R. J. Hyndman and Y. Fan. Sample quantiles in statistical packages. *The American Statistician*, 50(4):361–365, (1996).

[11] A. J. Izenman. Recent developments in nonparametric density estimation. *Journal of the American Statistical Association*, 86(413):205–224, (1991).

[12] N. L. Johnson, S. Kotz, and N. Balakrishnan. *Continuous univariate distributions*, volume 1. John Wiley\& Sons, (1994).

[13] M. C. Jones and S. J. Sheather. A brief survey of bandwidth selection for density estimation. *Journal of the American Statistical Association*, 91(433):401–407, (1996).

[14] J. F. Lawless. *Statistical Models and Methods for Lifetime Data*. John Wiley & Sons, (2011).

[15] E. L. Lehmann and G. Casella. *Theory of Point Estimation*. Springer Science & Business Media, (2006).

[16] Y. Liu. *Lower quantile estimation of wood strength data*. PhD thesis, University of British Columbia, (2012).

[17] Y. Liu, M. Salibián-Barrera, R. H. Zamar, and J. V. Zidek. Using artificial censoring to improve extreme tail quantile estimates. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 67(4):791–812, (2018).

[18] G. J. McLachlan. On bootstrapping the likelihood ratio test statistic for the number of components in a normal mixture. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 36(3):318–324, (1987).

[19] H. Rinne. *The Weibull Distribution: A Handbook Crc Press*. (2009).

[20] R. Serfling. *Approximation Theorems of Mathematical Statistics*, volume 162. John Wiley & Sons, (2009).

[21] S. J. Sheather and M. C. Jones. A reliable data-based bandwidth selection method for kernel density estimation. *Journal of the Royal Statistical Society: Series B (Methodological)*, 53(3):683–690, (1991).

[22] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Routledge, (2018).

[23] E. W. Stacy. A generalization of the gamma distribution. *The Annals of Mathematical Statistics*, 33(3):1187–1192, (1962).

[24] W. Weibull. A statistical distribution function of wide applicability. *Journal of Applied Mechanics*, 18(3):293–297, (1951).

