2011-03-03

Parameter Estimation for the Two-Parameter Weibull Distribution

Mark A. Nielsen
Brigham Young University - Provo

Parameter Estimation for the Two-Parameter Weibull Distribution

Mark A. Nielsen

A project submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

David A. Engler PhD, Chair
W. Evan Johnson PhD
H. Dennis Tolley PhD

Department of Statistics

Brigham Young University

April 2011

ABSTRACT

Parameter Estimation for the Two-Parameter Weibull Distribution

Mark A. Nielsen
Department of Statistics, BYU
Master of Science

The Weibull distribution, an extreme value distribution, is frequently used to model survival, reliability, wind speed, and other data. One reason for this is its flexibility; it can mimic various distributions like the exponential or normal. The two-parameter Weibull has a shape (γ) and scale (β) parameter. Parameter estimation has been an ongoing search to find efficient, unbiased, and minimal variance estimators. Through data analysis and simulation studies, the following three methods of estimation will be discussed and compared: maximum likelihood estimation (MLE), method of moments estimation (MME), and median rank regression (MRR). The analysis of wind speed data from the TW Daniels Experimental Forest are used for this study to test the performance and flexibility of the Weibull distribution.

Keywords: weibull, extreme value distribution, parameter estimation, wind speed, TWDEF

ACKNOWLEDGMENTS

I appreciate the help and time Dr. Engler gave to assist in the completion of this project. I also thank those family and friends that helped in the editing and revisions of my final project. I also appreciate the opportunity to work with Dr. Scott Jones, and for the use of the TWDEF data.

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 The Two-Parameter Weibull Distribution

There are many applications for the Weibull distribution in statistics. Although it was first identified by Fréchet in 1927, it is named after Waalobi Weibull and is a cousin to both the Fréchet and Gumbel distributions. Waalobi Weibull was the first to promote the usefulness of this distribution by modelling data sets from various disciplines (Murthy, Xie, and Jiang 2004). If $T \sim$ Weibull(γ,β) then its density function is defined as:

$$f(t|\gamma,\beta) = \frac{\gamma}{\beta^{\gamma}} t^{(\gamma-1)} \exp\left\{-\left(\frac{t}{\beta}\right)^{\gamma}\right\}, \text{for } \gamma > 0, \text{ and } \beta > 0.$$

In this density function, our γ represents the shape parameter, and our β represents the scale parameter. The distribution function can also be derived and is defined as:

$$F(t|\gamma,\beta) = 1 - \exp\left\{-\left(\frac{t}{\beta}\right)^{\gamma}\right\},$$

where, as noted before, γ and β are non-negative.

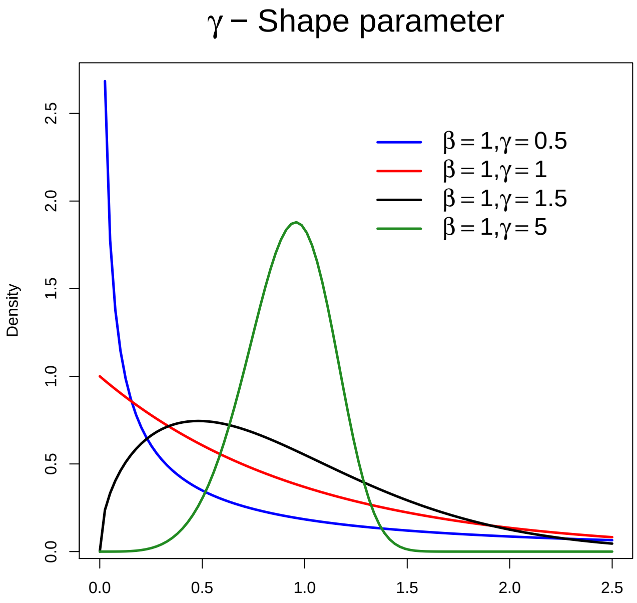


Figure 1.1: Examples of the Weibull density curve with various values of γ.

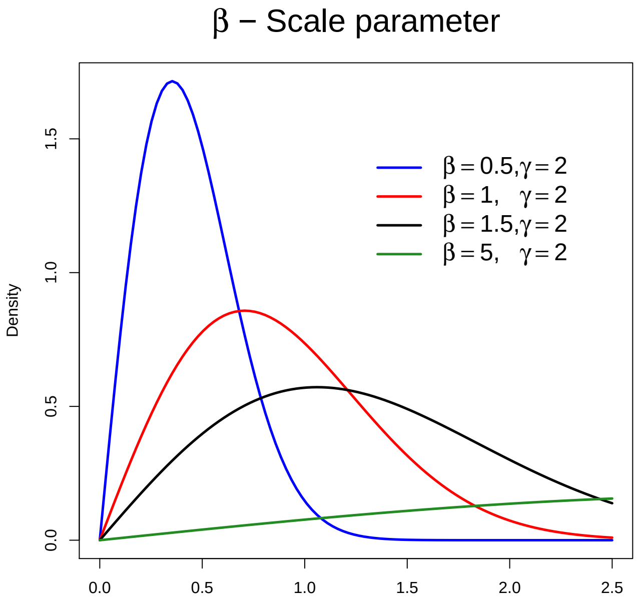


Figure 1.2: Examples of the Weibull density curve with various values of β.