Appendix

All code files used in this dissertation are available on a GitHub repository at: https://github.com/Jarod-Smithy/MSc-code-files/. It should be noted that for copyright purposes, the repository is private. In order to access the code files I request that the interested reader and or user email their GitHub username to jarodsmith706@gmail.com in order to collaborate with me and access the files. That being said, a large portion of the code is provided below, however, to save space due to the copious amount of code not everything is included.

```
################################
####    Chapter 1 code    ####
################################

## Exploring the MOR dataset
#read in the data and split into MOR1 and MOR2 datasets
setwd("C:/Users/Jarod/Google Drive/Jarod M- 2018 and 2019/Data
set/AC/Data")
MOR <- read.csv('MOR_Data.csv',sep = ',')
MOR1 <- MOR[MOR$Sample == 'MOR1',2]
MOR2 <- MOR[MOR$Sample == 'MOR2',2]

hist(MOR2,xlab="MOR2", breaks=15, main="")

################################
####    Chapter 2 code    ####
################################

### kde quantile estimate

setwd("C:\\Users\\Jarod\\Google Drive\\Jarod M- 2018 and 2019\\Data
set\\AC\\Data")
# Read CSV into R
MyData <- read.csv(file="MOR_Data.csv", header=TRUE, sep=",")
MOR1 <- MyData[MyData$Sample == 'MOR1',2]
MOR2 <- MyData[MyData$Sample == 'MOR2',2]

DS <- MOR2

#bandwidth selection
MOR_bw <- bw.SJ(DS, nb=15,
                method ="ste")

b_MOR = sample(DS, size = length(DS), replace = TRUE) + rnorm(length(DS),
0, MOR_bw)



hist(DS, breaks= 15, prob =TRUE, main="", xlab = "MOR2")
lines(density(b_MOR, bw = "SJ-ste"), lwd = 3, lty = 'dashed', col
="dodgerblue4")


dens = density(MOR1)
x = dens$x
F = cumsum(dens$y)/sum(dens$y)
approx(F,x,0.5)

quant_func <- function(data,q){
  dens = density(data)
  x    = dens$x
  F    = cumsum(dens$y)/sum(dens$y)
  out  = approx(F,x,q)$y
  return(out)
}

quant_func(b_MOR1,0.05)

##Chapter 2 - choice of empirical quantile definition

library(data.table)
```

```
library(future)
library(ordinal)

n  <- 300
MC <- 10000

EQ.choice <- function(MC,sample_size, rdens, q, param,gum.ind) {
  library(ordinal)
  quant.all <- c()
  quant.9 <- matrix(nrow = 9, ncol = length(q))

  for (j in 1:MC){
    #generate the data
    if(gum.ind==TRUE){
      dat  <- rdens(sample_size,param[1], param[2],max = FALSE)
    }
    if(gum.ind==FALSE){
      dat <- rdens(sample_size, param[1], param[2])
    }
    #Obtain quantile estimates for each definition
    for (i in 1:9) {
      quant.9[i,1:length(q)]  <- quantile(dat, q, type = i)
    }
    quant.all <- cbind(quant.all, quant.9)
  }

  return(quant.all)
}

plan(multisession)
tempjob1 %<-% EQ.choice(MC,n,rweibull,c(0.01,0.05), c(7,7), FALSE)
tempjob2 %<-% EQ.choice(MC,n,rgamma,c(0.01,0.05), c(14,1/0.6), FALSE)
tempjob3 %<-% EQ.choice(MC,n,rlnorm,c(0.01,0.05), c(2,0.3), FALSE)
tempjob4 %<-% EQ.choice(MC,n,rgumbel,c(0.01,0.05), c(7,0.5), TRUE)

start     <- Sys.time()
temp.list <- lapply(ls(pattern = "temp"), get)
df        <- cbind(temp.list)
rm(list=ls(pattern="temp"))
end       <- Sys.time()
end - start


type = 1
first <- df[[type]][,seq(1,2*MC,2)]
fifth <- df[[type]][,seq(2,2*MC,2)]

df_1 <- c()
df_5 <- c()
for (i in 1:MC) {
  df_1 <- cbind(df_1, first[,i])
  df_5 <- cbind(df_5, fifth[,i])
}

df_1 <- data.frame(df_1)
df_5 <- data.frame(df_5)

# RMSE
RMSE_1_gum <- sqrt(apply((t(df_1) - qgumbel(0.01,7,0.5,
max=FALSE))^2,2,sum)/MC)
```

```
RMSE_5_gum <- sqrt(apply((t(df_5) - qgumbel(0.05,7,0.5,
max=FALSE))^2,2,sum)/MC)

RMSE_1_w <- sqrt(apply((t(df_1) - qweibull(0.01,7,7))^2,2,sum)/MC)
RMSE_5_w <- sqrt(apply((t(df_5) - qweibull(0.05,7,7))^2,2,sum)/MC)

RMSE_1_gam <- sqrt(apply((t(df_1) - qgamma(0.01,14,1/0.6))^2,2,sum)/MC)
RMSE_5_gam <- sqrt(apply((t(df_5) - qgamma(0.05,14,1/0.6))^2,2,sum)/MC)

RMSE_1_ln <- sqrt(apply((t(df_1) - qlnorm(0.01,2,0.3))^2,2,sum)/MC)
RMSE_5_ln <- sqrt(apply((t(df_5) - qlnorm(0.05,2,0.3))^2,2,sum)/MC)

# Basic boxplot ####
# Type = 1
boxplot(t(df_5), xlab = "Empirical quantile definition", ylab = "Quantile
value")
abline(h=qweibull(0.05,7,7),col="firebrick1", lwd=4, lty=2)

# Type = 2
boxplot(t(df_5), xlab = "Empirical quantile definition", ylab = "Quantile
value")
abline(h=qgamma(0.05,14,1/0.6),col="darkgoldenrod3", lwd=4, lty=2)

# Type = 3
boxplot(t(df_5), xlab = "Empirical quantile definition", ylab = "Quantile
value")
abline(h=qlnorm(0.05,2,0.3),col="blueviolet", lwd=4, lty=2)

# Type = 4
boxplot(t(df_5), xlab = "Empirical quantile definition", ylab = "Quantile
value")
abline(h=qgumbel(0.05,7,0.5, max=FALSE),col="dodgerblue4", lwd=3, lty=2)

#Parametric quantile estimation

#read in the data and split into MOR1 and MOR2 datasets
setwd("C:/Users/Jarod/Google Drive/Jarod M- 2018 and 2019/Data
set/AC/Data")
MOR <- read.csv('MOR_Data.csv',sep = ',')
MOR1 <- MOR[MOR$Sample == 'MOR1',2]
MOR2 <- MOR[MOR$Sample == 'MOR2',2]

DS <- MOR1
#Fit of distributions by maximum likelihood estimation
library("fitdistrplus")

fw <- fitdist(DS, "weibull")
fg <- fitdist(DS, "gamma")
fln <-  fitdist(DS, "lnorm")

dgumbel <- function(x, a, b) 1/b * exp((x-a)/b) * exp(-exp((x-a)/b))
pgumbel <- function(q, a, b) 1-exp(-exp((q-a)/b))
qgumbel <- function(p, a, b) a + b * log(-log(1-p))
fgu <- fitdist(DS, "gumbel", start = list(a = 7.5, b = 1.5))

summary(fw)
summary(fg)
summary(fln)
summary(fgu)

#Plotting different distributions
```

```
plot.legend <- c("Weibull", "lognormal", "gamma", "Gumbel")
cdfcomp(list(fw, fln, fg, fgu), legendtext = plot.legend, xlegend =
"bottomright", main ="",lwd=2, xlim = c(1,15))
qqcomp(list(fw, fln, fg, fgu), legendtext = plot.legend, xlegend =
"topleft", main = "")

x=seq(0, max(DS), length = 1000)
hist(DS,  breaks=25, prob=TRUE, main="", xlab = "MOR1")
curve(dweibull(x, fw$estimate[1], fw$estimate[2]), col="red", lwd=3, add=T,
lty = 'dashed')
curve(dgamma(x, fg$estimate[1], fg$estimate[2]), col="blue", lwd=3, add=T)
curve(dlnorm(x, fln$estimate[1], fln$estimate[2]), col="green", lwd=3,
add=T, lty = 'dotdash' )
curve(dgumbel(x, fgu$estimate[1], fgu$estimate[2]), col="purple", lwd=3,
add=T, lty = 'twodash')
legend( x = c(6.6,9), y = c(0.415, 0.415), legend=c("Weibull", "lognormal",
"gamma", "Gumbel"),
        col=c("red", "blue", "green", "purple"), lty=1:4, cex=1.2,
box.lty=0, y.intersp=0.3)


#Fit statistics
gofstat(list(fw, fln, fg, fgu),
        fitnames = c("weibull", "lognormal", "gamma", "Gumbel"))

#Quantile estimates
wq5 <- quantile(fw, probs = 0.01)
lnq5 <- quantile(fln, probs = 0.01)
gq5 <- quantile(fg, probs = 0.01)
guq5 <- quantile(fgu, probs = 0.01)

quantile(fw, probs = 0.01)
quantile(fln, probs = 0.01)
quantile(fg, probs = 0.01)
quantile(fgu, probs = 0.01)

################################
####    Chapter 3 code    ####
################################

## censored parameter estimation
setwd("C:\\Users\\Jarod\\Google Drive\\Jarod M- 2018 and 2019\\Data
set\\AC\\Data")
# Read CSV into R
MyData <- read.csv(file="MOR_Data.csv", header=TRUE, sep=",")
MOR1 <- MyData[MyData$Sample == 'MOR1',2]
MOR2 <- MyData[MyData$Sample == 'MOR2',2]
DS <- MOR1
C <- quantile(DS, probs = 0.1, type = 3)
MOR1_ordered <- DS[order(DS)]
x <- MOR1_ordered[(MOR1_ordered<=C)]
m <- length(x)
n <- length(DS)


fr <- function(param) {   ## Censored Weibull
  beta <- param[1]
  psi <- param[2]
  (m*log(beta) + m*log(psi) + (beta-1)*sum(log(x)) - psi*sum(x^beta) - (n-
m)*psi*C^beta)*(-1)
}
```

```
optim(c(0.5,0.5), fr)


fr <- function(param) {    ## Censored Gumbel
  mu <- param[1]
  beta <- param[2]
  (-m*log(beta)+sum(x-mu)/beta - sum(exp((x-mu)/beta))-(n-m)*exp((C-
mu)/beta))*(-1)
}

optim(c(1,1), fr)

fr <- function(param) {    ## Censored logNormal
  mu <- param[1]
  sigma <- param[2]
  (-(m/2)*log(2*pi*sigma^2) - sum(log(x)) - sum(log(x)^2)/(2*sigma^2) +
sum(log(x)*mu)/sigma^2 -
      m*(mu^2)/(2*sigma^2) + (n-m)*(1-pnorm((log(C)-mu)/sigma)))*(-1)
}

optim(c(1,1), fr)

library(astro)
fr <- function(param) {    ## Censored Gamma
  theta <- param[1]
  kappa <- param[2]
  (-m*log(gamma(theta)) - m*theta*log(kappa) +(theta-1)*sum(log(x)) -
sum(x)/kappa
    +(n-m)*log(1-pgamma(C,theta,1/kappa)))*(-1)
}

optim(c(0.2,5), fr)

dgumbel <- function(x, a, b) 1/b * exp((x-a)/b) * exp(-exp((x-a)/b))
pgumbel <- function(q, a, b) 1-exp(-exp((q-a)/b))
qgumbel <- function(p, a, b) a + b * log(-log(1-p))


library(shape)
x=seq(0, max(DS), length = 1000)
hist(DS,  breaks=25, prob=TRUE, main="", ylim = c(0,0.38), xlab = "MOR1")
curve(dweibull(x, 7.378, 6.739), col="red", lwd=3, add=T, lty = 1)
curve(dgamma(x, 16.168 , 1/0.440), col="blue", lwd=3, add=T, lty = 2)
curve(dlnorm(x, 1.951, 0.286), col="green", lwd=3, add=T, lty = 3 )
curve(dgumbel(x, 6.319, 0.601), col="purple", lwd=3, add=T, lty = 4)
Arrows(C,0,C,0.0001, arr.type = "triangle",col="gray29", lwd =0.1)
abline(v=C, col="gray29", lwd =2)
legend(7, 0.4, legend=c("Weibull", "lognormal", "gamma", "Gumbel",
"Threshold"),
       col=c("red", "blue", "green", "purple", "gray29"),
       cex=1, box.lty=0, y.intersp=0.35, bg="transparent", lty = 1:4,
x.intersp = 0.3)

#Truncated Kolmogoriv-Smirnov test weibull
index_C <- which(MOR1_ordered == C)
ecdf <- ecdf(DS)
wdf <- pweibull(DS, 6.823, 7.172, lower.tail = TRUE, log.p = FALSE)
max(abs(ecdf[1:index_C]-wdf[1:index_C]))
```

```
plot(MOR2,ecdf, xlab = 'Sample Quantiles of MOR2', ylab = '', main = '',
ylim = c(0,1))
mtext(bquote(tilde(F)[n](x) ~ ~vs~ ~ hat(F)[n](x)),side = 2, line = 2.5)
abline(h = 1, col = "red", lty=2)
lines(MOR2,wdf,col='blue', lwd = 2)
Arrows(C,0,C,0.0001, arr.type = "triangle",col="gray29", lwd =0.1)
abline(v=C, col="gray29", lwd =2)

#Truncated Kolmogoriv-Smirnov test gamma
gdf <- pgamma(DS, 12.96,  1/0.599, lower.tail = TRUE, log.p = FALSE)
max(abs(ecdf[1:index_C]-gdf[1:index_C]))

plot(MOR2,ecdf, xlab = 'Sample Quantiles of MOR2', ylab = '', main = '',
ylim = c(0,1))
mtext(bquote(tilde(F)[n](x) ~ ~vs~ ~ hat(F)[n](x)),side = 2, line = 2.5)
abline(h = 1, col = "red", lty=2)
lines(MOR2,gdf,col='blue', lwd = 2)
Arrows(C,0,C,0.0001, arr.type = "triangle",col="gray29", lwd =0.1)
abline(v=C, col="gray29", lwd =2)

#Truncated Kolmogoriv-Smirnov test log-normal
lndf <- plnorm(DS, 2.072, 0.336, lower.tail = TRUE, log.p = FALSE)
max(abs(ecdf[1:index_C]-lndf[1:index_C]))

plot(MOR2,ecdf, xlab = 'Sample Quantiles of MOR2', ylab = '', main = '',
ylim = c(0,1))
mtext(bquote(tilde(F)[n](x) ~ ~vs~ ~ hat(F)[n](x)),side = 2, line = 2.5)
abline(h = 1, col = "red", lty=2)
lines(MOR2,lndf,col='blue', lwd = 2)
Arrows(C,0,C,0.0001, arr.type = "triangle",col="gray29", lwd =0.1)
abline(v=C, col="gray29", lwd =2)

#Truncated Kolmogoriv-Smirnov test Gumbel
gudf <- pgumbel(DS, 6.623, 0.651)
max(abs(ecdf[1:index_C]-gudf[1:index_C]))

plot(MOR2,ecdf, xlab = 'Sample Quantiles of MOR2', ylab = '', main = '',
ylim = c(0,1))
mtext(bquote(tilde(F)[n](x) ~ ~vs~ ~ hat(F)[n](x)),side = 2, line = 2.5)
abline(h = 1, col = "red", lty=2)
lines(MOR2,gudf,col='blue', lwd = 2)
Arrows(C,0,C,0.0001, arr.type = "triangle",col="gray29", lwd =0.1)
abline(v=C, col="gray29", lwd =2)


#Quantile estimates for censored distributions
qgumbel <- function(p, a, b) a + b * log(-log(1-p))
qweibull(0.05, 6.823130, 7.173, lower.tail = TRUE, log.p = FALSE)
qlnorm(0.05, 2.0442105, 0.3296024, lower.tail = TRUE, log.p = FALSE)
qgamma(0.05, 12.959570,  1/0.599397, lower.tail = TRUE,
       log.p = FALSE)
qgumbel(0.05, 6.6232032, 0.6505291)

qgumbel <- function(p, a, b) a + b * log(-log(1-p))
qweibull(0.05, 7.377541, 6.738, lower.tail = TRUE, log.p = FALSE)
qlnorm(0.05, 1.9514936, 0.2855398, lower.tail = TRUE, log.p = FALSE)
qgamma(0.05, 16.1675782,  1/0.4403953, lower.tail = TRUE,
       log.p = FALSE)
qgumbel(0.05, 6.3192829, 0.6011575)

#Standard error
```

```
library(ordinal)
MC       <- 10000
set.seed(1234)

fr_w <- function(param) {   ## Censored Weibull
  beta <- param[1]
  psi <- param[2]
  (m*log(beta) + m*log(psi) + (beta-1)*sum(log(x)) - psi*sum(x^beta) - (n-
m)*psi*C^beta)*(-1)
}

library(astro)
fr_g <- function(param) {   ## Censored Gamma
  theta <- param[1]
  kappa <- param[2]
  (-m*log(gamma(theta)) - m*theta*log(kappa) +(theta-1)*sum(log(x)) -
sum(x)/kappa
    +(n-m)*log(1-pgamma(C,theta,1/kappa)))*(-1)
}


results  <- matrix(nrow = MC, ncol = 8)
for (j in 1:nrow(results)){
  w             = rweibull(300, shape = 6.823130, scale = 7.173)
  C             = quantile(w, probs = 0.1, type = 3)
  MOR1_ordered = w[order(w)]
  x             = MOR1_ordered[(MOR1_ordered<=C)]
  m             = length(x)
  n             = length(w)
  par           = optim(c(0.5,0.5), fr_w)
  results[j,1] = qweibull(0.05, par$par[1], (1/par$par[2])^(1/par$par[1]))

  w             = rweibull(300, shape = 7.377541, scale = 6.738)
  C             = quantile(w, probs = 0.1, type = 3)
  MOR1_ordered = w[order(w)]
  x             = MOR1_ordered[(MOR1_ordered<=C)]
  m             = length(x)
  n             = length(w)
  par           = optim(c(0.5,0.5), fr_w)
  results[j,5] = qweibull(0.05, par$par[1], (1/par$par[2])^(1/par$par[1]))

  g             = rgamma(300, shape = 12.959570, scale = 0.599397)
  C             = quantile(g, probs = 0.1, type = 3)
  MOR1_ordered = g[order(g)]
  x             = MOR1_ordered[(MOR1_ordered<=C)]
  m             = length(x)
  n             = length(g)
  par           = optim(c(0.2,0.5), fr_g )
  results[j,2] = qgamma(0.05, par$par[1], (1/par$par[2]))

  g             = rgamma(300, shape = 16.1675782, scale = 0.4403953)
  C             = quantile(g, probs = 0.1, type = 3)
  MOR1_ordered = g[order(g)]
  x             = MOR1_ordered[(MOR1_ordered<=C)]
  m             = length(x)
  n             = length(g)
  par           = optim(c(0.2,0.5), fr_g )
  results[j,6] = qgamma(0.05, par$par[1], (1/par$par[2]))

  ln            = rlnorm(300, 2.0442105, 0.3296024)
```

```
  C            = quantile(ln, probs = 0.1, type = 3)
  MOR1_ordered = ln[order(ln)]
  x            = MOR1_ordered[(MOR1_ordered<=C)]
  m            = length(x)
  n            = length(ln)
  par          = optim(c(1,1), fr_ln )
  results[j,3] = qlnorm(0.05, par$par[1], par$par[2])

  ln           = rlnorm(300, 1.9514936, 0.2855398)
  C            = quantile(ln, probs = 0.1, type = 3)
  MOR1_ordered = ln[order(ln)]
  x            = MOR1_ordered[(MOR1_ordered<=C)]
  m            = length(x)
  n            = length(ln)
  par          = optim(c(1,1), fr_ln )
  results[j,7] = qlnorm(0.05, par$par[1], par$par[2])

  gu           = rgumbel(300, location = 6.6232032, scale =0.6505291 , max
= FALSE)
  C            = quantile(gu, probs = 0.1, type = 3)
  MOR1_ordered = gu[order(gu)]
  x            = MOR1_ordered[(MOR1_ordered<=C)]
  m            = length(x)
  n            = length(gu)
  par          = optim(c(1,1), fr_gu )
  results[j,4] = qgumbel(0.05, location = par$par[1], scale = par$par[2],
max = FALSE)

  gu           = rgumbel(300, location = 6.3192829, scale = 0.6011575 , max
= FALSE)
  C            = quantile(gu, probs = 0.1, type = 3)
  MOR1_ordered = gu[order(gu)]
  x            = MOR1_ordered[(MOR1_ordered<=C)]
  m            = length(x)
  n            = length(gu)
  par          = optim(c(1,1), fr_gu )
  results[j,8] = qgumbel(0.05, location = par$par[1], scale = par$par[2],
max = FALSE)
}

for(i in 1:ncol(results)){
  results[is.na(results[,i]), i] <- mean(results[,i], na.rm = TRUE)
}

SE <- matrix(nrow = 8, ncol = 1)
SE <- apply(results,2,sd)

# ANOVA test difference in quantiles

dat <- data.frame(
  y = c(results[,5],results[,6],results[,7],results[,8]),
  groups = sort(rep(LETTERS[1:4], length(results[,1])))
)

boxplot(y ~ groups, dat, names=c("Weibull","Gamma","Log-normal", "Gumbel"),
medcol="red")
par(cex.axis=0.5)
summary(aov(y ~ groups, dat))

## EM mixtures
```

```
setwd("C:\\Users\\Jarod\\Google Drive\\Jarod M- 2018 and 2019\\Data
set\\AC\\Data")
# Read CSV into R
MyData <- read.csv(file="MOR_Data.csv", header=TRUE, sep=",")
MOR1 <- MyData[MyData$Sample == 'MOR1',2]
MOR2 <- MyData[MyData$Sample == 'MOR2',2]

DS <- MOR1
#EM algorithm for Weibull mixtures

em_w <- function(DS) {

  #initialise mixing probabilities to be equal
  probs <- 0.5
  #initialise parameters
  alpha <- c(6.823130, 6.823130 + rnorm(1,0,1))
  eta <- c(7.173, 7.173 + rnorm(1,0,1))

  diff = 1e+11

  while(diff > 0.001)
  {
    #likelihood function
    weibull <- cbind(dweibull(DS, alpha[1], eta[1], log = FALSE),
dweibull(DS, alpha[2], eta[2], log = FALSE))

    #responsibilities
    gam <- (probs*weibull[,1])/(probs*weibull[,1] + (1-probs)*weibull[,2])

    #estimate parameters
    alpha_new <- cbind(sum(gam*(DS^alpha[1])*log(DS))/sum(gam*DS^alpha[1])
- sum(gam*log(DS))/sum(gam),
                       sum((1-gam)*(DS^alpha[2])*log(DS))/sum((1-
gam)*DS^alpha[2]) - sum((1-gam)*log(DS))/sum((1-gam)))

    eta_new <-
cbind((sum(gam*DS^(1/alpha_new[1]))/sum(gam))^(alpha_new[1]),
                     (sum((1-gam)*DS^(1/alpha_new[2]))/sum((1-
gam)))^(alpha_new[2]))

    probs_new <- sum((1-gam))/length(DS)

    #update parameters
    diff <- sum(abs((alpha - 1/alpha_new)) + abs((eta - eta_new)) +
abs((probs - probs_new)))
    probs <- probs_new
    alpha <- 1/alpha_new
    eta <- eta_new

    diff <- ifelse(alpha[1]>=60, 10, ifelse(alpha[2]>=60, 10, diff))
  }

  return(list(probs,alpha, eta, gam))
}

#find quantiles for mixture distribution
f <- function (x){
  probs*(1-exp(-(x/eta[1])^alpha[1])) + (1-probs)*(1-exp(-
(x/eta[2])^alpha[2])) - 0.05
}
```

```
param_w <- em_w(DS)

f <- function (x){
  param[[1]]*pweibull(x, param[[2]][1], param[[3]][1], lower.tail = TRUE,
log.p = FALSE) +
    (1- param[[1]])*pweibull(x, param[[2]][2], param[[3]][2], lower.tail =
TRUE, log.p = FALSE) - 0.05
}

q_05 <- uniroot(f, lower = 0, upper = 20)


##########################################################################
#####################################
#EM algorithm for Normal mixtures

em_n <- function(DS) {

  #initialise mixing probabilities to be equal
  probs <- 0.5
  #initialise parameters
  u <- c(mean(DS), mean(DS) + rnorm(1,0,1))
  sig <- c(var(DS), var(DS) + rnorm(1,0,1))

  diff = 1e+11

  probs <- 0.5
  #initialise parameters
  u <- c(mean(DS), mean(DS) + rnorm(1,0,1))
  sig <- c(var(DS), var(DS) + rnorm(1,0,1))

  while(diff > 0.001)
  {
    #likelihood function
    normal <- cbind(dnorm(DS, u[1], sqrt(sig[1]), log = FALSE), dnorm(DS,
u[2], sqrt(sig[2]), log = FALSE))

    #responsibilities
    gam <- cbind((probs*normal[,1])/(probs*normal[,1] + (1-
probs)*normal[,2]),
                 ((1-probs)*normal[,2])/(probs*normal[,1] + (1-
probs)*normal[,2]))

    #estimate parameters
    u_new <- cbind(sum(gam[,1]*DS)/sum(gam[,1]),
sum(gam[,2]*DS)/sum(gam[,2]))

    sig_new <- cbind(sum(gam[,1]*(DS-u_new[,1])^2)/sum(gam[,1]),
                     sum(gam[,2]*(DS-u_new[,2])^2)/sum(gam[,2]))

    probs_new <- sum(gam[,1])/length(DS)

    #update parameters
    diff <- sum(sum(abs(u - u_new)) + sum(abs(sig - sig_new)) +
sum(abs(probs - probs_new)))
    probs <- probs_new
    u <- u_new
    sig <- sig_new
  }

  return(list(probs,u,sqrt(sig)))
```

```
}

#find quantiles for mixture distribution
param_n <- em_n(DS)

f <- function (x){
  param[[1]]*pnorm(x, param[[2]][1], sqrt(param[[3]][1]), lower.tail =
TRUE, log.p = FALSE) +
    (1- param[[1]])*pnorm(x, param[[2]][2], sqrt(param[[3]][2]), lower.tail
= TRUE, log.p = FALSE) - 0.05
}

q_05 <- uniroot(f, lower = 0, upper = 20)
test <- normalmixEM(DS, lambda = .5, mu = c(1, 1), sigma = c(0.6,0.5))
summary(test)

##########################################################################
#####################################
#EM algorithm for log-Normal mixtures

em_ln <- function(DS) {

  #initialise mixing probabilities to be equal
  probs <- 0.5
  #initialise parameters
  u <- c(log(mean(DS)), log(mean(DS) + rnorm(1,0,1)))
  sig <- c(log(var(DS)), log(var(DS) + rnorm(1,0,1)))

  diff = 1e+11

  while(diff > 0.001)
  {
    #likelihood function
    log_normal <- cbind(dlnorm(DS, u[1], sqrt(sig[1]), log = FALSE),
dlnorm(DS, u[2], sqrt(sig[2]), log = FALSE))

    #responsibilities
    gam <- cbind((probs*log_normal[,1])/(probs*log_normal[,1] + (1-
probs)*log_normal[,2]),
                 ((1-probs)*log_normal[,2])/(probs*log_normal[,1] + (1-
probs)*log_normal[,2]))

    #estimate parameters
    u_new <- cbind(sum(gam[,1]*log(DS))/sum(gam[,1]),
sum(gam[,2]*log(DS))/sum(gam[,2]))

    sig_new <- cbind(sum(gam[,1]*(log(DS)-u_new[,1])^2)/sum(gam[,1]),
                     sum(gam[,2]*(log(DS)-u_new[,2])^2)/sum(gam[,2]))

    probs_new <- sum(gam[,1])/length(DS)

    #update parameters
    diff <- sum(sum(abs(u - u_new)) + sum(abs(sig - sig_new)) +
sum(abs(probs - probs_new)))
    probs <- probs_new
    u <- u_new
    sig <- sig_new
  }

  return(list(probs,u,sqrt(sig)))
}
```

```
#find quantiles for mixture distribution
param_ln <- em_ln(DS)

f <- function (x){
  param[[1]]*plnorm(x, param[[2]][1], sqrt(param[[3]][1]), lower.tail =
TRUE, log.p = FALSE) +
    (1- param[[1]])*plnorm(x, param[[2]][2], sqrt(param[[3]][2]),
lower.tail = TRUE, log.p = FALSE) - 0.05
}

q_05 <- uniroot(f, lower = 0, upper = 20)

##############################################################################
##############################################
#use for distributions
fr <- function(param) {
  alpha_1 <- param[1]
  eta_1 <- param[2]
  alpha_2 <- param[3]
  eta_2 <- param[4]
  (sum(gam[,1]*log(probs*(dweibull(DS, alpha_1, eta_1, log = FALSE))) +
gam[,2]*log((1-probs)*(dweibull(DS, alpha_2, eta_2, log = FALSE)))))*(-1)
}

nr <- optim(c(5,  5, 5,  5.), fr)
alpha_new <- cbind(nr$par[1], nr$par[3])
eta_new <- cbind(nr$par[2], nr$par[4])


#find quantiles for mixture distribution
f <- function (x){
  probs*(1-exp(-(x/eta[1])^alpha[1])) + (1-probs)*(1-exp(-
(x/eta[2])^alpha[2])) - 0.05
}

q_05 <- uniroot(f, lower = 0, upper = 20)

##############################################################################
###################################
x=seq(0, max(DS), length = 1000)
hist(DS,  breaks=15, prob=TRUE, main="", ylim = c(0,0.4), xlab="MOR1")
curve(param_w[[1]]*dweibull(x, param_w[[2]][1], param_w[[3]][1]) +
        (1-param_w[[1]])*dweibull(x, param_w[[2]][2], param_w[[3]][2]) ,
col="firebrick4", lwd=2, add=T, lty = 'dashed')
curve(param_n[[1]]*dnorm(x, param_n[[2]][1], param_n[[3]][1]) +
        (1-param_n[[1]])*dnorm(x, param_n[[2]][2], param_n[[3]][2]) ,
col="deepskyblue4", lwd=2, add=T, lty = 'solid')
curve(param_ln[[1]]*dlnorm(x, param_ln[[2]][1], param_ln[[3]][1]) +
        (1-param_ln[[1]])*dlnorm(x, param_ln[[2]][2], param_ln[[3]][2]) ,
col="forestgreen", lwd=2, add=T, lty = 'twodash')
legend(8.1, 0.43, legend=c("Weibull", "Normal", "Log-normal"),
       col=c("firebrick4", "deepskyblue4", "forestgreen"), lty=1:4,
cex=0.8, box.lty=0)

par  <- param_w
x=seq(0, max(DS), length = 1000)
hist(DS,  breaks=15, prob=TRUE, main="", ylim = c(0,0.4), xlab="MOR1")
curve(par[[1]]*dweibull(x, par[[2]][1], par[[3]][1]) + (1-
par[[1]])*dweibull(x, par[[2]][2], par[[3]][2]) , col="firebrick4", lwd=3,
add=T, lty = 1)
```

```
curve(dweibull(x, par[[2]][1], par[[3]][1]) , col="deepskyblue4", lwd=3,
add=T, lty = 2)
curve(dweibull(x, par[[2]][2], par[[3]][2]) , col="forestgreen", lwd=3,
add=T, lty = 3)
legend(8.1, 0.43, legend=c("Mixture", "Majority", "Minority"),
       col=c("firebrick4", "deepskyblue4", "forestgreen"), lty=1:3,
cex=0.8, box.lty=0)


x=seq(0, max(MOR1), length = 1000)
hist(DS,  breaks=15, prob=TRUE, main="", ylim = c(0,0.4), xlab="MOR1")
curve(0.7448*dweibull(x, 5.494, 7.599) +
        (1-0.7448)*dweibull(x, 15.81, 5.983) , col="firebrick4", lwd=3,
add=T, lty = 4)
curve(0.5629*dnorm(x, 5.953, 0.970) +
        (1-0.5629)*dnorm(x, 7.676, 1.215) , col="deepskyblue4", lwd=3,
add=T, lty = 5)
curve(0.9758*dlnorm(x, 1.897, 0.189) +
        (1-0.9758)*dlnorm(x, 1.245, 0.102) , col="forestgreen", lwd=3,
add=T, lty = 6)
legend(6,0.45, legend=c("Weibull Mixture", "Normal Mixture", "Log-normal
Mixture"),
       col=c("firebrick4", "deepskyblue4", "forestgreen"), lty=4:6,
cex=1.2, box.lty=0,
       x.intersp = 0.3,y.intersp=0.35, bg="transparent")

x=seq(0, max(MO2), length = 1000)
hist(DS,  breaks=15, prob=TRUE, main="", ylim = c(0,0.4), xlab="MOR2")
curve(0.7932*dweibull(x, 5.427, 7.642) +
        (1-0.7932)*dweibull(x, 12.01, 6.186) , col="firebrick4", lwd=3,
add=T, lty = 4)
curve(0.5406*dnorm(x, 5.924, 1.042) +
        (1-0.5629)*dnorm(x, 7.859, 1.095) , col="deepskyblue4", lwd=3,
add=T, lty = 5)
curve(0.6649*dlnorm(x, 1.976, 0.167) +
        (1-0.6649)*dlnorm(x, 1.736, 0.226) , col="forestgreen", lwd=3,
add=T, lty = 6)
legend(6,0.45, legend=c("Weibull Mixture", "Normal Mixture", "Log-normal
Mixture"),
       col=c("firebrick4", "deepskyblue4", "forestgreen"), lty=4:6,
cex=1.2, box.lty=0,
       x.intersp = 0.3,y.intersp=0.35, bg="transparent")

library(shape)
x=seq(0, max(MOR1), length = 1000)
hist(MOR1,  breaks=15, prob=TRUE, main="", ylim = c(0,0.4))
curve(0.7446*dweibull(x, 5.495, 7.599) + (1-0.7446)*dweibull(x, 15.805,
5.983) , col="firebrick4", lwd=3, add=T, lty = 1)
curve(dweibull(x, 5.495, 7.599) , col="deepskyblue4", lwd=3, add=T, lty =
2)
curve(dweibull(x, 15.805, 5.983) , col="forestgreen", lwd=3, add=T, lty =
3)
#Arrows(C,0,C,0.0001, arr.type = "triangle",col="gray29", lwd =2, lty = 1)
abline(v=C, col="gray29", lwd =2)
legend(6,0.45, legend=c("Mixture", "Majority", "Minority"),
       col=c("firebrick4", "deepskyblue4", "forestgreen", "gray29"),
lty=c(1,2,3), cex=1.2, box.lty=0,
       x.intersp = 0.3,y.intersp=0.35, bg="transparent")

library(shape)
x=seq(0, max(MOR1), length = 1000)
```

```
hist(MOR2,  breaks=15, prob=TRUE, main="", ylim = c(0,0.4))
curve(0.7943*dweibull(x, 5.425, 7.646) + (1-0.7943)*dweibull(x, 11.992,
6.173) , col="firebrick4", lwd=3, add=T, lty = 1)
curve(dweibull(x, 5.425, 7.646) , col="deepskyblue4", lwd=3, add=T, lty =
2)
curve(dweibull(x, 11.992, 6.173) , col="forestgreen", lwd=3, add=T, lty =
3)
#Arrows(C,0,C,0.0001, arr.type = "triangle",col="gray29", lwd =2, lty = 1)
abline(v=C, col="gray29", lwd =2)
legend(6,0.45, legend=c("Mixture", "Majority", "Minority"),
       col=c("firebrick4", "deepskyblue4", "forestgreen", "gray29"),
lty=c(1,2,3), cex=1.2, box.lty=0,
       x.intersp = 0.3,y.intersp=0.35, bg="transparent")
#Bootstrap homogeneity test
#Reliability of p-value study

library(fitdistrplus)
library(foreach)
library(doParallel)
em_w <- function(DS) {
  count <- 0
  #initialise mixing probabilities to be equal
  probs <- 0.5
  #initialise parameters
  alpha <- c(6.823130, 6.823130 + rnorm(1,0,1))
  eta <- c(7.173, 7.173 + rnorm(1,0,1))

  diff = 1e+11

  while(diff > 0.01)
  {
    #likelihood function
    weibull <- cbind(dweibull(DS, alpha[1], eta[1], log = FALSE),
dweibull(DS, alpha[2], eta[2], log = FALSE))

    #responsibilities
    gam <- (probs*weibull[,1])/(probs*weibull[,1] + (1-probs)*weibull[,2])

    #estimate parameters
    alpha_new <- cbind(sum(gam*(DS^alpha[1])*log(DS))/sum(gam*DS^alpha[1])