1.2 Applications of the Weibull Distribution

Extreme value theory is a unique statistical discipline that develops "models for describing the unusual rather than the usual (Coles 2001)." Perhaps the simplest example of an extreme value distribution is the exponential distribution. The Weibull distribution is specifically used to model extreme value data. One example of this is the frequent use of the Weibull distribution to model failure time data (Murthy et al. 2004). Its use is also applicable in various situations including data containing product failure times to data containing survival times of cancer patients. The extreme values in these analyses would be the unusual longevity of the shelf life of a product or survival of affected patients. One reason it is widely used in reliability and life data analysis is due to its flexibility. It can mimic various distributions like the normal. Special cases of the Weibull include the exponential ($\gamma = 1$) and Rayleigh ($\gamma = 2$) distributions. Because of its uses in lifetime analysis, a more useful function is the probability that the lifetime exceeds any given time, (i.e. $\mathrm{P}(T > t)$). This is called the survival function or in the case of a product, the reliability. For the Weibull distribution, this is the following equation:

$$S(t) = 1 - P(T \leq t) = 1 - F(t|\gamma, \beta) = \exp\left\{-\left(\frac{t}{\beta}\right)^{\gamma}\right\}.$$

Another function of interest is the hazard function, which is often called the instantaneous failure rate. The hazard function is computed using both the reliability or survival function and the density function, $h(t) = \frac{f(t)}{S(t)}$. For the Weibull distribution, this is derived as follows:

$$h(t) = \frac{f(t)}{S(t)} = \frac{\frac{\gamma}{\beta^{\gamma}} t^{(\gamma-1)} \exp\left\{-\left(\frac{t}{\beta}\right)^{\gamma}\right\}}{\exp\left\{-\left(\frac{t}{\beta}\right)^{\gamma}\right\}} = \frac{\gamma}{\beta^{\gamma}} t^{(\gamma-1)}.$$

An interesting property of this equation is that this function is non-constant in comparison to the hazard function for the exponential distribution (i.e. $h(t) = \lambda$, for $f(t|\lambda) = \lambda e^{-\lambda t}$). This reveals an interesting property of the Weibull distribution. When the parameter $\gamma > 1$, it is said the "instantaneous failure rate," or hazard function, increases with time, if $\gamma = 1$ it remains constant, and if $\gamma < 1$ then the hazard function decreases over time (Collett 2009).

Often studies involving product failure times and patient survival times are censored for a given subset of subjects. The motivation to include these data in analyses, is that they do contain information about the reliability or survival times of the subjects. It would be unwise to simply throw out these data points. It would also be inappropriate to treat them the same as the subjects who each experienced their "events." An alternate form of the probability density function is suggested, as follows:

$$f(t_i|\gamma,\beta) = \left[\frac{\gamma}{\beta^\gamma} t_i^{(\gamma-1)} \exp\left\{-\left(\frac{t_i}{\beta}\right)^\gamma\right\}\right]^{\delta_i} \times \left[\exp\left\{-\left(\frac{t_i}{\beta}\right)^\gamma\right\}\right]^{(1-\delta_i)},$$

where γ and β are nonnegative and δ_i is the indicator function of the i^{th} subject's event. (i.e. $\delta_i = 1$ if the "event" has occurred.) This study will not address censored data.

Another discipline where the Weibull is applicable is in the analysis of wind speeds. According to Weisser (2003), wind energy, in comparison to other renewable sources of energy, such as solar or tidal energy, "has a more variable and diffuse energy flux." Because of this variability, the Weibull distribution is used to model the variation of wind speeds in order to maximize the benefit obtained from wind energy. This information can be used to optimize the design of wind energy conversion technologies (WECTs). In the past, the two-parameter Weibull distribution has been shown to effectively describe the variation of wind speed and is commonly used in modelling such data (Weisser 2003; Seguro and Lambert 2000; Celik 2003). Wind speed data is usually in time series format. It is reasonable to use the Weibull distribution to summarize the information contained in large sets of wind speed data into a couple parameter estimates. This project's focus will be the parameter estimation of wind speed data.

There are several functional and mechanical reasons for using the Weibull distribution in modeling wind speeds. In a study of wind speed durations above and below fixed speeds, Corotis (1979) determined that the Rayleigh distribution is reasonable in modeling wind speeds. In his previous research the following cumulative distribution function had been developed to model run duration times (t):

$$\begin{aligned} F_1(t) &= 1 - \left(\frac{t}{t_0}\right)^{(1-b)} \qquad \text{for } t_0 \leq t \leq t_1, \\ F_2(t) &= 1 - A\exp\{-\lambda t\} \qquad \text{for } t_1 \leq t \leq \infty, \end{aligned}$$

where

$$\begin{aligned} t_0 &= \frac{1}{2}, \\ t_1 &= t_0 G^{\frac{1}{1-b}}, \\ A &= G\exp\{b-1\}, \text{ and} \\ \lambda &= \frac{b-1}{t_1}. \end{aligned}$$

G being approximated at 1/4 for run levels of practical interest. The remaining parameter b can be found using maximum likelihood estimation.

His research showed a strong correlation between the b parameter and seasonal mean wind speeds. He then presented a family of smoothed b-parameter curves, which indicated that the family of curves seemed to vary as a function of the run level to mean speed. The run level being the fixed wind speed. The Rayleigh distribution, a special case of the Weibull distribution (i.e. where $\gamma = 2$), is a one parameter distribution that implies a constant ratio of standard deviation to the mean. This implies a constant ratio of run level to the mean, where run level is considered as the number of standard deviations away from the mean. The reasonable fit of this distribution implies that the higher the run level is in comparison to the mean, the probability that a wind speed will continue to increase or that we will observe a "long duration above the run level" decreases. Now that the usefulness of the Rayleigh distribution has been established, the flexibility of the shape of the distribution is added by using the Weibull distribution because the skewness of a Weibull curve depends only on γ (Hennessey 1977).

Another functional reason for using the Weibull is because wind power (i.e. (wind speed)^3) is easily modeled by using a cubed transformation of the Weibull which is also distributed as a Weibull($\gamma/3$, β) (Hennessey 1977).

CHAPTER 2

PARAMETER ESTIMATION

2.1 Maximum Likelihood Estimator

The maximum likelihood estimator (MLE) is a well known estimator. It is defined by treating our parameters as unknown values and finding the joint density of all observations of a data set, which are assumed to be independent and identically distributed (iid). Once the likelihood function is defined, the maximum of that function is found. If the data points are all highly likely under specific parameter values, then their product will be the “most likely” outcome, or the maximum likelihood. This estimator is important in statistics because of is asymptotic unbiasedness and minimal variance.