- sum(gam*log(DS))/sum(gam),
                       sum((1-gam)*(DS^alpha[2])*log(DS))/sum((1-
gam)*DS^alpha[2]) - sum((1-gam)*log(DS))/sum((1-gam)))

    eta_new <-
cbind((sum(gam*DS^(1/alpha_new[1]))/sum(gam))^(alpha_new[1]),
                     (sum((1-gam)*DS^(1/alpha_new[2]))/sum((1-
gam)))^(alpha_new[2]))

    probs_new <- sum((1-gam))/length(DS)

    #update parameters
    diff  <- sum(abs((alpha - 1/alpha_new)) + abs((eta - eta_new)) +
abs((probs - probs_new)))
    probs <- probs_new
    alpha <- 1/alpha_new
    eta   <- eta_new

    diff  <- ifelse(alpha[1]>=40, 1, ifelse(alpha[2]>=40, 1, diff))
    count <- count+1
    diff  <- ifelse(count>=70,0.001,diff)
```

```
    diff  <- ifelse(is.na(diff), 0.001, diff)
  }

  return(list(probs,alpha, eta, gam, count))
}

out_fit <- function(n, shape, scale) {
  #Generate the data set
  dat <-rweibull(n, shape, scale)

  #Obtain MLE estimates for single and mixture Weibull

  ##Single
  fit_single    <- fitdist(dat, "weibull")
  loglik_single <- fit_single$loglik
  MLE_single    <- fit_single$estimate

  #Mixture
  fit_mix       <- em_w(dat)
  loglik_mix    <-
sum(fit_mix[[4]][1]*log(fit_mix[[1]]*dweibull(dat,fit_mix[[2]][1],
fit_mix[[3]][1]))
                       + (1-fit_mix[[4]][1])*log((1-
fit_mix[[1]])*dweibull(dat,fit_mix[[2]][2], fit_mix[[3]][2])))

  #Likelihood ratio 1
  lamda         <- loglik_single-loglik_mix
  L1            <- -2*lamda
  return(list(MLE_single, L1))
}
in_fit  <- function(n,B) {

  L1      <- out_fit(n, shape, scale) [[2]][1]
  shape_2 <- out_fit(n, shape, scale) [[1]][1]
  scale_2 <- out_fit(n, shape, scale) [[1]][2]
  ind_sum <- 0

  for(j in 1:B) {

    dat2              <- rweibull(n, shape_2, scale_2)

    ##Single
    fit_single_2      <- fitdist(dat2, "weibull")
    loglik_single_2   <- fit_single_2$loglik

    #Mixture
    fit_mix_2         <- em_w(dat2)
    loglik_mix_2      <-
sum(fit_mix_2[[4]][1]*log(fit_mix_2[[1]]*dweibull(dat2,fit_mix_2[[2]][1],
fit_mix_2[[3]][1]))
                              + (1-fit_mix_2[[4]][1])*log((1-
fit_mix_2[[1]])*dweibull(dat2,fit_mix_2[[2]][2], fit_mix_2[[3]][2])))
    #Likelihood ratio
    lamda_2           <- loglik_single_2-loglik_mix_2
    L2                <- -2*lamda_2
    test              <- (L2 > L1)
    test              <- ifelse(is.na(test),0,test)
    ind_sum           <- ind_sum + test

  }
```

```
  return(ind_sum/B)
}

cores=detectCores()
cl <- makeCluster(cores[1]-1)
registerDoParallel(cl)
on.exit(stopCluster(cl))

B     <- 500
N     <- 500
n     <- 300
shape <- 7
scale <- 7

system.time({

  p_value  <- foreach(i = 1:N, .packages = 'fitdistrplus',.combine = rbind)
%dopar% {
    in_fit(n,B)
  }
  stopCluster(cl)
})


p_value <- p_value[order(p_value)]
x <-runif(10000,0,1)


##Plot empirical cdf of p values
##against cdf of uniform(0, 1)
t = ecdf(x[order(x)])
p_value.cdf <- ecdf(p_value[order(p_value)])
plot(p_value.cdf, xlab = 'p-value', ylab = '', main = '', lty=1, col =
'black')
lines(t, lty=1, col = 'red', lwd=2)
mtext(text = expression(hat(F)[n](x)), side = 2, line = 2.5)
legend(x = 'topleft', legend=c("Empirical CDF of p-values", "Uniform (0,
1)"),
       col=c("black", "red"), lty=1:2, cex=1.2, box.lty=0, x.intersp =
0.3,y.intersp=0.35, bg="transparent")


#KS test
ks_test <- ks.test(t,p_value.cdf)
ks_test$statistic
ks_test$p.value

## score functions
#Fit of distributions by maximum likelihood estimation
library("fitdistrplus")
Data <- rweibull(500, 7,7)


#OMLE
fw <- fitdist(Data, "weibull")
param_o <- fw$estimate

#CMLE
C <- quantile(Data, probs = 0.1, type = 3)
Data_ordered <- Data[order(Data)]
```

```
x <- Data_ordered[(Data_ordered<=C)]
m <- length(x)
n <- length(Data)


fr <- function(param) {    ## Censored Weibull
  beta <- param[1]
  psi <- param[2]
  (m*log(beta) + m*log(psi) + (beta-1)*sum(log(x)) - psi*sum(x^beta) - (n-
m)*psi*C^beta)*(-1)
}

p <- optim(c(0.5,0.5), fr)
param_c <- c(p$par[1],(1/p$par[2])^(1/p$par[1]))

#score functions OMLE
o_s_beta <- function(x) (-(x/param_o[2])^param_o[1]*log(x/param_o[1]) +
                           log(x/param_o[2]) + 1/param_o[1])
o_s_eta <- function(x) (-param_o[1]/param_o[2] +
(param_o[1]*x^param_o[1])/param_o[2]^(param_o[1]+1))

#score functions CMLE
c_s_beta <- function(x) {
  y <- x[order(x)]
  (y<=C)*(-(y[(y<=C)]/param_o[2])^param_o[1]*log(y[(y<=C)]/param_o[1]) +
            log(y[(y<=C)]/param_o[2]) + 1/param_o[1]) +
    (y>C)*(-(C/param_o[2])^param_o[1]*log(C/param_o[2]))
}


c_s_eta <- function(x) {
  y <- x[order(x)]
  (y<=C)*(-param_o[1]/param_o[2] +
(param_o[1]*y[(y<=C)]^param_o[1])/param_o[2]^(param_o[1]+1)) +
    (y>C)*(param_o[1]*C^param_o[1])/param_o[2]^(param_o[1]+1)
}


#Plots
library(shape)
x=seq(min(Data),10,length = 100)
curve(-c_s_beta(x), col="deepskyblue4", lwd=3, lty = 1,xlim = c(min(x),
max(x)), ylim = c(-0.25,2), ylab = "")
curve(-o_s_beta(x), col="firebrick4", lwd=3, lty = 2,add=T, xlim = c(C,
max(x)), ylab = "")
Arrows(C,-0.3,C,-0.21, arr.type = "triangle",col="gray29", lwd =0.1)
mtext(bquote(-psi[beta](x)),side = 2, line = 2.5)
legend(4, 2, legend=c("Censored Weibull", "Ordinary Weibull","Threshold"),
       col=c("deepskyblue4", "firebrick4", "gray29"), lty=c(1,2,1),cex=1.2,
box.lty=0,
       x.intersp = 0.3,y.intersp=0.35, bg="transparent")
abline(v=C, col="gray29", lwd =3, lty = 1)
abline(h=0, lty = 3, lwd=3 )

x=seq(min(Data),10,length = 100)
curve(c_s_eta(x), col="deepskyblue4", lwd=3, lty = 1,xlim = c(min(x),
max(x)), ylim = c(-1.2,10), ylab = "")
curve(o_s_eta(x), col="firebrick4", lwd=3, lty = 2,add=T, xlim = c(C,
max(x)), ylab = "")
Arrows(C,-2,C,-1.89, arr.type = "triangle",col="gray29", lwd =0.1)
mtext(bquote(psi[eta](x)),side = 2, line = 2.5)
```

```
legend(4, 10, legend=c("Censored Weibull", "Ordinary Weibull","Threshold"),
       col=c("deepskyblue4", "firebrick4", "gray29"), lty=c(1,2,1),
cex=1.2, box.lty=0,
       x.intersp = 0.3,y.intersp=0.35, bg="transparent")
abline(v=C, col="gray29", lwd =3, lty = 1)
abline(h=0, lty = 3,lwd=3)

## CMLE simulation

## AKSMLE RMSE of quantiles for all models
#MOR1
library(data.table)
library(future)
library(ordinal)
library(NLRoot)
library(fitdistrplus)

CMLE <- function(N,q,rdens,qdens,pdens,param,mix.ind,gum.ind) {
  bisection <- function(f, a, b, n = 1000, tol = 1e-7) {
    # If the signs of the function at the evaluated points, a and b, stop
the function and return message.
    if (!(f(a) < 0) && (f(b) > 0)) {
      stop('signs of f(a) and f(b) differ')
    }
    else if ((f(a) > 0) && (f(b) < 0)) {
      stop('signs of f(a) and f(b) differ')
    }

    for (i in 1:n) {
      c <- (a + b) / 2 # Calculate midpoint

      # If the function equals 0 at the midpoint or the midpoint is below
the desired tolerance, stop the
      # function and return the root.
      if ((f(c) == 0) || ((b - a) / 2) < tol) {
        return(c)
      }

      # If another iteration is required,
      # check the signs of the function at the points c and a and reassign
      # a or b accordingly as the midpoint to be used in the next
iteration.
      ifelse(sign(f(c)) == sign(f(a)),
             a <- c,
             b <- c)
    }
    # If the max number of iterations is reached and no root has been
found,
    # return message and end function.
    print('Too many iterations')
  }
  f         <- function(x) {param[1]*pdens(x,param[2],param[3]) +
     (1-param[1])*pdens(x,param[4],param[5]) - q}
  fr <- function(param2) {  ## Censored Weibull Type II
    beta <- param2[1]
    psi <- param2[2]
    (m*log(beta) + m*log(psi) + (beta-1)*sum(log(x)) - psi*sum(x^beta) -
(n-m)*psi*C^beta)*(-1)
  }
  N <- N
  q <- q
```

```
  q.estimate <- matrix(nrow=N, ncol = 1)
  for (j in (1:N)) {

    x <- tryCatch(
      {

        if(mix.ind == FALSE && gum.ind==FALSE){
          data  <- rdens(300, param[1], param[2])
        }
        if(mix.ind == TRUE && gum.ind==FALSE){

          ind  <- runif(300, 0, 1)
          data  <-  (ind <= param[1])*rdens(300, param[2], param[3]) +
            (ind  > param[1])*rdens(300, param[4], param[5])
        }
        if(mix.ind == FALSE && gum.ind==TRUE){
          data  <- rdens(300, param[1], param[2],max = FALSE)
        }


        C      <- quantile(data, 0.1, type = 3)
        x.star_ordered <- data[order(data)]
        x      <- x.star_ordered[(x.star_ordered<=C)]
        m      <- length(x)
        n      <- length(data)
        par    <- optim(c(0.5,0.5), fr)
        test   <- qweibull(q, par$par[1], (1/par$par[2])^(1/par$par[1]))
      },
      error = function(e){
        test <- NA
      }
    )
    q.estimate[j] <- x

  }
  if (mix.ind == FALSE && gum.ind == FALSE){
    MSE <- ((1/N)*sum((q.estimate-qdens(0.05, param[1], param[2]))**2))
  }
  if (mix.ind == FALSE && gum.ind == TRUE){
    MSE <-
((1/length(q.estimate[!is.na(q.estimate)]))*sum((q.estimate[!is.na(q.estima
te)]-qgumbel(0.05, param[1], param[2], max = FALSE))**2))
  }
  if (mix.ind == TRUE && gum.ind == FALSE) {
    MSE <- (sum((q.estimate[!is.na(q.estimate)]-
bisection(f,0,10))**2)/length(q.estimate[!is.na(q.estimate)]))
  }
  #q.estimate <- NULL

  sd_2  <- 100*apply(na.omit(q.estimate),2,sd)
  bias  <- 100*sqrt(abs(MSE - (sd_2/100)^2))
  return(list(sqrt(MSE),sd_2,bias))
}
OMLE <- function(N,q,rdens,qdens,pdens,param,mix.ind,gum.ind) {
  bisection <- function(f, a, b, n = 1000, tol = 1e-7) {
    # If the signs of the function at the evaluated points, a and b, stop
the function and return message.
    if (!(f(a) < 0) && (f(b) > 0)) {
      stop('signs of f(a) and f(b) differ')
    }
    else if ((f(a) > 0) && (f(b) < 0)) {
```

```
      stop('signs of f(a) and f(b) differ')
    }

    for (i in 1:n) {
      c <- (a + b) / 2 # Calculate midpoint

      # If the function equals 0 at the midpoint or the midpoint is below
the desired tolerance, stop the
      # function and return the root.
      if ((f(c) == 0) || ((b - a) / 2) < tol) {
        return(c)
      }

      # If another iteration is required,
      # check the signs of the function at the points c and a and reassign
      # a or b accordingly as the midpoint to be used in the next
iteration.
      ifelse(sign(f(c)) == sign(f(a)),
             a <- c,
             b <- c)
    }
    # If the max number of iterations is reached and no root has been
found,
    # return message and end function.
    print('Too many iterations')
  }
  f         <- function(x) {param[1]*pdens(x,param[2],param[3]) +
     (1-param[1])*pdens(x,param[4],param[5]) - q}

  N <- N
  q <- q
  q.estimate <- matrix(nrow=N, ncol = 1)
  for (j in (1:N)) {

    x <- tryCatch(
      {

        if(mix.ind == FALSE && gum.ind==FALSE){
          data  <- rdens(300, param[1], param[2])
        }
        if(mix.ind == TRUE && gum.ind==FALSE){

          ind  <- runif(300, 0, 1)
          data  <-  (ind <= param[1])*rdens(300, param[2], param[3]) +
            (ind  > param[1])*rdens(300, param[4], param[5])
        }
        if(mix.ind == FALSE && gum.ind==TRUE){
          data  <- rdens(300, param[1], param[2],max = FALSE)
        }


        fw            = fitdist(data, "weibull")
        q.estimate[j] = quantile(fw, probs = q)$quantiles[[1]]
      },
      error = function(e){
        test <- NA
      }
    )
    q.estimate[j] <- x

  }
```

```
  if (mix.ind == FALSE && gum.ind == FALSE){
    MSE <- ((1/N)*sum((q.estimate-qdens(0.05, param[1], param[2]))**2))
  }
  if (mix.ind == FALSE && gum.ind == TRUE){
    MSE <-
((1/length(q.estimate[!is.na(q.estimate)]))*sum((q.estimate[!is.na(q.estima
te)]-qgumbel(0.05, param[1], param[2], max = FALSE))**2))
  }
  if (mix.ind == TRUE && gum.ind == FALSE) {
    MSE <- (sum((q.estimate[!is.na(q.estimate)]-
bisection(f,0,10))**2)/length(q.estimate[!is.na(q.estimate)]))
  }
  #q.estimate <- NULL
  sd_2  <- 100*apply(na.omit(q.estimate),2,sd)
  bias  <- 100*sqrt(abs(MSE - (sd_2/100)^2))
  return(list(sqrt(MSE),sd_2,bias))
}
KDE  <- function(N,q,rdens,qdens,pdens,param,mix.ind,gum.ind, MOR.ind) {
  bisection  <- function(f, a, b, n = 1000, tol = 1e-7) {
    # If the signs of the function at the evaluated points, a and b, stop
the function and return message.
    if (!(f(a) < 0) && (f(b) > 0)) {
      stop('signs of f(a) and f(b) differ')
    }
    else if ((f(a) > 0) && (f(b) < 0)) {
      stop('signs of f(a) and f(b) differ')
    }

    for (i in 1:n) {
      c <- (a + b) / 2 # Calculate midpoint

      # If the function equals 0 at the midpoint or the midpoint is below
the desired tolerance, stop the
      # function and return the root.
      if ((f(c) == 0) || ((b - a) / 2) < tol) {
        return(c)
      }

      # If another iteration is required,
      # check the signs of the function at the points c and a and reassign
      # a or b accordingly as the midpoint to be used in the next
iteration.
      ifelse(sign(f(c)) == sign(f(a)),
             a <- c,
             b <- c)
    }
    # If the max number of iterations is reached and no root has been
found,
    # return message and end function.
    print('Too many iterations')
  }
  f          <- function(x) {param[1]*pdens(x,param[2],param[3]) +
     (1-param[1])*pdens(x,param[4],param[5]) - q}
  quant func <- function(data,q){
    dens = density(data)
    x    = dens$x
    F    = cumsum(dens$y)/sum(dens$y)
    out  = approx(F,x,q)$y
    return(out)
  }
  if(MOR.ind>1)  { MOR1_bw <- bw.SJ(MOR2, nb=15,
```

```
                                    method ="ste")}
  if(MOR.ind==1) { MOR1_bw <- bw.SJ(MOR1, nb=15,
                                    method ="ste")}


  N <- N
  q <- q
  q.estimate <- matrix(nrow=N, ncol = 1)
  for (j in (1:N)) {

    x <- tryCatch(
      {

        if(mix.ind == FALSE && gum.ind==FALSE){
          data  <- rdens(300, param[1], param[2])
        }
        if(mix.ind == TRUE && gum.ind==FALSE){

          ind  <- runif(300, 0, 1)
          data  <-  (ind <= param[1])*rdens(300, param[2], param[3]) +
            (ind  > param[1])*rdens(300, param[4], param[5])
        }
        if(mix.ind == FALSE && gum.ind==TRUE){
          data  <- rdens(300, param[1], param[2],max = FALSE)
        }


        b_MOR1          = sample(data, size = length(data), replace = TRUE)
+ rnorm(length(data), 0, MOR1_bw)
        q.estimate[j]   = quant_func(data,q)
      },
      error = function(e){
        test <- NA
      }
    )
    q.estimate[j] <- x

  }
  if (mix.ind == FALSE && gum.ind == FALSE){
    MSE <- ((1/N)*sum((q.estimate-qdens(0.05, param[1], param[2]))**2))
  }
  if (mix.ind == FALSE && gum.ind == TRUE){
    MSE <-
((1/length(q.estimate[!is.na(q.estimate)]))*sum((q.estimate[!is.na(q.estima
te)]-qgumbel(0.05, param[1], param[2], max = FALSE))**2))
  }
  if (mix.ind == TRUE && gum.ind == FALSE) {
    MSE <- (sum((q.estimate[!is.na(q.estimate)]-
bisection(f,0,10))**2)/length(q.estimate[!is.na(q.estimate)]))
  }
  #q.estimate <- NULL
  sd_2  <- 100*apply(na.omit(q.estimate),2,sd)
  bias  <- 100*sqrt(abs(MSE - (sd_2/100)^2))
  return(list(sqrt(MSE),sd_2,bias))

}
EMP  <- function(N,q,rdens,qdens,pdens,param,mix.ind,gum.ind) {
  bisection <- function(f, a, b, n = 1000, tol = 1e-7) {
    # If the signs of the function at the evaluated points, a and b, stop
the function and return message.
    if (!(f(a) < 0) && (f(b) > 0)) {
```

```
      stop('signs of f(a) and f(b) differ')
    }
    else if ((f(a) > 0) && (f(b) < 0)) {
      stop('signs of f(a) and f(b) differ')
    }

    for (i in 1:n) {
      c <- (a + b) / 2 # Calculate midpoint

      # If the function equals 0 at the midpoint or the midpoint is below
the desired tolerance, stop the
      # function and return the root.
      if ((f(c) == 0) || ((b - a) / 2) < tol) {
        return(c)
      }

      # If another iteration is required,
      # check the signs of the function at the points c and a and reassign
      # a or b accordingly as the midpoint to be used in the next
iteration.
      ifelse(sign(f(c)) == sign(f(a)),
             a <- c,
             b <- c)
    }
    # If the max number of iterations is reached and no root has been
found,
    # return message and end function.
    print('Too many iterations')
  }
  f         <- function(x) {param[1]*pdens(x,param[2],param[3]) +
     (1-param[1])*pdens(x,param[4],param[5]) - q}

  N <- N
  q <- q
  q.estimate <- matrix(nrow=N, ncol = 1)
  for (j in (1:N)) {

    x <- tryCatch(
      {

        if(mix.ind == FALSE && gum.ind==FALSE){
          data  <- rdens(300, param[1], param[2])
        }
        if(mix.ind == TRUE && gum.ind==FALSE){

          ind  <- runif(300, 0, 1)
          data  <-  (ind <= param[1])*rdens(300, param[2], param[3]) +
            (ind  > param[1])*rdens(300, param[4], param[5])
        }
        if(mix.ind == FALSE && gum.ind==TRUE){
          data  <- rdens(300, param[1], param[2],max = FALSE)
        }

        q.estimate[j] = quantile(data, probs = q, type = 9)
      },
      error = function(e){
        test <- NA
      }
    )
    q.estimate[j] <- x
```

```
  }
  if (mix.ind == FALSE && gum.ind == FALSE){
    MSE <- ((1/N)*sum((q.estimate-qdens(0.05, param[1], param[2]))**2))
  }
  if (mix.ind == FALSE && gum.ind == TRUE){
    MSE <-
((1/length(q.estimate[!is.na(q.estimate)]))*sum((q.estimate[!is.na(q.estima
te)]-qgumbel(0.05, param[1], param[2], max = FALSE))**2))
  }
  if (mix.ind == TRUE && gum.ind == FALSE) {
    MSE <- (sum((q.estimate[!is.na(q.estimate)]-
bisection(f,0,10))**2)/length(q.estimate[!is.na(q.estimate)]))
  }
  #q.estimate <- NULL
  sd_2  <- 100*apply(na.omit(q.estimate),2,sd)
  bias  <- 100*sqrt(abs(MSE - (sd_2/100)^2))
  return(list(sqrt(MSE),sd_2,bias))
}

q=0.05
N=10000

plan(multisession)

tempjob1 %<-% CMLE(N,q,rweibull,qweibull,NULL,c(6.822, 7.173),FALSE,FALSE)
tempjob2 %<-% CMLE(N,q,rgamma,qgamma,NULL,c(12.93, 1/0.601),FALSE,FALSE)
tempjob3 %<-% CMLE(N,q,rlnorm,qlnorm,NULL,c(2.072, 0.336),FALSE,FALSE)
tempjob4 %<-% CMLE(N,q,rgumbel,qgumbel,NULL,c(6.319, 0.601),FALSE,TRUE)
tempjob5 %<-%
CMLE(N,q,rnorm,NULL,pnorm,c(0.5629,5.953,0.970,7.676,1.215),TRUE,FALSE)
tempjob6 %<-%
CMLE(N,q,rlnorm,NULL,plnorm,c(0.9758,1.897,0.189,1.245,0.102),TRUE,FALSE)
tempjob7 %<-%
CMLE(N,q,rweibull,NULL,pweibull,c(0.7448,5.494,7.599,15.81,5.983),TRUE,FALS
E)

s <- Sys.time()
temp.list <- lapply(ls(pattern = "temp"), get)
results1  <-  as.data.frame(matrix(unlist(temp.list),
nrow=length(unlist(temp.list[1]))))
#names(results) <- c("wei", "gam", "lnorm")
e <- Sys.time()
e-s
rm(temp.list)

#MOR2
plan(multisession)

tempjob1 %<-% CMLE(N,q,rweibull,qweibull,NULL,c(7.378, 6.738),FALSE,FALSE)
tempjob2 %<-% CMLE(N,q,rgamma,qgamma,NULL,c(16.16, 1/0.4407),FALSE,FALSE)
tempjob3 %<-% CMLE(N,q,rlnorm,qlnorm,NULL,c(1.976, 0.2916),FALSE,FALSE)
tempjob4 %<-% CMLE(N,q,rgumbel,qgumbel,NULL,c(6.319, 0.601),FALSE,TRUE)
tempjob5 %<-%
CMLE(N,q,rnorm,NULL,pnorm,c(0.5406,5.924,1.042,7.859,1.095),TRUE,FALSE)
tempjob6 %<-%
CMLE(N,q,rlnorm,NULL,plnorm,c(0.6649,1.976,0.167,1.736,0.226),TRUE,FALSE)
tempjob7 %<-%
CMLE(N,q,rweibull,NULL,pweibull,c(0.7932,5.427,7.642,12.01,6.186),TRUE,FALS
E)

s <- Sys.time()
```

```
temp.list <- lapply(ls(pattern = "temp"), get)
results2 <-  as.data.frame(matrix(unlist(temp.list),
nrow=length(unlist(temp.list[1]))))
#names(results) <- c("wei", "gam", "lnorm")
e <- Sys.time()
e-s

names(results2) = c("Weibull", "Gamma", "Log-normal", "Minimum Gumbel",
"Normal Mixture", "Log-normal Mixture", "Weibull Mixture")
boxplot(results2, las =2, ylab ="Threshold")
mtext("Model imitating MOR2", side=1, line=8)

## updated CMLE

setwd("C:\\Users\\Jarod\\Google Drive\\Jarod M- 2018 and 2019\\Data
set\\AC\\Data")
# Read CSV into R
MyData   <- read.csv(file="MOR_Data.csv", header=TRUE, sep=",")
MOR1     <- MyData[MyData$Sample == 'MOR1',2]
MOR2     <- MyData[MyData$Sample == 'MOR2',2]

library(data.table)
library(future)
library(ordinal)
library(NLRoot)
library(fitdistrplus)

q=0.05
N=10000

##CMLE#######
#MOR1####

plan(multisession)

tempjob1 %<-% CMLE(N,q,rweibull,qweibull,NULL,c(6.822, 7.173),FALSE,FALSE)
tempjob2 %<-% CMLE(N,q,rgamma,qgamma,NULL,c(12.93, 1/0.601),FALSE,FALSE)
tempjob3 %<-% CMLE(N,q,rlnorm,qlnorm,NULL,c(2.072, 0.336),FALSE,FALSE)
tempjob4 %<-% CMLE(N,q,rgumbel,qgumbel,NULL,c(6.319, 0.601),FALSE,TRUE)
tempjob5 %<-%
CMLE(N,q,rnorm,NULL,pnorm,c(0.5629,5.953,0.970,7.676,1.215),TRUE,FALSE)
tempjob6 %<-%
CMLE(N,q,rlnorm,NULL,plnorm,c(0.9758,1.897,0.189,1.245,0.102),TRUE,FALSE)
tempjob7 %<-%
CMLE(N,q,rweibull,NULL,pweibull,c(0.7448,5.494,7.599,15.81,5.983),TRUE,FALS
E)

s <- Sys.time()
temp.list <- lapply(ls(pattern = "temp"), get)
results1  <-  as.data.frame(matrix(unlist(temp.list),
nrow=length(unlist(temp.list[1]))))
#names(results) <- c("wei", "gam", "lnorm")
e <- Sys.time()
e-s
rm(temp.list)

#MOR2####
plan(multisession)

tempjob1 %<-% CMLE(N,q,rweibull,qweibull,NULL,c(7.378, 6.738),FALSE,FALSE)
tempjob2 %<-% CMLE(N,q,rgamma,qgamma,NULL,c(16.16, 1/0.4407),FALSE,FALSE)
```

```
tempjob3 %<-% CMLE(N,q,rlnorm,qlnorm,NULL,c(1.976, 0.2916),FALSE,FALSE)
tempjob4 %<-% CMLE(N,q,rgumbel,qgumbel,NULL,c(6.319, 0.601),FALSE,TRUE)
tempjob5 %<-%
CMLE(N,q,rnorm,NULL,pnorm,c(0.5406,5.924,1.042,7.859,1.095),TRUE,FALSE)
tempjob6 %<-%
CMLE(N,q,rlnorm,NULL,plnorm,c(0.6649,1.976,0.167,1.736,0.226),TRUE,FALSE)
tempjob7 %<-%
CMLE(N,q,rweibull,NULL,pweibull,c(0.7932,5.427,7.642,12.01,6.186),TRUE,FALS
E)

s <- Sys.time()
temp.list <- lapply(ls(pattern = "temp"), get)
results2 <-  as.data.frame(matrix(unlist(temp.list),
nrow=length(unlist(temp.list[1]))))
#names(results) <- c("wei", "gam", "lnorm")
e <- Sys.time()
e-s

##OMLE#######
#MOR1####

plan(multisession)

tempjob1 %<-% OMLE(N,q,rweibull,qweibull,NULL,c(6.822, 7.173),FALSE,FALSE)
tempjob2 %<-% OMLE(N,q,rgamma,qgamma,NULL,c(12.93, 1/0.601),FALSE,FALSE)
tempjob3 %<-% OMLE(N,q,rlnorm,qlnorm,NULL,c(2.072, 0.336),FALSE,FALSE)
tempjob4 %<-% OMLE(N,q,rgumbel,qgumbel,NULL,c(6.319, 0.601),FALSE,TRUE)
tempjob5 %<-%
OMLE(N,q,rnorm,NULL,pnorm,c(0.5629,5.953,0.970,7.676,1.215),TRUE,FALSE)
tempjob6 %<-%
OMLE(N,q,rlnorm,NULL,plnorm,c(0.9758,1.897,0.189,1.245,0.102),TRUE,FALSE)
tempjob7 %<-%
OMLE(N,q,rweibull,NULL,pweibull,c(0.7448,5.494,7.599,15.81,5.983),TRUE,FALS
E)

s <- Sys.time()
temp.list <- lapply(ls(pattern = "temp"), get)
results1  <-  as.data.frame(matrix(unlist(temp.list),
nrow=length(unlist(temp.list[1]))))
#names(results) <- c("wei", "gam", "lnorm")
e <- Sys.time()
e-s
rm(temp.list)

#MOR2####
plan(multisession)

tempjob1 %<-% OMLE(N,q,rweibull,qweibull,NULL,c(7.378, 6.738),FALSE,FALSE)
tempjob2 %<-% OMLE(N,q,rgamma,qgamma,NULL,c(16.16, 1/0.4407),FALSE,FALSE)
tempjob3 %<-% OMLE(N,q,rlnorm,qlnorm,NULL,c(1.976, 0.2916),FALSE,FALSE)
tempjob4 %<-% OMLE(N,q,rgumbel,qgumbel,NULL,c(6.319, 0.601),FALSE,TRUE)
tempjob5 %<-%
OMLE(N,q,rnorm,NULL,pnorm,c(0.5406,5.924,1.042,7.859,1.095),TRUE,FALSE)
tempjob6 %<-%
OMLE(N,q,rlnorm,NULL,plnorm,c(0.6649,1.976,0.167,1.736,0.226),TRUE,FALSE)
tempjob7 %<-%
OMLE(N,q,rweibull,NULL,pweibull,c(0.7932,5.427,7.642,12.01,6.186),TRUE,FALS
E)

s <- Sys.time()
temp.list <- lapply(ls(pattern = "temp"), get)
```

```
results2 <-  as.data.frame(matrix(unlist(temp.list),
nrow=length(unlist(temp.list[1]))))
#names(results) <- c("wei", "gam", "lnorm")
e <- Sys.time()
e-s

##KDE#######
#MOR1####

plan(multisession)

tempjob1 %<-% KDE(N,q,rweibull,qweibull,NULL,c(6.822, 7.173),FALSE,FALSE,1)
tempjob2 %<-% KDE(N,q,rgamma,qgamma,NULL,c(12.93, 1/0.601),FALSE,FALSE,1)
tempjob3 %<-% KDE(N,q,rlnorm,qlnorm,NULL,c(2.072, 0.336),FALSE,FALSE,1)
tempjob4 %<-% KDE(N,q,rgumbel,qgumbel,NULL,c(6.319, 0.601),FALSE,TRUE,1)
tempjob5 %<-%
KDE(N,q,rnorm,NULL,pnorm,c(0.5629,5.953,0.970,7.676,1.215),TRUE,FALSE,1)
tempjob6 %<-%
KDE(N,q,rlnorm,NULL,plnorm,c(0.9758,1.897,0.189,1.245,0.102),TRUE,FALSE,1)
tempjob7 %<-%
KDE(N,q,rweibull,NULL,pweibull,c(0.7448,5.494,7.599,15.81,5.983),TRUE,FALSE
,1)

s <- Sys.time()
temp.list <- lapply(ls(pattern = "temp"), get)
results1  <-  as.data.frame(matrix(unlist(temp.list),
nrow=length(unlist(temp.list[1]))))
#names(results) <- c("wei", "gam", "lnorm")
e <- Sys.time()
e-s
rm(temp.list)

#MOR2####
plan(multisession)

tempjob1 %<-% KDE(N,q,rweibull,qweibull,NULL,c(7.378, 6.738),FALSE,FALSE,2)
tempjob2 %<-% KDE(N,q,rgamma,qgamma,NULL,c(16.16, 1/0.4407),FALSE,FALSE,2)
tempjob3 %<-% KDE(N,q,rlnorm,qlnorm,NULL,c(1.976, 0.2916),FALSE,FALSE,2)
tempjob4 %<-% KDE(N,q,rgumbel,qgumbel,NULL,c(6.319, 0.601),FALSE,TRUE,2)
tempjob5 %<-%
KDE(N,q,rnorm,NULL,pnorm,c(0.5406,5.924,1.042,7.859,1.095),TRUE,FALSE,2)
tempjob6 %<-%
KDE(N,q,rlnorm,NULL,plnorm,c(0.6649,1.976,0.167,1.736,0.226),TRUE,FALSE,2)
tempjob7 %<-%
KDE(N,q,rweibull,NULL,pweibull,c(0.7932,5.427,7.642,12.01,6.186),TRUE,FALSE
,2)

s <- Sys.time()
temp.list <- lapply(ls(pattern = "temp"), get)
results2 <-  as.data.frame(matrix(unlist(temp.list),
nrow=length(unlist(temp.list[1]))))
#names(results) <- c("wei", "gam", "lnorm")
e <- Sys.time()
e-s

##EMP#######
#MOR1####

plan(multisession)

tempjob1 %<-% EMP(N,q,rweibull,qweibull,NULL,c(6.822, 7.173),FALSE,FALSE)
```

```
tempjob2 %<-% EMP(N,q,rgamma,qgamma,NULL,c(12.93, 1/0.601),FALSE,FALSE)
tempjob3 %<-% EMP(N,q,rlnorm,qlnorm,NULL,c(2.072, 0.336),FALSE,FALSE)
tempjob4 %<-% EMP(N,q,rgumbel,qgumbel,NULL,c(6.319, 0.601),FALSE,TRUE)
tempjob5 %<-%
EMP(N,q,rnorm,NULL,pnorm,c(0.5629,5.953,0.970,7.676,1.215),TRUE,FALSE)
tempjob6 %<-%
EMP(N,q,rlnorm,NULL,plnorm,c(0.9758,1.897,0.189,1.245,0.102),TRUE,FALSE)
tempjob7 %<-%
EMP(N,q,rweibull,NULL,pweibull,c(0.7448,5.494,7.599,15.81,5.983),TRUE,FALSE
)

s <- Sys.time()
temp.list <- lapply(ls(pattern = "temp"), get)
results1  <-  as.data.frame(matrix(unlist(temp.list),
nrow=length(unlist(temp.list[1]))))
#names(results) <- c("wei", "gam", "lnorm")
e <- Sys.time()
e-s
rm(temp.list)

#MOR2####
plan(multisession)

tempjob1 %<-% EMP(N,q,rweibull,qweibull,NULL,c(7.378, 6.738),FALSE,FALSE)
tempjob2 %<-% EMP(N,q,rgamma,qgamma,NULL,c(16.16, 1/0.4407),FALSE,FALSE)
tempjob3 %<-% EMP(N,q,rlnorm,qlnorm,NULL,c(1.976, 0.2916),FALSE,FALSE)
tempjob4 %<-% EMP(N,q,rgumbel,qgumbel,NULL,c(6.319, 0.601),FALSE,TRUE)
tempjob5 %<-%
EMP(N,q,rnorm,NULL,pnorm,c(0.5406,5.924,1.042,7.859,1.095),TRUE,FALSE)
tempjob6 %<-%
EMP(N,q,rlnorm,NULL,plnorm,c(0.6649,1.976,0.167,1.736,0.226),TRUE,FALSE)
tempjob7 %<-%
EMP(N,q,rweibull,NULL,pweibull,c(0.7932,5.427,7.642,12.01,6.186),TRUE,FALSE
)

s <- Sys.time()
temp.list <- lapply(ls(pattern = "temp"), get)
results2 <-  as.data.frame(matrix(unlist(temp.list),
nrow=length(unlist(temp.list[1]))))
#names(results) <- c("wei", "gam", "lnorm")
e <- Sys.time()
e-s

#Censored and uncensored mixture
#MOR1
library(data.table)
library(future)
library(ordinal)
library(NLRoot)

#Simulation study for mixture models
## MOR1
mix <- function(N,name,rdens,qdens,pdens,par,q){
  bisection <- function(f, a, b, n = 1000, tol = 1e-7) {
    # If the signs of the function at the evaluated points, a and b, stop
the function and return message.
    if (!(f(a) < 0) && (f(b) > 0)) {