Our MLE is computed by first assuming that $T_i \overset{iid}{\sim}$ Weibull(γ, β), with the probability given by its density function:

$$f(t_i|\gamma,\beta) \;=\; \frac{\gamma}{\beta^\gamma} t_i^{(\gamma-1)} \exp\left\{-\left(\frac{t_i}{\beta}\right)^\gamma\right\}, \text{for } \gamma > 0, \text{ and } \beta > 0.$$

The joint density of the likelihood is the product of the densities of each data point.

$$\begin{aligned} \mathrm{L}(\gamma,\beta|\mathbf{t}) &= \prod_{i=1}^{n} f(t_i|\gamma,\beta) \\ &= \prod_{i=1}^{n}\left[\frac{\gamma}{\beta^\gamma} t_i^{(\gamma-1)} \exp\left\{-\left(\frac{t_i}{\beta}\right)^\gamma\right\}\right] \\ &= \left(\frac{\gamma}{\beta^\gamma}\right)^n \prod_{i=1}^{n} t_i^{(\gamma-1)} \times \exp\left\{-\left(\frac{\sum t_i}{\beta}\right)^\gamma\right\}. \end{aligned}$$

Next, to make the math somewhat simpler, the natural log of our joint density is computed. This can be done while still preserving the true maximum of our likelihood function because the log transformation is a monotonically increasing function.

$$\begin{aligned}
\ell(\gamma,\beta|\mathbf{t}) &= \log\left[\left(\frac{\gamma}{\beta^\gamma}\right)^n \prod_{i=1}^{n} t_i^{(\gamma-1)} \times \exp\left\{-\left(\frac{\sum t_i}{\beta}\right)^\gamma\right\}\right] \\
&= \log\left(\frac{\gamma}{\beta^\gamma}\right)^n + \log\left[\prod_{i=1}^{n} t_i^{(\gamma-1)}\right] - \left(\frac{\sum t_i}{\beta}\right)^\gamma \\
&= n\log\left(\frac{\gamma}{\beta^\gamma}\right) + (\gamma-1)\log\left[\prod_{i=1}^{n} t_i\right] - \sum_{i=1}^{n}\left(\frac{t_i}{\beta}\right)^\gamma \\
&= n\log\gamma - n\log\beta^\gamma + (\gamma-1)\sum_{i=1}^{n}\log(t_i) - \sum_{i=1}^{n}\left(\frac{t_i}{\beta}\right)^\gamma \\
&= n\log(\gamma) - \gamma n\log(\beta) + (\gamma-1)\sum_{i=1}^{n}\log(t_i) - \sum_{i=1}^{n}\left(\frac{t_i}{\beta}\right)^\gamma.
\end{aligned}$$

Once the likelihood function has been obtained, the function is optimized by negating it and finding its minimum. This is done iteratively using an optimization function in the *gsl* library in C and using the *nlm* function in R.

2.2 Method of Moments Estimator

Although in many cases the method of moments estimator (MME) is superseded by Fisher's MLE concerning asymptotic unbiasedness and minimal variance, the method of moments estimators can, in many cases and quite acurately, be derived by hand. The MME is defined by computing the sample moments,

$$\mu_k = \frac{1}{n}\sum_{i=1}^{n} t_i^k,$$

and setting them equal to the theoretical moments from the moment generating function, $M_T(t)$. The moment generating function for the Weibull is as follows:

$$M_K(t) = \beta^k\Gamma\left(1+\frac{k}{\gamma}\right),$$

where k represents the k^{th} theoretical moment, and $\Gamma(\cdot)$ represents the gamma function, $\Gamma(\alpha) = \int_0^\infty x^{\alpha-1}e^{-x}dx$, where $\alpha > 0$ (Casella and Berger 2002).

Only the first two moments are needed to derive the parameter estimates for the Weibull. Using the equation, $\frac{\sigma^2}{\mu_1^2} = \frac{M_2(t)-[M_1(t)]^2}{[M_1(t)]^2}$, from Murthy, Xie, and Jiang (2004), we see that it can be simplified to the following equation:

$$\begin{aligned}
\frac{\sigma^2}{\mu_1^2} &= \frac{M_2(t) - [M_1(t)]^2}{[M_1(t)]^2} \\
\Rightarrow \frac{\mu_2 - \mu_1^2}{\mu_1^2} &= \frac{M_2(t)}{[M_1(t)]^2} - 1 \\
\Rightarrow \frac{\mu_2}{\mu_1^2} - 1 &= \frac{M_2(t)}{[M_1(t)]^2} - 1 \\
\Rightarrow \frac{\mu_2}{\mu_1^2} &= \frac{M_2(t)}{[M_1(t)]^2}.
\end{aligned}$$

Next, taking the second sample moment divided by the square of the first sample moment, a function of the theoretical moments using our data is approximated.

$$\begin{aligned}
\frac{\mu_2}{\mu_1^2} &= \frac{M_2(t)}{[M_1(t)]^2} \\
\Rightarrow \frac{\frac{1}{n}\sum_{i=1}^n t_i^2}{\left[\frac{1}{n}\sum_{i=1}^n t_i\right]^2} &\approx \frac{\beta^2\Gamma\left(1+\frac{2}{\hat{\gamma}}\right)}{\beta^2\Gamma^2\left(1+\frac{1}{\hat{\gamma}}\right)} \\
\Rightarrow \frac{n\sum_{i=1}^n t_i^2}{\left[\sum_{i=1}^n t_i\right]^2} &\approx \frac{\Gamma\left(1+\frac{2}{\hat{\gamma}}\right)}{\Gamma^2\left(1+\frac{1}{\hat{\gamma}}\right)}.
\end{aligned}$$

This function solely depends on $\hat{\gamma}$ and therefore we can solve using root finding techniques. I used the bisection method (Jones, Maillardet, and Robinson 2009) because of its robustness and simplicity. It is, however, very slow in comparison to other root finding methods such as the Newton-Raphson method.

We begin by defining our function $f(x)$ as follows:

$$f(x) = \frac{\Gamma\left(1+\frac{2}{x}\right)}{\Gamma^2\left(1+\frac{1}{x}\right)} - \frac{\mu_2}{\mu_1^2} = 0,$$

knowing that μ_2/μ_1^2 is a constant given our data. The bisection method requires two initial points, x_1 and x_2, where $f(x_1)$ and $f(x_2)$ have opposite signs. The interval (x_1,x_2) is divided in half and the half where the end points evaluated in the function have opposite signs is found. These values become the endpoints of our new interval and we divide it again. This process is repeated until the difference of two consecutive midpoints are within ϵ of each other, where $\epsilon > 0$. Once we have found our root, we set this equal to $\hat{\gamma}$. Finally, we can estimate β by the equation

$$\begin{aligned} \mu_1 = \bar{t} &= \hat{\beta}\Gamma\left(1 + \frac{1}{\hat{\gamma}}\right) \\ \Rightarrow \hat{\beta} &= \frac{\bar{t}}{\Gamma\left(1 + \frac{1}{\hat{\gamma}}\right)}, \end{aligned}$$

plugging in the value obtained in our root finding iteration for $\hat{\gamma}$.

2.3 Median Rank Regression Estimator

The last estimator is computed by using median rank regression. This method is the simplest of the three in this paper. Although it has neither the asymptotic properties of the MLE nor the accuracy of the MME estimator, it is quick, simple, and fairly accurate. This method was specifically used in the past as a method that could be done by hand. Thus, it is safe to assume that this method's accuracy will be mediocre in comparison to the previous methods.

The median rank regression estimator includes a simple algorithm. First, the distribution function for our ordered data is approximated. The data are first sorted in ascending order. The data are "ranked" by using the median rank approximation by solving the following equation for Z_i.

$$\sum_{k=i}^{n} \binom{n}{k} (Z_i)^k (1 - Z_i)^{n-k}. \tag{2.1}$$

Because this is computationally intensive we use Bernard's approximation,

$$F_T(t_i) \approx Z_i \approx \frac{i - 0.3}{n + 0.4},$$

which is an estimate to the solution for Equation 2.1, where i is the ascending rank of our data point and n is the total number of data in our data set (ReliaSoft Corportation 1996–2006). These estimates will be used later in setting up a regression model.

Finally, the distribution function is used to derive a linear relation between a transformation of the distribution function estimates and the log survival times.

$$\begin{aligned}
1 - F_T(t) &= \exp\left\{-\left(\frac{t}{\beta}\right)^{\gamma}\right\} \\
\log\left[1 - F_T(t)\right] &= -\left(\frac{t}{\beta}\right)^{\gamma} \\
\log\left[\frac{1}{S(t)}\right] &= \left(\frac{t}{\beta}\right)^{\gamma} \\
\log\left[\log\left[\frac{1}{S(t)}\right]\right] &= \gamma \log t - \gamma \log \beta
\end{aligned}$$

Now we have a linear model and can use regression to find the parameter estimates $\dot{\gamma}$ and $\dot{\beta}$. Allowing $\mathbf{y}$ to equal the twice-logged inverse median-rank estimated survival function, and $\mathbf{x}$ to be equal to the log survival times,

$$\begin{aligned}
\log\left[\log\left[\frac{1}{S(t)}\right]\right] &= \gamma \log t - \gamma \log \beta \\
\Rightarrow \mathbf{y} &= \gamma \mathbf{x} - \gamma \log \beta \\
\Rightarrow \mathbf{y} &= \psi_0 + \psi_1 \mathbf{x},
\end{aligned}$$

$$\begin{aligned}
\text{where } \dot{\gamma} &= \psi_1, \text{ and} \\
\dot{\beta} &= \exp\left\{\frac{-\psi_0}{\dot{\gamma}}\right\}.
\end{aligned}$$

Note that ψ_0 is the intercept of the least squares regression model and ψ_1 is the slope. This concept is illustrated in Figure 2.1 on the following page.

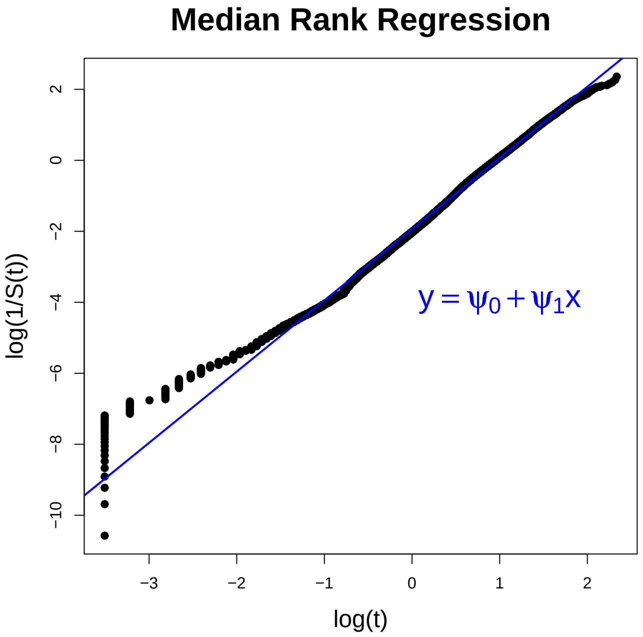


Figure 2.1: The linear relationship of the twice-logged inverse estimated survival times(Z_i) and logged survival times is graphically demonstrated.