      stop('signs of f(a) and f(b) differ')
    } else if ((f(a) > 0) && (f(b) < 0)) {
      stop('signs of f(a) and f(b) differ')
    }
```

```
    for (i in 1:n) {
      c <- (a + b) / 2 # Calculate midpoint

      # If the function equals 0 at the midpoint or the midpoint is below
the desired tolerance, stop the
      # function and return the root.
      if ((f(c) == 0) || ((b - a) / 2) < tol) {
        return(c)
      }

      # If another iteration is required,
      # check the signs of the function at the points c and a and reassign
      # a or b accordingly as the midpoint to be used in the next
iteration.
      ifelse(sign(f(c)) == sign(f(a)),
             a <- c,
             b <- c)
    }
    # If the max number of iterations is reached and no root has been
found,
    # return message and end function.
    print('Too many iterations')
  }
  f         <- function(x)
{param_w[[1]][1]*pweibull(x,param_w[[2]][1],param_w[[3]][1]) +
      (1-param_w[[1]][1])*pweibull(x,param_w[[2]][2],param_w[[3]][2]) - q}
  f1        <- function(x) {par[1]*pdens(x,par[2],par[3]) +
      (1-par[1])*pdens(x,par[4],par[5]) - q}
  em_w      <- function(DS) {
    count  <- 0
    #initialise mixing probabilities to be equal
    probs  <- 0.5
    #initialise parameters
    alpha <- c(6.823130, 11 + rnorm(1,0,1))
    eta    <- c(7.173, 7.173 + rnorm(1,0,1))
    C      <- quantile(DS, probs = 0.9, type = 3)
    DS_ordered = DS[order(DS)]
    x          = DS_ordered[(DS_ordered<=C)]
    m          = length(x)
    n          = length(DS)
    diff       = 1e+11

    while(count < 1000)
    {
      #likelihood function
      weibull <- cbind(dweibull(x, alpha[1], eta[1], log = FALSE),
dweibull(x, alpha[2], eta[2], log = FALSE))

      #responsibilities

      gam <- rbind(cbind((probs*weibull[,1])/(probs*weibull[,1] + (1-
probs)*weibull[,2]),
                         ((1-probs)*weibull[,2])/(probs*weibull[,1] + (1-
probs)*weibull[,2])),
                   cbind((probs*(1-pweibull(C,alpha[1], eta[1],lower.tail =
TRUE, log.p = FALSE))/(probs*(1-pweibull(C,alpha[1], eta[1],lower.tail =
TRUE, log.p = FALSE))

+(1- probs)*(1-pweibull(C,alpha[2], eta[2],lower.tail = TRUE, log.p =
FALSE)))),
```

```
                          ((1-probs)*(1-pweibull(C,alpha[2],
eta[2],lower.tail = TRUE, log.p = FALSE))/(probs*(1-pweibull(C,alpha[1],
eta[1]))

+ (1- probs)*(1-pweibull(C,alpha[2], eta[2],lower.tail = TRUE, log.p =
FALSE))))))

      #estimate parameters
      alpha_new <- cbind((sum(head(gam[,1],m)*(x^alpha[1])*log(x))+(n-
m)*tail(gam[,1],1)*C^alpha[1]*log(C))/(sum(head(gam[,1],m)*x^alpha[1])+(n-
m)*tail(gam[,1],1)*C^alpha[1]) -
sum(head(gam[,1],m)*log(x))/sum(head(gam[,1],m)),
                         (sum(head(gam[,2],m)*(x^alpha[2])*log(x))+(n-
m)*tail(gam[,2],1)*C^alpha[2]*log(C))/(sum(head(gam[,2],m)*x^alpha[2])+(n-
m)*tail(gam[,2],1)*C^alpha[2]) -
sum(head(gam[,2],m)*log(x))/sum(head(gam[,2],m)))

      eta_new <- cbind(((sum(head(gam[,1],m)*x^(1/alpha_new[1]))+(n-
m)*tail(gam[,1],1)*C^(1/alpha_new[1]))/sum(head(gam[,1],m)))^(alpha_new[1])
,
                       ((sum(head(gam[,2],m)*x^(1/alpha_new[2]))+(n-
m)*tail(gam[,2],1)*C^(1/alpha_new[2]))/sum(head(gam[,2],m)))^(alpha_new[2])
)

      probs_new <- (sum(head(gam[,1],m)) + tail(gam[,1],1)*(n-
m))/length(DS)

      #update parameters
      a0    <- sum(abs((alpha - 1/alpha_new)) + abs((eta - eta_new)) +
abs((probs - probs_new)))
      probs <- probs_new
      alpha <- 1/alpha_new
      eta   <- eta_new

      centres     <- kmeans(DS,2)
      alpha[1]    <- ifelse(alpha[1]>=30, centres$centers[1], alpha[1])
      alpha[2]    <- ifelse(alpha[2]>=30, centres$centers[2], alpha[2])
      count    <- count+1
      a2  <- ifelse(count>=500,0.0001,0)
      a3  <- ifelse(is.na(a0), 0.0001, 0)
      d   <- a0+a2+a3
      diff <- d
    }

    return(list(probs,alpha, eta))
  }
  q.estimate <- matrix(nrow=N, ncol = 1)
  for (j in (1:N)) {
    x <- tryCatch(
      {

        if(name!="mix"){
          if(name=="gumbel"){dat <- rdens(300,par[1], par[2],max = FALSE)}
          if(name!="gumbel"){dat <- rdens(300,par[1], par[2])}
        }
        if(name=="mix"){
          ind           <- runif(300, 0, 1)
          dat           <- (ind <=par[1])*1*rdens(300, par[2], par[3]) +
            (ind  > par[1])*1*rdens(300, par[4], par[5])

        }
```

```
        param_w   <- em_w(dat)
        quant     <- bisection(f, 0, 10)
        test      <-quant
      },
      error = function(e){
        test <- NA
      }
    )
    q.estimate[j] <- x
  }

  if (name!="mix" && name!="gumbel"){
    MSE <- ((1/length(q.estimate[!is.na(q.estimate)]))*sum((q.estimate-
qdens(0.05, par[1], par[2]))**2))
  }
  if (name=="gumbel"){
    MSE <- ((1/length(q.estimate))*sum((na.omit(q.estimate)-qgumbel(0.05,
par[1], par[2], max = FALSE))**2))
  }
  if (name=="mix") {
    MSE <- (sum((q.estimate[!is.na(q.estimate)]-
bisection(f1,0,10))**2)/length(q.estimate[!is.na(q.estimate)]))
  }
  #q.estimate <- NULL
  sd_2  <- 100*apply(na.omit(q.estimate),2,sd)
  bias  <- 100*sqrt(abs(MSE - (sd_2/100)^2))
  return(list(sqrt(MSE),sd_2,bias))

}

q=0.05
N=10000

plan(multisession)
tempjob1 %<-% mix(N,"w",rweibull,qweibull,NULL,c(7.378,6.739),q)
tempjob2 %<-% mix(N,"g",rgamma,qgamma,NULL,c(16.16, 1/0.4407),q)
tempjob3 %<-% mix(N,"ln",rlnorm,qlnorm,NULL,c(1.976, 0.2916),q)
tempjob4 %<-% mix(N,"gumbel",rgumbel,qgumbel,NULL,c(6.319, 0.601),q)
tempjob5 %<-%
mix(N,"mix",rnorm,NULL,pnorm,c(0.5406,5.924,1.042,7.859,1.095),q)
tempjob6 %<-%
mix(N,"mix",rlnorm,NULL,plnorm,c(0.6649,1.976,0.167,1.736,0.226),q)
tempjob7 %<-%
mix(N,"mix",rweibull,NULL,pweibull,c(0.7932,5.427,7.642,12.01,6.186),q)

s <- Sys.time()
temp.list <- lapply(ls(pattern = "temp"), get)
M2 <-  as.data.frame(matrix(unlist(temp.list),
nrow=length(unlist(temp.list[1]))))
#names(results) <- c("wei", "gam", "lnorm")
e <- Sys.time()
e-s
rm(temp.list)
## MOR1
plan(multisession)
tempjob1 %<-% mix(N,"w",rweibull,qweibull,NULLc(6.822, 7.173),q)
tempjob2 %<-% mix(N,"g",rgamma,qgamma,NULL,c(12.93, 1/0.601),q)
tempjob3 %<-% mix(N,"ln",rlnorm,qlnorm,NULL,c(2.072, 0.336),q)
tempjob4 %<-% mix(N,"gumbel",rgumbel,qgumbel,NULL,c(6.319, 0.601),q)
tempjob5 %<-%
mix(N,"mix",rnorm,NULL,pnorm,c(0.5629,5.953,0.970,7.676,1.215),q)
```

```
tempjob6 %<-%
mix(N,"mix",rlnorm,NULL,plnorm,c(0.9758,1.897,0.189,1.245,0.102),q)
tempjob7 %<-%
mix(N,"mix",rweibull,NULL,pweibull,c(0.7448,5.494,7.599,15.81,5.983),q)
s <- Sys.time()
temp.list <- lapply(ls(pattern = "temp"), get)
M1 <-  as.data.frame(matrix(unlist(temp.list),
nrow=length(unlist(temp.list[1]))))
#names(results) <- c("wei", "gam", "lnorm")
e <- Sys.time()
e-s
rm(temp.list)

#Bootstrap homogeneity test
#Reliability of p-value study

library(fitdistrplus)
library(foreach)
library(doParallel)
em_w <- function(DS) {
  count <- 0
  #initialise mixing probabilities to be equal
  probs <- 0.5
  #initialise parameters
  alpha <- c(6.823130, 6.823130 + rnorm(1,0,1))
  eta <- c(7.173, 7.173 + rnorm(1,0,1))

  diff = 1e+11

  while(diff > 0.01)
  {
    #likelihood function
    weibull <- cbind(dweibull(DS, alpha[1], eta[1], log = FALSE),
dweibull(DS, alpha[2], eta[2], log = FALSE))

    #responsibilities
    gam <- (probs*weibull[,1])/(probs*weibull[,1] + (1-probs)*weibull[,2])

    #estimate parameters
    alpha_new <- cbind(sum(gam*(DS^alpha[1])*log(DS))/sum(gam*DS^alpha[1])
- sum(gam*log(DS))/sum(gam),
                       sum((1-gam)*(DS^alpha[2])*log(DS))/sum((1-
gam)*DS^alpha[2]) - sum((1-gam)*log(DS))/sum((1-gam)))

    eta_new <-
cbind((sum(gam*DS^(1/alpha_new[1]))/sum(gam))^(alpha_new[1]),
                     (sum((1-gam)*DS^(1/alpha_new[2]))/sum((1-
gam)))^(alpha_new[2]))

    probs_new <- sum((1-gam))/length(DS)

    #update parameters
    diff  <- sum(abs((alpha - 1/alpha_new)) + abs((eta - eta_new)) +
abs((probs - probs_new)))
    probs <- probs_new
    alpha <- 1/alpha_new
    eta   <- eta_new

    diff  <- ifelse(alpha[1]>=40, 1, ifelse(alpha[2]>=40, 1, diff))
    count <- count+1
    diff  <- ifelse(count>=70,0.001,diff)
```

```
    diff  <- ifelse(is.na(diff), 0.001, diff)
  }

  return(list(probs,alpha, eta, gam, count))
}

out_fit <- function(n, shape, scale) {
  #Generate the data set
  dat <-rweibull(n, shape, scale)

  #Obtain MLE estimates for single and mixture Weibull

  ##Single
  fit_single    <- fitdist(dat, "weibull")
  loglik_single <- fit_single$loglik
  MLE_single    <- fit_single$estimate

  #Mixture
  fit_mix       <- em_w(dat)
  loglik_mix    <-
sum(fit_mix[[4]][1]*log(fit_mix[[1]]*dweibull(dat,fit_mix[[2]][1],
fit_mix[[3]][1]))
                       + (1-fit_mix[[4]][1])*log((1-
fit_mix[[1]])*dweibull(dat,fit_mix[[2]][2], fit_mix[[3]][2])))

  #Likelihood ratio 1
  lamda         <- loglik_single-loglik_mix
  L1            <- -2*lamda
  return(list(MLE_single, L1))
}
in_fit  <- function(n,B) {

  L1      <- out_fit(n, shape, scale) [[2]][1]
  shape_2 <- out_fit(n, shape, scale) [[1]][1]
  scale_2 <- out_fit(n, shape, scale) [[1]][2]
  ind_sum <- 0

  for(j in 1:B) {

    dat2              <- rweibull(n, shape_2, scale_2)

    ##Single
    fit_single_2      <- fitdist(dat2, "weibull")
    loglik_single_2   <- fit_single_2$loglik

    #Mixture
    fit_mix_2         <- em_w(dat2)
    loglik_mix_2      <-
sum(fit_mix_2[[4]][1]*log(fit_mix_2[[1]]*dweibull(dat2,fit_mix_2[[2]][1],
fit_mix_2[[3]][1]))
                              + (1-fit_mix_2[[4]][1])*log((1-
fit_mix_2[[1]])*dweibull(dat2,fit_mix_2[[2]][2], fit_mix_2[[3]][2])))
    #Likelihood ratio
    lamda_2           <- loglik_single_2-loglik_mix_2
    L2                <- -2*lamda_2
    test              <- (L2 > L1)
    test              <- ifelse(is.na(test),0,test)
    ind_sum           <- ind_sum + test

  }
```

```
  return(ind_sum/B)
}

cores=detectCores()
cl <- makeCluster(cores[1]-1)
registerDoParallel(cl)
on.exit(stopCluster(cl))

B     <- 500
N     <- 1000
n     <- 300
shape <- 7
scale <- 7

system.time({

  p_value  <- foreach(i = 1:N, .packages = 'fitdistrplus',.combine = rbind)
%dopar% {
    in_fit(n,B)
  }
  stopCluster(cl)
})


p_value <- p_value[order(p_value)]
x <-runif(1000,0,1)


##Plot empirical cdf of p values
##against cdf of uniform(0, 1)

p_value.cdf <- ecdf(p_value)
plot(p_value.cdf, xlab = 'p-value', ylab = '', main = '', lty=1, col =
'black')
lines(ecdf(x), lty=1, col = 'grey')
mtext(text = expression(hat(F)[n](x)), side = 2, line = 2.5)
legend(x = c(-0.11, 0.7), y = c(0.95, 0.95),-0.11, 0.95,
legend=c("Empirical CDF of p-values", "Uniform (0, 1)"),
       col=c("black", "grey"), lty=1:2, cex=1.2, box.lty=0,x.intersp =
0.3,y.intersp=0.35, bg="transparent")

#KS test
ks_test <- ks.test(x,p_value)
ks_test$statistic
ks_test$p.value

##############################
####    Chapter 4 code    ####
##############################

## BMLE RMSE of quantiles for all models
library(data.table)
library(future)
library(ordinal)
library(NLRoot)

BMLE <- function(N,B,q,rdens,qdens,pdens,param,mix.ind,gum.ind) {

  N <- N
  B <- B
```

```
  q <- q
  Candidate.C <- matrix(seq(0.05,1,0.05))
  bisection <- function(f, a, b, n = 1000, tol = 1e-7) {
    # If the signs of the function at the evaluated points, a and b, stop
the function and return message.
    if (!(f(a) < 0) && (f(b) > 0)) {
      stop('signs of f(a) and f(b) differ')
    } else if ((f(a) > 0) && (f(b) < 0)) {
      stop('signs of f(a) and f(b) differ')
    }

    for (i in 1:n) {
      c <- (a + b) / 2 # Calculate midpoint

      # If the function equals 0 at the midpoint or the midpoint is below
the desired tolerance, stop the
      # function and return the root.
      if ((f(c) == 0) || ((b - a) / 2) < tol) {
        return(c)
      }

      # If another iteration is required,
      # check the signs of the function at the points c and a and reassign
      # a or b accordingly as the midpoint to be used in the next
iteration.
      ifelse(sign(f(c)) == sign(f(a)),
             a <- c,
             b <- c)
    }
    # If the max number of iterations is reached and no root has been
found,
    # return message and end function.
    print('Too many iterations')
  }
  f         <- function(x) {param[1]*pdens(x,param[2],param[3]) +
     (1-param[1])*pdens(x,param[4],param[5]) - q}
  fr <- function(param2) {  ## Censored Weibull Type II
    beta <- param2[1]
    psi <- param2[2]
    (m*log(beta) + m*log(psi) + (beta-1)*sum(log(x)) - psi*sum(x^beta) -
(n-m)*psi*C^beta)*(-1)
  }
  small.boot <- function(p,B,data,q.IX,q) {
    q.boot.MSE <- matrix(nrow=B,ncol=1)
    C <- quantile(data, probs=p, type = 3)
    fr <- function(param2) {  ## Censored Weibull Type II
      beta <- param2[1]
      psi <- param2[2]
      (m*log(beta) + m*log(psi) + (beta-1)*sum(log(x)) - psi*sum(x^beta) -
(n-m)*psi*C^beta)*(-1)
    }
    for (i in 1:B) {
      x.star <- sample(data,replace=TRUE)
      x.star_ordered <- x.star[order(x.star)]
      x <- x.star_ordered[(x.star_ordered<=C)]
      m <- length(x)
      n <- length(x.star)
      par <- optim(c(0.5,0.5), fr)
      q.boot.MSE[i] <-  qweibull(q, par$par[1],
(1/par$par[2])^(1/par$par[1]))
    }
```

```
    MSE <- (1/B)*sum((q.boot.MSE-q.IX)^2)
    return(c(sqrt(MSE), p))
    q.boot.MSE <- NULL
  }
  q.estimate <- matrix(nrow=N, ncol = 1)
  for (j in (1:N)) {

    if(mix.ind == FALSE && gum.ind==FALSE){
      data  <- rdens(300, param[1], param[2])
    }
    if(mix.ind == TRUE && gum.ind==FALSE){

      ind  <- runif(300, 0, 1)
      data  <-  (ind <= param[1])*rdens(300, param[2], param[3]) +
        (ind  > param[1])*rdens(300, param[4], param[5])
    }
    if(mix.ind == FALSE && gum.ind==TRUE){
      data  <- rdens(300,param[1], param[2],max = FALSE)
    }

    #Bootstrapping component
    q.IX   <- quantile(data, probs=q, type = 9)
    df     <- data.frame(t(apply(Candidate.C,1,FUN = function(y)
small.boot(y,B,data,q.IX,q))))
    x.star <- sample(data, replace = TRUE)
    C      <- quantile(data, df[which.min(df[,1]),2], type = 3)
    rm(df)
    x.star_ordered <- x.star[order(x.star)]
    x      <- x.star_ordered[(x.star_ordered<=C)]
    m      <- length(x)
    n      <- length(x.star)
    par    <- optim(c(0.5,0.5), fr)
    q.estimate[j] <-  qweibull(q, par$par[1],
(1/par$par[2])^(1/par$par[1]))
  }
  if (mix.ind == FALSE && gum.ind == FALSE){
    MSE <- ((1/N)*sum((q.estimate-qdens(0.05, param[1], param[2]))**2))
  }
  if (mix.ind == FALSE && gum.ind == TRUE){
    MSE <- ((1/N)*sum((q.estimate-qgumbel(0.05, param[1], param[2], max =
FALSE))**2))
  }
  if (mix.ind == TRUE && gum.ind == FALSE) {
    MSE <- (sum((q.estimate[!is.na(q.estimate)]-
bisection(f,0,10))**2)/length(q.estimate[!is.na(q.estimate)]))
  }
  q.estimate <- NULL
  return(sqrt(MSE))
}

q=0.05
B=500
N=1000

plan(multisession)

tempjob1 %<-% BMLE(N,B,q,rweibull,qweibull,NULL,c(7.378,
6.739),FALSE,FALSE)
tempjob2 %<-% BMLE(N,B,q,rgamma,qgamma,NULL,c(16.168, 1/0.440),FALSE,FALSE)
tempjob3 %<-% BMLE(N,B,q,rlnorm,qlnorm,NULL,c(1.951, 0.286),FALSE,FALSE)
tempjob4 %<-% BMLE(N,B,q,rgumbel,qgumbel,NULL,c(6.319, 0.601),FALSE,TRUE)
```

```
tempjob5 %<-%
BMLE(N,B,q,rnorm,NULL,pnorm,c(0.7596791,6.283595,1.164798,8.432113,0.871932
8),TRUE,FALSE)
tempjob6 %<-%
BMLE(N,B,q,rlnorm,NULL,plnorm,c(0.6379,1.980,0.166,1.746,0.226),TRUE,FALSE)
tempjob7 %<-%
BMLE(N,B,q,rweibull,NULL,pweibull,c(0.7753,5.507,7.676,11.142,6.163),TRUE,F
ALSE)

temp.list <- lapply(ls(pattern = "temp"), get)
results  <-  as.data.frame(matrix(unlist(temp.list),
nrow=length(unlist(temp.list[1]))))
names(results) <- c("weibull", "gamma", "lnorm", "gumbel", "Mnorm",
"Mlnorm", "Mweibull")

s <- Sys.time()
BMLE(N,B,q,rweibull,qweibull,NULL,c(7.378, 6.739),FALSE,FALSE)
e <- Sys.time()

e-s

### simulation study to evaluate
### the consistency of the MSE estimate

library(ordinal)
library(data.table)
library(future)

MC.dist.diff.MSE <- function(N, B, n, q, param, rdens, qdens, pdens,
mix.ind,gum.ind){

  bisection <- function(f, a, b, n = 1000, tol = 1e-7) {
    # If the signs of the function at the evaluated points, a and b, stop
the function and return message.
    if (!(f(a) < 0) && (f(b) > 0)) {
      stop('signs of f(a) and f(b) differ')
    } else if ((f(a) > 0) && (f(b) < 0)) {
      stop('signs of f(a) and f(b) differ')
    }

    for (i in 1:n) {
      c <- (a + b) / 2 # Calculate midpoint

      # If the function equals 0 at the midpoint or the midpoint is below
the desired tolerance, stop the
      # function and return the root.
      if ((f(c) == 0) || ((b - a) / 2) < tol) {
        return(c)
      }

      # If another iteration is required,
      # check the signs of the function at the points c and a and reassign
      # a or b accordingly as the midpoint to be used in the next
iteration.
      ifelse(sign(f(c)) == sign(f(a)),
             a <- c,
             b <- c)
    }
    # If the max number of iterations is reached and no root has been
found,
    # return message and end function.
```

```
    print('Too many iterations')
  }
  f         <- function(x) {param[1]*pdens(x,param[2],param[3]) +
      (1-param[1])*pdens(x,param[4],param[5]) - q}
  fr <- function(param2) {  ## Censored Weibull Type II
    beta <- param2[1]
    psi <- param2[2]
    (m*log(beta) + m*log(psi) + (beta-1)*sum(log(x)) - psi*sum(x^beta) -
(n-m)*psi*C^beta)*(-1)
  }
  N=N
  #population
  q.pop.MSE = matrix(nrow=N,ncol=1)
  #bootstrap
  boot.MSE.j = matrix(nrow=N,ncol=1)
  for (j in 1:N) {

    #Simulate the data
    if(mix.ind == FALSE && gum.ind==FALSE){
      data  <- rdens(n, param[1], param[2])
    }
    if(mix.ind == TRUE && gum.ind==FALSE){

      ind  <- runif(n, 0, 1)
      data  <-  (ind <= param[1])*rdens(n, param[2], param[3]) +
        (ind  > param[1])*rdens(n, param[4], param[5])
    }
    if(mix.ind == FALSE && gum.ind==TRUE){
      data  <- rdens(n,param[1], param[2],max = FALSE)
    }
    ### population MSE estimate
    C <- quantile(data, probs = 0.1, type = 3)
    data_ordered <- data[order(data)]
    x <- data_ordered[(data_ordered<=C)]
    m <- length(x)
    n <- length(data)

    par <- optim(c(0.5,0.5), fr)
    q.pop.MSE[j] <-  qweibull(q, par$par[1], (1/par$par[2])^(1/par$par[1]))

    ### bootstrap MSE estimate
    B = B
    q.boot.MSE <- matrix(nrow=B,ncol=1)
    q.IX <- quantile(data, probs = q, type = 9)

    for (i in 1:B) {
      x.star <- sample(data, replace = TRUE)
      C <- quantile(x.star, probs = 0.1, type = 3)
      x.star_ordered <- x.star[order(x.star)]
      x <- x.star_ordered[(x.star_ordered<=C)]
      m <- length(x)
      n <- length(x.star)
      par <- optim(c(0.5,0.5), fr)
      q.boot.MSE[i] <-  qweibull(q, par$par[1],
(1/par$par[2])^(1/par$par[1]))
    }
    boot.MSE.j[j] = (1/B)*sum((q.boot.MSE-q.IX)^2)
  }

  if (mix.ind == FALSE && gum.ind == FALSE){
    MSE.pop <- (1/N)*sum(((q.pop.MSE-qdens(q, param[1], param[2]))^2))
```

```
  }
  if (mix.ind == FALSE && gum.ind == TRUE){
    MSE.pop <- ((1/N)*sum((q.pop.MSE-qgumbel(q, param[1], param[2], max =
FALSE))**2))
  }
  if (mix.ind == TRUE && gum.ind == FALSE) {
    MSE.pop <- (sum((q.pop.MSE[!is.na(q.pop.MSE)]-
bisection(f,0,10))**2)/length(q.pop.MSE[!is.na(q.pop.MSE)]))
  }
  MC.diff = boot.MSE.j-MSE.pop

  return (MC.diff)
}

N=10000
B=1
q=0.05
plan(multisession)

tempjob1 %<-% MC.dist.diff.MSE(N,B,n=300,q,c(7.378,
6.739),rweibull,qweibull,NULL,FALSE,FALSE)
tempjob2 %<-% MC.dist.diff.MSE(N,B,n=500,q,c(7.378,
6.739),rweibull,qweibull,NULL,FALSE,FALSE)
tempjob3 %<-% MC.dist.diff.MSE(N,B,n=1000,q,c(7.378,
6.739),rweibull,qweibull,NULL,FALSE,FALSE)
tempjob4 %<-% MC.dist.diff.MSE(N,B,n=2000,q,c(7.378,
6.739),rweibull,qweibull,NULL,FALSE,FALSE)
tempjob5 %<-% MC.dist.diff.MSE(N,B,n=3000,q,c(7.378,
6.739),rweibull,qweibull,NULL,FALSE,FALSE)
tempjob6 %<-% MC.dist.diff.MSE(N,B,n=5000,q,c(7.378,
6.739),rweibull,qweibull,NULL,FALSE,FALSE)
# tempjob2 %<-% MC.dist.diff.MSE(N,B,n,q,c(1.951,
0.286),rlnorm,qlnorm,NULL,FALSE,FALSE)
# tempjob3 %<-% MC.dist.diff.MSE(N,B,n,q,c(6.319,
0.601),rgumbel,qgumbel,NULL,FALSE,TRUE)
# tempjob4 %<-% MC.dist.diff.MSE(N,B,n,q,c(16.168,
1/0.440),rgamma,qgamma,NULL,FALSE,FALSE)
# tempjob5 %<-%
MC.dist.diff.MSE(N,B,n,q,c(0.7596791,6.283595,1.164798,8.432113,0.8719328),
rnorm,NULL,pnorm,TRUE,FALSE)
# tempjob6 %<-%
MC.dist.diff.MSE(N,B,n,q,c(0.6379,1.980,0.166,1.746,0.226),rlnorm,NULL,plno
rm,TRUE,FALSE)
# tempjob7 %<-% MC.dist.diff.MSE(N,B,n,q,c(7.378,
6.739),rweibull,NULL,pweibull,TRUE,FALSE)


start_time <- Sys.time()
temp.list <- lapply(ls(pattern = "temp"), get)
df  <-  as.data.frame(matrix(unlist(temp.list),
nrow=length(unlist(temp.list[1]))))
names(df) <- c('300','500','1000','2000','3000','5000')
rm(list=ls(pattern="temp"))
mar.default <- c(5,1,2,1) + 0.1
par(mar = mar.default + c(0, 4, 0, 0))
boxplot(df,ylab = expression(hat(M^i)[n] - M[n]),ylim = c(-0.05,0.2), xlab
= 'Sample size')
abline(h=0,col='red', lwd = 3)
end_time <- Sys.time()
end_time - start_time
```

```
## ## MIX and MIX7 RMSE of quantiles for all models
#MOR1
library(data.table)
library(future)
library(ordinal)
library(NLRoot)

MIX7 <- function(N,q,rdens,qdens,pdens,param,mix.ind,gum.ind) {
  bisection <- function(f, a, b, n = 1000, tol = 1e-7) {
    # If the signs of the function at the evaluated points, a and b, stop
the function and return message.
    if (!(f(a) < 0) && (f(b) > 0)) {
      stop('signs of f(a) and f(b) differ')
    }
    else if ((f(a) > 0) && (f(b) < 0)) {
      stop('signs of f(a) and f(b) differ')
    }

    for (i in 1:n) {
      c <- (a + b) / 2 # Calculate midpoint

      # If the function equals 0 at the midpoint or the midpoint is below
the desired tolerance, stop the
      # function and return the root.
      if ((f(c) == 0) || ((b - a) / 2) < tol) {
        return(c)
      }

      # If another iteration is required,
      # check the signs of the function at the points c and a and reassign
      # a or b accordingly as the midpoint to be used in the next
iteration.
      ifelse(sign(f(c)) == sign(f(a)),
             a <- c,
             b <- c)
    }
    # If the max number of iterations is reached and no root has been
found,
    # return message and end function.
    print('Too many iterations')
  }
  f1        <- function(x) {par[[1]]*pweibull(x,par[[2]][1],par[[3]][1]) +
     (1-par[[1]])*pweibull(x,par[[2]][2],par[[3]][1]) - q}
  f         <- function(x) {param[1]*pdens(x,param[2],param[3]) +
     (1-param[1])*pdens(x,param[4],param[5]) - q}
  em_wc   <- function(DS) {
    count  <- 0
    #initialise mixing probabilities to be equal
    probs  <- 0.5
    #initialise parameters
    alpha <- c(6.823130, 11 + rnorm(1,0,1))
    eta    <- c(7.173, 7.173 + rnorm(1,0,1))
    C      <- quantile(DS, probs = 0.7, type = 3)
    DS_ordered = DS[order(DS)]
    x          = DS_ordered[(DS_ordered<=C)]
    m          = length(x)
    n          = length(DS)
    diff       = 1e+11

    while(count < 800)
    {
```

```
      #likelihood function
      weibull <- cbind(dweibull(x, alpha[1], eta[1], log = FALSE),
dweibull(x, alpha[2], eta[2], log = FALSE))

      #responsibilities

      gam <- rbind(cbind((probs*weibull[,1])/(probs*weibull[,1] + (1-
probs)*weibull[,2]),
                         ((1-probs)*weibull[,2])/(probs*weibull[,1] + (1-
probs)*weibull[,2])),
                   cbind((probs*(1-pweibull(C,alpha[1], eta[1],lower.tail =
TRUE, log.p = FALSE))/(probs*(1-pweibull(C,alpha[1], eta[1],lower.tail =
TRUE, log.p = FALSE))

+(1- probs)*(1-pweibull(C,alpha[2], eta[2],lower.tail = TRUE, log.p =
FALSE)))),
                         ((1-probs)*(1-pweibull(C,alpha[2],
eta[2],lower.tail = TRUE, log.p = FALSE))/(probs*(1-pweibull(C,alpha[1],
eta[1]))

+ (1- probs)*(1-pweibull(C,alpha[2], eta[2],lower.tail = TRUE, log.p =
FALSE))))))

      #estimate parameters
      alpha_new <- cbind((sum(head(gam[,1],m)*(x^alpha[1])*log(x))+(n-
m)*tail(gam[,1],1)*C^alpha[1]*log(C))/(sum(head(gam[,1],m)*x^alpha[1])+(n-
m)*tail(gam[,1],1)*C^alpha[1]) -
sum(head(gam[,1],m)*log(x))/sum(head(gam[,1],m)),
                         (sum(head(gam[,2],m)*(x^alpha[2])*log(x))+(n-
m)*tail(gam[,2],1)*C^alpha[2]*log(C))/(sum(head(gam[,2],m)*x^alpha[2])+(n-
m)*tail(gam[,2],1)*C^alpha[2]) -
sum(head(gam[,2],m)*log(x))/sum(head(gam[,2],m)))

      eta_new <- cbind(((sum(head(gam[,1],m)*x^(1/alpha_new[1]))+(n-
m)*tail(gam[,1],1)*C^(1/alpha_new[1]))/sum(head(gam[,1],m)))^(alpha_new[1])
,
                       ((sum(head(gam[,2],m)*x^(1/alpha_new[2]))+(n-
m)*tail(gam[,2],1)*C^(1/alpha_new[2]))/sum(head(gam[,2],m)))^(alpha_new[2])
)

      probs_new <- (sum(head(gam[,1],m)) + tail(gam[,1],1)*(n-
m))/length(DS)

      #update parameters
      a0    <- sum(abs((alpha - 1/alpha_new)) + abs((eta - eta_new)) +
abs((probs - probs_new)))
      probs <- probs_new
      alpha <- 1/alpha_new
      eta   <- eta_new

      centres     <- kmeans(DS,2)
      alpha[1]    <- ifelse(alpha[1]>=25, centres$centers[1], alpha[1])
      alpha[2]    <- ifelse(alpha[2]>=25, centres$centers[2], alpha[2])
      count    <- count+1
      a2  <- ifelse(count>=500,0.0001,0)
      a3  <- ifelse(is.na(a0), 0.0001, 0)
      d   <- a0+a2+a3
      diff <- d
    }

    return(list(probs,alpha, eta))
```

```
  }
  N         <- N
  q         <- q
  q.estimate <- matrix(nrow=N, ncol = 1)
  for (j in (1:N)) {

    x <- tryCatch(
      {

        if(mix.ind == FALSE && gum.ind==FALSE){
          data  <- rdens(300, param[1], param[2])
        }
        if(mix.ind == TRUE && gum.ind==FALSE){

          ind  <- runif(300, 0, 1)
          data  <-  (ind <= param[1])*rdens(300, param[2], param[3]) +
            (ind  > param[1])*rdens(300, param[4], param[5])
        }
        if(mix.ind == FALSE && gum.ind==TRUE){
          data  <- rdens(300, param[1], param[2],max = FALSE)
        }

        par    <-  em_wc(data)
        test   <-  bisection(f1,0,10)
      },
      error = function(e){
        test <- NA
      }
    )
    q.estimate[j] <- x

  }
  if (mix.ind == FALSE && gum.ind == FALSE){
    MSE <- ((1/length(q.estimate[!is.na(q.estimate)]))*sum((q.estimate-
qdens(0.05, param[1], param[2]))**2))
  }
  if (mix.ind == FALSE && gum.ind == TRUE){
    MSE <- ((1/length(q.estimate))*sum((na.omit(q.estimate)-qgumbel(0.05,
param[1], param[2], max = FALSE))**2))
  }
  if (mix.ind == TRUE && gum.ind == FALSE) {
    MSE <- (sum((q.estimate[!is.na(q.estimate)]-
bisection(f,0,10))**2)/length(q.estimate[!is.na(q.estimate)]))
  }
  #q.estimate <- NULL
  sd_2  <- 100*apply(na.omit(q.estimate),2,sd)
  bias  <- 100*sqrt(abs(MSE - (sd_2/100)^2))
  return(list(sqrt(MSE),sd_2,bias))
}

q=0.05
N=500

plan(multisession)

##MOR1
tempjob1 %<-% MIX7(N,q,rweibull,qweibull,NULL,c(6.822, 7.173),FALSE,FALSE)
tempjob2 %<-% MIX7(N,q,rgamma,qgamma,NULL,c(12.93, 1/0.601),FALSE,FALSE)
tempjob3 %<-% MIX7(N,q,rlnorm,qlnorm,NULL,c(2.072, 0.336),FALSE,FALSE)
tempjob4 %<-% MIX7(N,q,rgumbel,qgumbel,NULL,c(6.319, 0.601),FALSE,TRUE)
```

```