CHAPTER 3

SIMULATION STUDIES

The first step in this simulation study will be to determine and justify parameter values and sample sizes in order to generate simulation data. For each combination of parameter values and sample sizes, 1000 data set are simulated. After the new data are generated, the parameter estimation methods discussed in chapter 2 will be used. Because the true parameter values of the different data sets are known, statistics measuring bias and variance of the estimated parameters can be computed using the 1000 repetitions. Once the bias, variance, and mean square error (defined and discussed later) are computed, the different methods can be compared in accuracy. Finally, data is then simulated under three different density functions to determine the accuracy of these parameterizations if misspecification is an issue.

3.1 Justification of Parameters and Sample Sizes

I chose the first parameters,γ and β, to be 2 and 2.5, respectively. I wanted to see how the estimators would handle values similar to the shape and scale parameters from the wind data. Next, I chose a second set of differing values. These values were 5 for γ and 90 for β. I conducted tests of accuracy for these values with small, moderate, and large ($n = 20, 100, 10000$) data sets in order to determine the accuracy of each method for various data sizes.

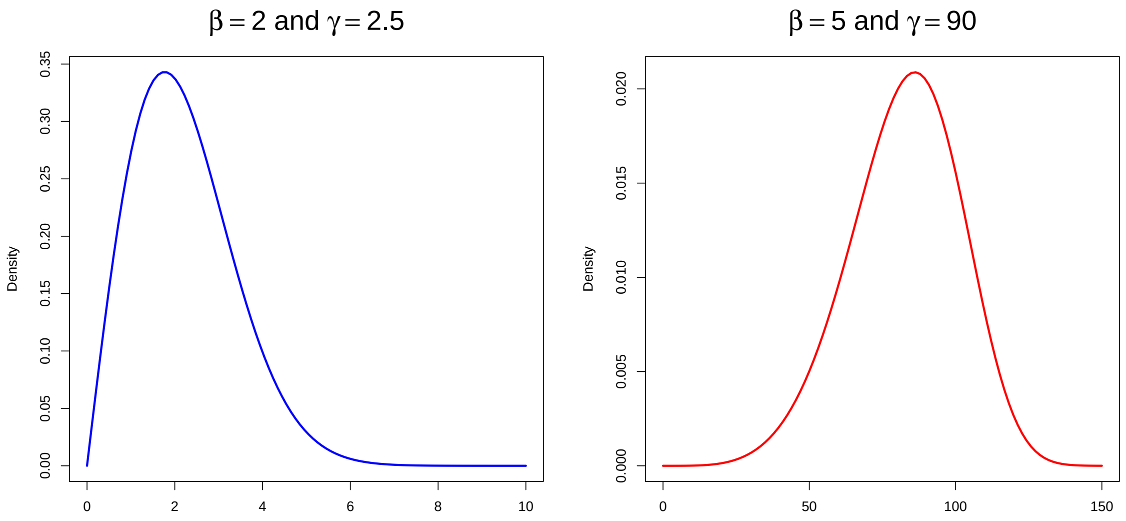


Figure 3.1: Weibull density curves for the parameters specified above. These parameters were selected for the simulation studies.

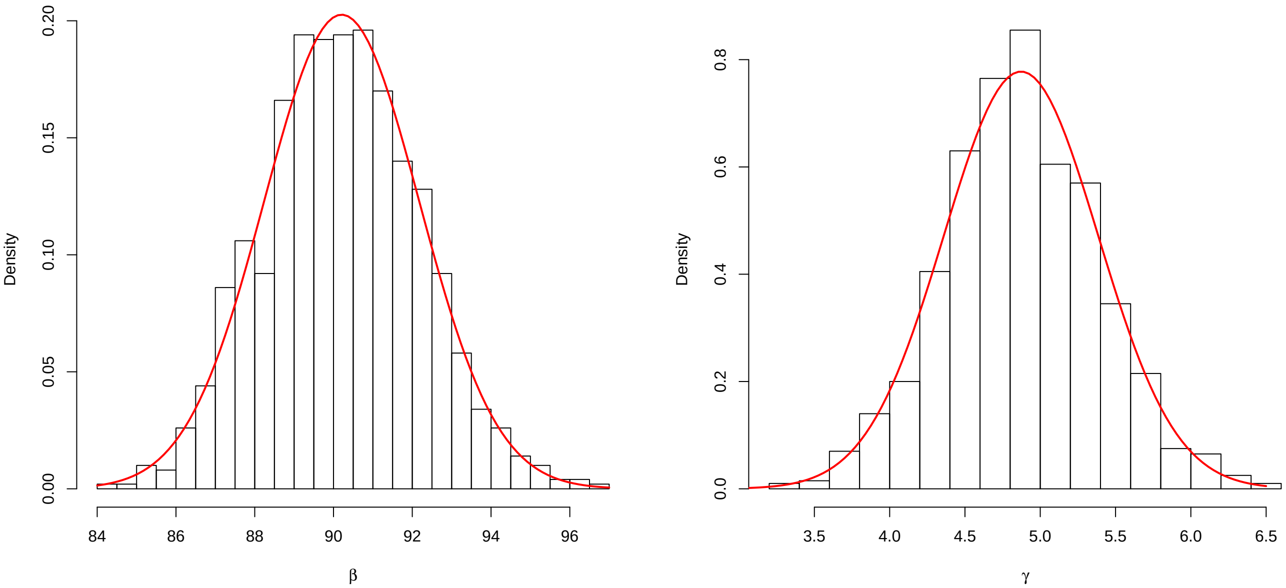


Figure 3.2: **Normally Distributed Parameter Estimates:** Both are distributions of the parameters for the Regression Method, with $\gamma = 5$, $\beta = 90$, and $n = 100$.

After simulating the data sets and finding our parameter estimates, we assume that our parameter estimates are normally distributed. This assumption allows us to take the 1,000 repetitions of each parameter estimation and calculate the mean square error (MSE), or the bias squared plus the variance of the distribution of our parameter estimates. Graphs showing normality for all parameter estimates can be found in Appendix A.