tempjob5 %<-%
MIX7(N,q,rnorm,NULL,pnorm,c(0.5629,5.953,0.970,7.676,1.215),TRUE,FALSE)
tempjob6 %<-%
MIX7(N,q,rlnorm,NULL,plnorm,c(0.9758,1.897,0.189,1.245,0.102),TRUE,FALSE)
tempjob7 %<-%
MIX7(N,q,rweibull,NULL,pweibull,c(0.7448,5.494,7.599,15.81,5.983),TRUE,FALS
E)

##MOR2
tempjob1 %<-% MIX7(N,q,rweibull,qweibull,NULL,c(7.378, 6.738),FALSE,FALSE)
tempjob2 %<-% MIX7(N,q,rgamma,qgamma,NULL,c(16.16, 1/0.4407),FALSE,FALSE)
tempjob3 %<-% MIX7(N,q,rlnorm,qlnorm,NULL,c(1.976, 0.2916),FALSE,FALSE)
tempjob4 %<-% MIX7(N,q,rgumbel,qgumbel,NULL,c(6.319, 0.601),FALSE,TRUE)
tempjob5 %<-%
MIX7(N,q,rnorm,NULL,pnorm,c(0.5406,5.924,1.042,7.859,1.095),TRUE,FALSE)
tempjob6 %<-%
MIX7(N,q,rlnorm,NULL,plnorm,c(0.6649,1.976,0.167,1.736,0.226),TRUE,FALSE)
tempjob7 %<-%
MIX7(N,q,rweibull,NULL,pweibull,c(0.7932,5.427,7.642,12.01,6.186),TRUE,FALS
E)

s <- Sys.time()
temp.list <- lapply(ls(pattern = "temp"), get)
results2 <-  as.data.frame(matrix(unlist(temp.list),
nrow=length(unlist(temp.list[1]))))
#names(results) <- c("wei", "gam", "lnorm")
e <- Sys.time()
e-s

# MIX and MIX7 supplement
library(data.table)
library(future)
library(ordinal)
library(NLRoot)
library(ggplot2)

C.select <- function(MC,rdens,qdens,pdens,param,mix.ind,gum.ind) {
  bisection <- function(f, a, b, n = 1000, tol = 1e-7) {
    # If the signs of the function at the evaluated points, a and b, stop
the function and return message.
    if (!(f(a) < 0) && (f(b) > 0)) {
      stop('signs of f(a) and f(b) differ')
    } else if ((f(a) > 0) && (f(b) < 0)) {
      stop('signs of f(a) and f(b) differ')
    }

    for (i in 1:n) {
      c <- (a + b) / 2 # Calculate midpoint

      # If the function equals 0 at the midpoint or the midpoint is below
the desired tolerance, stop the
      # function and return the root.
      if ((f(c) == 0) || ((b - a) / 2) < tol) {
        return(c)
      }

      # If another iteration is required,
      # check the signs of the function at the points c and a and reassign
      # a or b accordingly as the midpoint to be used in the next
iteration.
      ifelse(sign(f(c)) == sign(f(a)),
```

```
            a <- c,
            b <- c)
    }
    # If the max number of iterations is reached and no root has been
found,
    # return message and end function.
    print('Too many iterations')
  }
  f         <- function(x) {param[1]*pdens(x,param[2],param[3]) +
     (1-param[1])*pdens(x,param[4],param[5]) - 0.05}
  ## censored weibull log-like
  fr_w      <- function(initial) {   ## Censored Weibull
    beta <- initial[1]
    psi <- initial[2]
    (m*log(beta) + m*log(psi) + (beta-1)*sum(log(x)) - psi*sum(x^beta) -
(n-m)*psi*C^beta)*(-1)
  }
  ## Monte Carlo simulations
  MC        <- MC
  ## store threshold with MSE values
  model.thresh <- matrix(nrow = length(seq(0.05,1,0.05)), ncol = 1)
  k            <- 1
  for (i in seq(0.05,1,0.05)) {


    ## store the quantile estimate values
    q.estimate <- matrix(nrow=MC, ncol = 1)
    for (j in (1:MC)) {
      if(mix.ind == FALSE && gum.ind==FALSE){
        dat  <- rdens(500, param[1], param[2])
      }
      if(mix.ind == TRUE && gum.ind==FALSE){

        ind  <- runif(500, 0, 1)
        dat  <-  (ind <= param[1])*rdens(500, param[2], param[3]) +
          (ind  > param[1])*rdens(500, param[4], param[5])
      }
      if(mix.ind == FALSE && gum.ind==TRUE){
        dat  <- rdens(500,param[1], param[2],max = FALSE)
      }


      C            = quantile(dat, probs = i, type = 3)
      dat_ordered  = dat[order(dat)]
      x            = abs(dat_ordered[(dat_ordered<=C)])
      m            = length(x)
      n            = length(dat)
      par          = optim(c(0.5,0.5), fr_w)
      q.estimate[j] = qweibull(0.05, par$par[1],
(1/par$par[2])^(1/par$par[1]))
    }

    if (mix.ind == FALSE && gum.ind == FALSE){
      MSE <- ((1/MC)*sum((q.estimate-qdens(0.05, param[1], param[2]))**2))
    }
    if (mix.ind == FALSE && gum.ind == TRUE){
      MSE <- ((1/MC)*sum((q.estimate-qgumbel(0.05, param[1], param[2], max
= FALSE))**2))
    }
    if (mix.ind == TRUE && gum.ind == FALSE) {
```

```
      MSE <- (sum((q.estimate[!is.na(q.estimate)]-
bisection(f,0,10))**2)/length(q.estimate[!is.na(q.estimate)]))
    }
    model.thresh[k] <- MSE
    k=k+1
  }
  return(model.thresh)

}

plan(multisession)
MC = 500

tempjob1 %<-% C.select(MC,rweibull,qweibull,NULL,c(7.378,
6.739),FALSE,FALSE)
tempjob2 %<-% C.select(MC,rgamma,qgamma,NULL,c(16.168,
1/0.440),FALSE,FALSE)
tempjob3 %<-% C.select(MC,rlnorm,qlnorm,NULL,c(1.951, 0.286),FALSE,FALSE)
tempjob4 %<-% C.select(MC,rgumbel,qgumbel,NULL,c(6.319, 0.601),FALSE,TRUE)
tempjob5 %<-%
C.select(MC,rnorm,NULL,pnorm,c(0.7596791,6.283595,1.164798,8.432113,0.87193
28),TRUE,FALSE)
tempjob6 %<-%
C.select(MC,rlnorm,NULL,plnorm,c(0.6379,1.980,0.166,1.746,0.226),TRUE,FALSE
)
tempjob7 %<-%
C.select(MC,rweibull,NULL,pweibull,c(0.7753,5.507,7.676,11.142,6.163),TRUE,
FALSE)

s <- Sys.time()
temp.list <- lapply(ls(pattern = "temp"), get)
results  <-  as.data.frame(matrix(unlist(temp.list),
nrow=length(unlist(temp.list[1]))))
rm(list=ls(pattern="temp"))
e <- Sys.time()
e-s

x <- seq(0.05,1,0.05)

spar = 0.6
plot(x,smooth.spline(x, results[,1], spar=spar)$y,
type='b',col="chartreuse4", lwd=3, pch=16, ylim = c(0,0.14),
     xlab = "Censoring Threshold (Empirical Percentile)", ylab = "MSE of
5th Percentile Estimate")
points(x, smooth.spline(x, results[,2], spar=spar)$y,type='b', pch=17,
lty=2, lwd=3, col = "coral2")
points(x, smooth.spline(x, results[,3], spar=spar)$y,type='b', pch=18,
lty=3, lwd=3, col = "cornflowerblue")
points(x, smooth.spline(x, results[,4], spar=spar)$y,type="b", pch=19,
lty=2, lwd=3, col = "dimgrey")
legend(0, 0.14, legend=c("Weibull", "Gamma", "Log Normal", "Minimum
Gumbel"),
       col=c("chartreuse4", "coral2", "cornflowerblue", "dimgrey"),
       pch=16:19, cex=1.2, box.lty=0,y.intersp=0.3, bg="transparent",
x.intersp = 0.3)

plot(x,smooth.spline(x, results[,5], spar=spar)$y,
type='b',col="darkorchid4", lwd=3, pch=20, ylim = c(0,0.14),
     xlab = "Censoring Threshold (Empirical Percentile)", ylab = "MSE of
5th Percentile Estimate")
```

```
points(x, smooth.spline(x, results[,6], spar=spar)$y,type='b', pch=21,
lty=2, lwd=3, col = "darkred")
points(x, smooth.spline(x, results[,7], spar=spar)$y,type='b', pch=22,
lty=3, lwd=3, col = "mediumseagreen")
legend(0, 0.14, legend=c("Normal Mixture", "Log-normal Mixture", "Weibull
Mixture"),
       col=c("darkorchid4", "darkred", "mediumseagreen"),
       pch=20:22, cex=1.2, box.lty=0,y.intersp=0.3, bg="transparent",
x.intersp = 0.3)

p <- ggplot(results, aes(x, V1)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ poly(x, 3), se = FALSE)
p


##############################
####    Chapter 4 code    ####
##############################

## SWAKS-MLE RMSE of quantiles for all models
#MOR1
library(data.table)
library(future)
library(ordinal)
library(NLRoot)

AKSMLE <- function(N,q,rdens,qdens,pdens,param,mix.ind,gum.ind) {
  bisection <- function(f, a, b, n = 1000, tol = 1e-7) {
    # If the signs of the function at the evaluated points, a and b, stop
the function and return message.
    if (!(f(a) < 0) && (f(b) > 0)) {
      stop('signs of f(a) and f(b) differ')
    }
    else if ((f(a) > 0) && (f(b) < 0)) {
      stop('signs of f(a) and f(b) differ')
    }

    for (i in 1:n) {
      c <- (a + b) / 2 # Calculate midpoint

      # If the function equals 0 at the midpoint or the midpoint is below
the desired tolerance, stop the
      # function and return the root.
      if ((f(c) == 0) || ((b - a) / 2) < tol) {
        return(c)
      }

      # If another iteration is required,
      # check the signs of the function at the points c and a and reassign
      # a or b accordingly as the midpoint to be used in the next
iteration.
      ifelse(sign(f(c)) == sign(f(a)),
             a <- c,
             b <- c)
    }
    # If the max number of iterations is reached and no root has been
found,
    # return message and end function.
    print('Too many iterations')
  }
```

```
  f         <- function(data) {param[1]*pdens(x,param[2],param[3]) +
      (1-param[1])*pdens(x,param[4],param[5]) - q}
  fr <- function(param2) {  ## Censored Weibull Type II
    beta <- param2[1]
    psi <- param2[2]
    (m*log(beta) + m*log(psi) + (beta-1)*sum(log(x)) - psi*sum(x^beta) -
(n-m)*psi*C^beta)*(-1)
  }
  TKS <- function(p,data) {
    fr <- function(param2) {  ## Censored Weibull Type II
      beta <- param2[1]
      psi <- param2[2]
      (m*log(beta) + m*log(psi) + (beta-1)*sum(log(x)) - psi*sum(x^beta) -
(n-m)*psi*C^beta)*(-1)
    }

    C            <- quantile(data, probs = p, type = 3)
    Data_ordered <- data[order(data)]
    x            <- Data_ordered[(Data_ordered<=C)]
    m            <- length(x)
    n            <- length(data)
    par          <- optim(c(0.5,0.5), fr)
    emp          <- ecdf(Data_ordered)
    true         <- emp(x)
    fit          <- pweibull(x, par$par[1], (1/par$par[2])^(1/par$par[1]))
    D1           <- max(abs(log(fit/true))*sqrt((fit)*(1-fit)/m))
    return (c(D1,p))
  }


  N <- N
  q <- q
  Candidate.C <- matrix(seq(0.1,0.5,0.01))
  q.estimate <- matrix(nrow=N, ncol = 1)
  for (j in (1:N)) {

    y <- tryCatch(
      {

        if(mix.ind == FALSE && gum.ind==FALSE){
          data  <- rdens(3000, param[1], param[2])
        }
        if(mix.ind == TRUE && gum.ind==FALSE){

          ind  <- runif(3000, 0, 1)
          data  <-  (ind <= param[1])*rdens(3000, param[2], param[3]) +
            (ind  > param[1])*rdens(3000, param[4], param[5])
        }
        if(mix.ind == FALSE && gum.ind==TRUE){
          data  <- rdens(3000, param[1], param[2],max = FALSE)
        }

        df<-data.frame(t(apply(Candidate.C,1,FUN = function(y)
TKS(y,data))))
        smoothingSpline = smooth.spline(df$X2, df$X1, all.knots = TRUE)
        df$X1  <- smoothingSpline$y
        C      <- quantile(data, df[which.min(df[,1]),2], type = 3)
        x.star_ordered <- data[order(data)]
        x      <- x.star_ordered[(x.star_ordered<=C)]
        m      <- length(x)
        n      <- length(data)
```

```
        par    <- optim(c(0.5,0.5), fr)
        test <-  qweibull(q, par$par[1], (1/par$par[2])^(1/par$par[1]))
      },
      error = function(e){
        test <- NA
      }
    )
    q.estimate[j] <- y

  }
  # if (mix.ind == FALSE && gum.ind == FALSE){
  #   MSE <- ((1/N)*sum((q.estimate-qdens(0.05, param[1], param[2]))**2))
  # }
  # if (mix.ind == FALSE && gum.ind == TRUE){
  #   MSE <- ((1/N)*sum((q.estimate-qgumbel(0.05, param[1], param[2], max =
FALSE))**2))
  # }
  # if (mix.ind == TRUE && gum.ind == FALSE) {
  #   MSE <- (sum((q.estimate[!is.na(q.estimate)]-
bisection(f,0,10))**2)/length(q.estimate[!is.na(q.estimate)]))
  # }
  # #q.estimate <- NULL
  # rm(x.star_ordered,x,m,par,C,data)
  return(q.estimate)
}

q=0.05
N=100

plan(multisession)

tempjob1 %<-% AKSMLE(N,q,rweibull,qweibull,NULL,c(6.822,
7.173),FALSE,FALSE)
tempjob2 %<-% AKSMLE(N,q,rgamma,qgamma,NULL,c(12.93, 1/0.601),FALSE,FALSE)
tempjob3 %<-% AKSMLE(N,q,rlnorm,qlnorm,NULL,c(2.072, 0.336),FALSE,FALSE)
tempjob4 %<-% AKSMLE(N,q,rgumbel,qgumbel,NULL,c(6.620, 0.650),FALSE,TRUE)
tempjob5 %<-%
AKSMLE(N,q,rnorm,NULL,pnorm,c(0.5629,5.953,0.970,7.676,1.215),TRUE,FALSE)
tempjob6 %<-%
AKSMLE(N,q,rlnorm,NULL,plnorm,c(0.9758,1.897,0.189,1.245,0.102),TRUE,FALSE)
tempjob7 %<-%
AKSMLE(N,q,rweibull,NULL,pweibull,c(0.7448,5.494,7.599,15.81,5.983),TRUE,FA
LSE)

s <- Sys.time()
temp.list <- lapply(ls(pattern = "temp"), get)
results3  <-  as.data.frame(matrix(unlist(temp.list),
nrow=length(unlist(temp.list[1]))))
#names(results) <- c("wei", "gam", "lnorm")
e <- Sys.time()
e-s
rm(temp.list)

#MOR2
plan(multisession)

tempjob1 %<-% AKSMLE(N,q,rweibull,qweibull,NULL,c(7.378,
6.738),FALSE,FALSE)
tempjob2 %<-% AKSMLE(N,q,rgamma,qgamma,NULL,c(16.16, 1/0.4407),FALSE,FALSE)
tempjob3 %<-% AKSMLE(N,q,rlnorm,qlnorm,NULL,c(1.976, 0.2916),FALSE,FALSE)
tempjob4 %<-% AKSMLE(N,q,rgumbel,qgumbel,NULL,c(6.319, 0.601),FALSE,TRUE)
```

```
tempjob5 %<-%
AKSMLE(N,q,rnorm,NULL,pnorm,c(0.5406,5.924,1.042,7.859,1.095),TRUE,FALSE)
tempjob6 %<-%
AKSMLE(N,q,rlnorm,NULL,plnorm,c(0.6649,1.976,0.167,1.736,0.226),TRUE,FALSE)
tempjob7 %<-%
AKSMLE(N,q,rweibull,NULL,pweibull,c(0.7932,5.427,7.642,12.01,6.186),TRUE,FA
LSE)

s <- Sys.time()
temp.list <- lapply(ls(pattern = "temp"), get)
results21 <-  as.data.frame(matrix(unlist(temp.list),
nrow=length(unlist(temp.list[1]))))
#names(results) <- c("wei", "gam", "lnorm")
e <- Sys.time()
e-s

w1     <- na.omit(results21$V1)
g1     <- na.omit(results21$V2)
ln1    <- na.omit(results21$V3)
gu1    <- na.omit(results21$V4)
nm1    <- na.omit(results21$V5)
lnm1   <- na.omit(results21$V6)
wm1    <- na.omit(results21$V7)
MSE_w  <- (1/length(w1))*sum((w1 - qweibull(0.05, 7.378, 6.738))^2)
MSE_g  <- (1/length(g1))*sum((g1 - qgamma(0.05, 16.16, 1/0.4407))^2)
MSE_ln <- (1/length(ln1))*sum((ln1 - qlnorm(0.05, 1.976, 0.2916))^2)
MSE_gu <- (1/length(gu1))*sum((gu1 - qgumbel(0.05, 6.319, 0.601, max =
FALSE))^2)
MSE_wm <- (sum((wm1[!is.na(wm1)]-
bisection(f,0,10))**2)/length(wm1[!is.na(wm1)]))
MSE_nm <- (sum((nm1[!is.na(nm1)]-
bisection(f,0,10))**2)/length(nm1[!is.na(nm1)]))
MSE_lnm<- (sum((lnm1[!is.na(lnm1)]-
bisection(f,0,10))**2)/length(lnm1[!is.na(lnm1)]))
mse    <- c(MSE_w,MSE_g,MSE_ln,MSE_gu,MSE_nm,MSE_lnm,MSE_wm)
rmse_m2<- sapply(mse,sqrt)
sd_m2  <- 100*sapply(na.omit(results21),sd)
bias_m2<- 100*sqrt(abs(mse - (sd_m2/100)^2)
```