3.2 Bias

Bias is defined as the difference between the true parameter value and the parameter estimate. Mathematically this is defined to be $E(\hat{\theta}) - \theta$, where $\hat{\theta}$ is our estimated parameter and θ is the true parameter value. This is computed by finding the average of each of the 1000 parameter estimates and subtracting the true value, which was used to generate the data.

$$\text{bias} = \frac{1}{n}\sum_{i=1}^{n} x_i - \text{truth}.$$

Below are the biases for each set of parameters and sample sizes.

	$\gamma = 5$			$\beta = 90$		
$n =$	20	100	10000	20	100	10000
MLE	0.38625	0.06758	0.00027	-0.18701	-0.06877	0.00302
MME	0.40456	0.06451	0.00067	-0.23079	-0.07345	0.00277
MRR	-0.16904	-0.12781	-0.00497	0.50473	0.20719	0.01181

Table 3.1: Biases for first set of parameters. Specifically for the MLE bias approaches zero as the sample size, n, gets large. Here all three methods demonstrate the tendency to asymptotic unbiasedness.

	$\gamma = 2$			$\beta = 2.5$		
$n =$	20	100	10000	20	100	10000
MLE	0.15450	0.02703	0.00011	-0.00192	-0.00267	0.00023
MME	0.14956	0.02705	0.00006	-0.00317	-0.00277	0.00022
MRR	-0.06761	-0.05112	-0.00199	0.04678	0.01666	0.00084

Table 3.2: Biases for the second set of parameters. Once again, the bias for all three methods approaches zero as n gets large.

Notice that the bias for the MLE estimates are all smaller than the biases for both MME and MRR estimates. Also, as n increases, the bias goes to zero for all estimators. This is a trait that was expected for the MLE, but which is also expressed in the other two methods. It appears that the MLE and MME converge the quickest to the true value.

3.3 Variance

Variance is defined to be the deviation about the mean. In this case, it is the sample variance for each of the parameters and sample sizes. This is computed by finding the sum of squared deviances of each data point from the data's average value and dividing by $n - 1$:

$$\text{variance} = \frac{1}{n-1}\left[\sum_{i=1}^{n} x_i^2 - \left(\frac{1}{n}\sum_{i=1}^{n} x_i\right)^2\right].$$

The variances for the data simulation studies are shown in the tables below.

	$\gamma = 5$			$\beta = 90$		
$n =$	20	100	10000	20	100	10000
MLE	1.00863	0.16372	0.00160	19.16663	3.64719	0.03487
MME	1.19413	0.18421	0.00183	19.15801	3.65998	0.03489
MRR	1.22024	0.26308	0.00290	20.01155	3.87677	0.03652

Table 3.3: Variances for the first set of parameters. The variances' tendencies toward zero is expected as the sample size increases.

	$\gamma = 2$			$\beta = 2.5$		
$n =$	20	100	10000	20	100	10000
MLE	0.16138	0.02620	0.00026	0.09090	0.01758	0.00017
MME	0.15715	0.02673	0.00026	0.09106	0.01763	0.00017
MRR	0.19524	0.04209	0.00046	0.09732	0.01890	0.00018

Table 3.4: Variances for the second set of parameters.

In Tables 3.3 and 3.4, we see that the variance is smallest for the MLE estimates. However, note that the variances of the MME and MRR estimates are approximately the same as each of the variances for the MLE estimates.

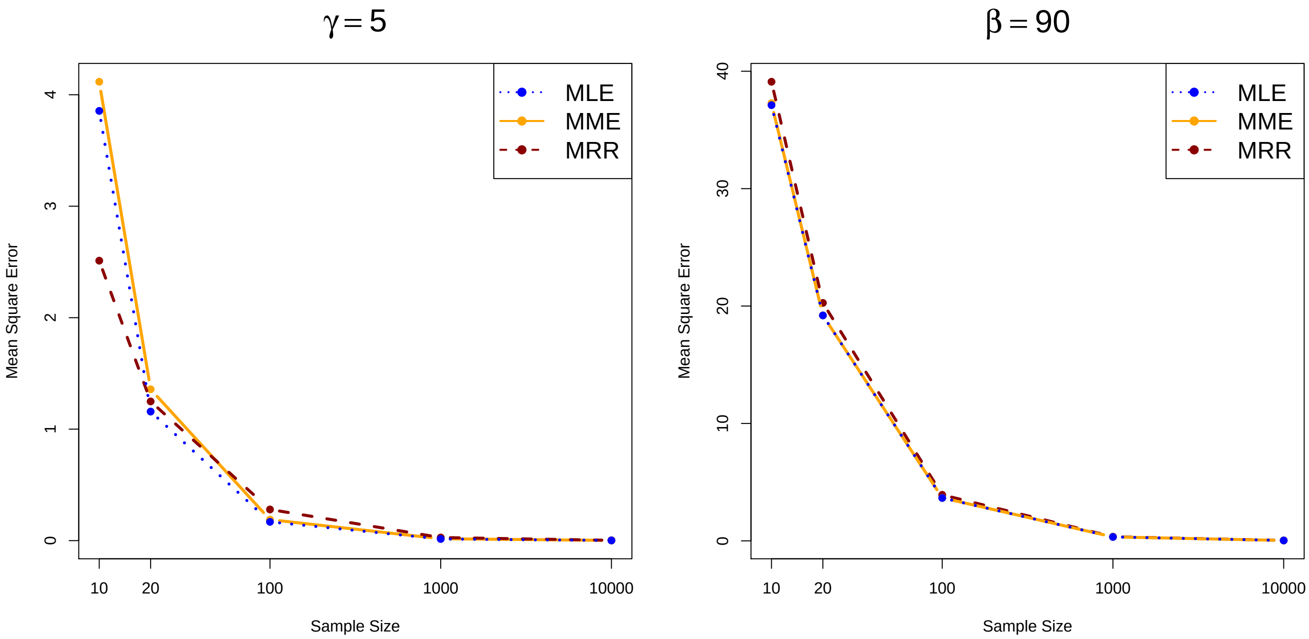
3.4 Mean Square Error

Once we have the measures for bias and variance we can compute the mean square error as defined below:

$$\text{MSE} = \text{bias}^2 + \text{variance}$$

Once again, Tables 3.5 and 3.6 below report the MSE for each set of parameters and sample sizes. Notice also Figure 3.2 which is a graphical representation of the measures of MSE for each method. In all three cases for both parameters the MSE converges to zero, showing that as sample sizes increase, the accuracies of our methods also increase. Figure 3.3 tells a similar story for $\gamma = 2$ and $\beta = 2.5$.

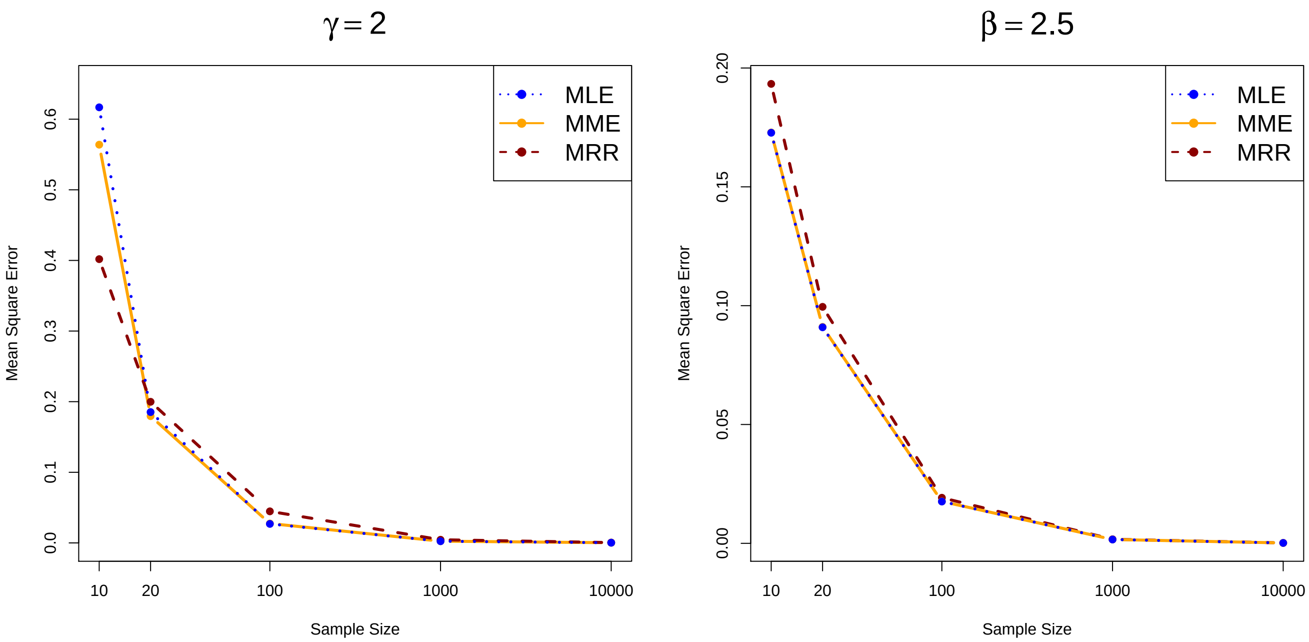
Figure 3.3: These two graphs show the mean square error for various sample sizes. The parameters $\gamma = 5$ and $\beta = 90$ are used for these estimations.



	$\gamma = 5$			$\beta = 90$		
$n =$	20	100	10000	20	100	10000
MLE	1.15782	0.16829	0.00160	19.20161	3.65192	0.03488
MME	1.35779	0.18837	0.00184	19.21128	3.66537	0.03489
MRR	1.24881	0.27942	0.00293	20.26631	3.91970	0.03666

Table 3.5: MSE for the first set of parameters.

Figure 3.4: These two graphs show the mean square error for various sample sizes. The parameters $\gamma = 2$ and $\beta = 2.5$ are used for these estimations.



	$\gamma = 2$			$\beta = 2.5$		
$n =$	20	100	10000	20	100	10000
MLE	0.18525	0.02693	0.00026	0.09090	0.01759	0.00017
MME	0.17952	0.02746	0.00026	0.09107	0.01764	0.00017
MRR	0.19981	0.04471	0.00047	0.09951	0.01918	0.00018

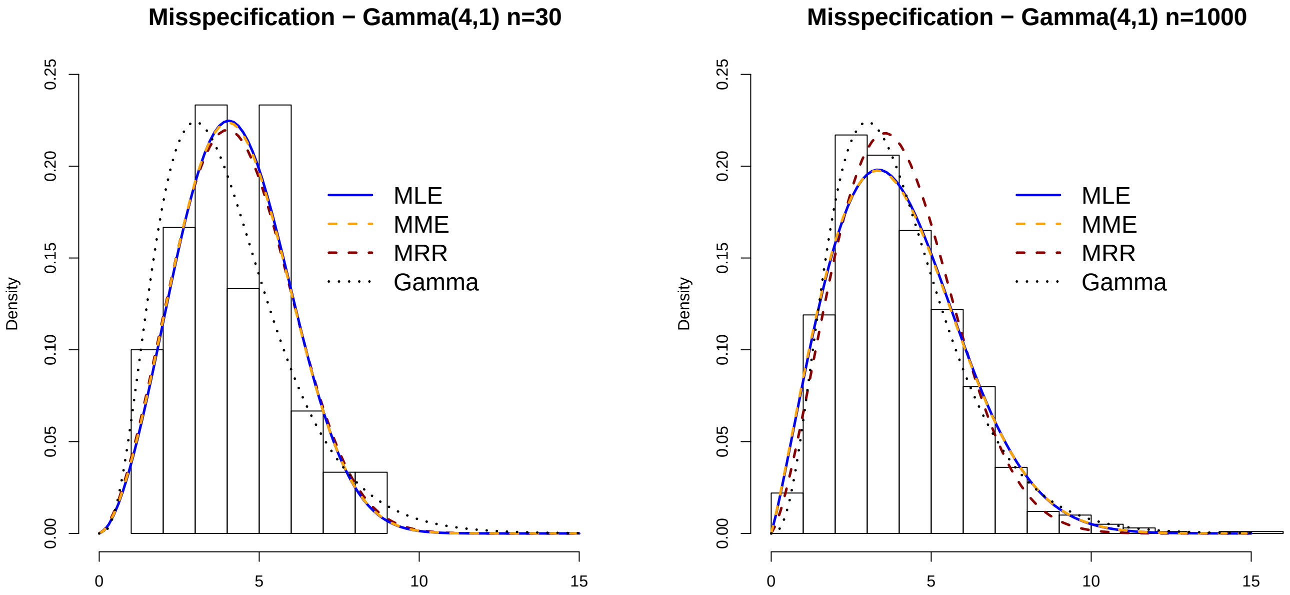
Table 3.6: MSE for second set of parameters.

Notice that the MLE had the smallest MSE for all four parameters. An interesting attribute of the median rank regression estimator is that for small data sets ($n = 10$) the MSE for γ is the smallest. However, we see again that the MLE is the best estimator, according to the MSE.

3.5 Distribution Misspecification

In general, the Weibull distribution is used in modeling a wide variety of data, including wind speed, patient survival, and product lifetime. There is the possibility that these data are not truly distributed as Weibull data. I simulated three different data sets which are distributed as gamma(4,1), normal(5,1), and zero-truncated normal(2,3). After simulating the data with sizes of n =30 and 1000, I used my three methods of parameter estimation for the Weibull to obtain estimates for the six data sets. To determine which parameter estimates fit the data best, I will use the Kolmgorov-Smirnov test in R. The KoImgorov-Smirnov test assesses whether or not data comes from a reference distribution. The null hypothesis is that the generated data is from the Weibull distribution, thus a p-value that is less than $\alpha = 0.05$ would suggest that the data did not come from the Weibull distribution. Figure 3.— below, you will find the histograms of the data with an overlaid Weibull curve with the estimated parameter estimates followed by the p-values for the Kolmgorov-Smirnov (KS) tests.

Figure 3.5: Data misspecification is a plausible error in modelling data. These plots show possible misspecification of the data.

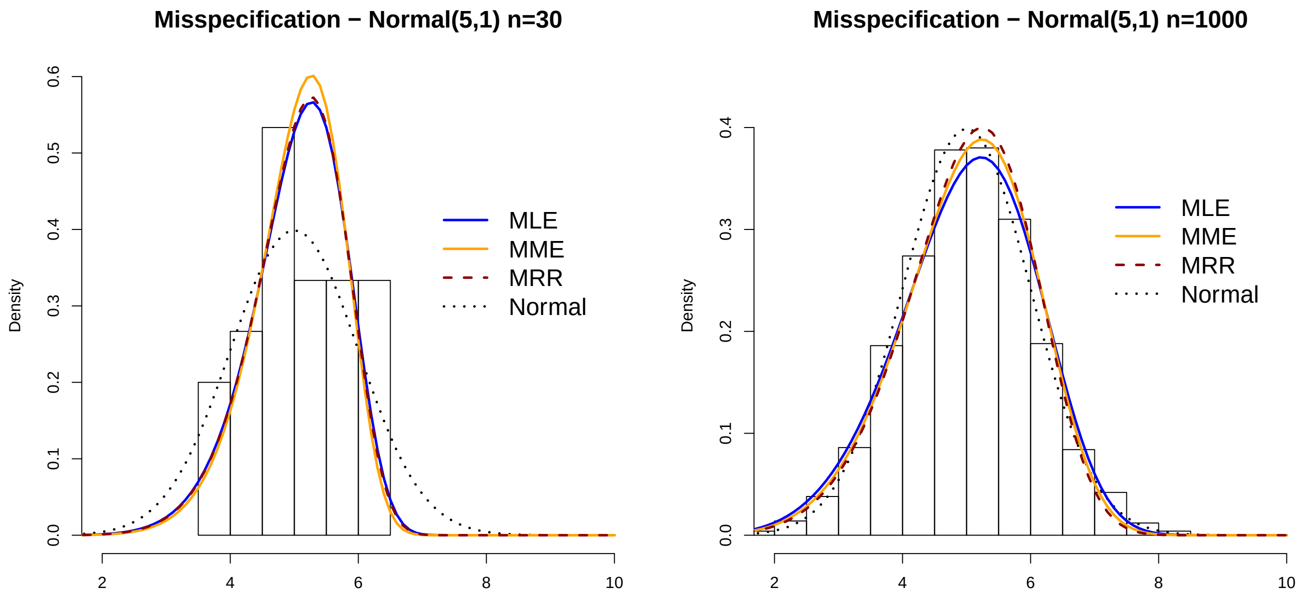


	$n = 30$	$n = 1000$
MLE	0.9739	0.2264
MME	0.9812	0.1800
MRR	0.9883	0.0634
Gamma	0.3442	0.9662

Table 3.7: p-values for the KS tests for the gamma misspecification. H_0: The data come from a Weibull distribution with corresponding parameter estimates.

Notice here that the MRR method performs best with small sample size and the MLE with large sample size. Because the p-values are not that different for the smaller sample data, I would conclude that the MLE is best for this misspecification.

Figure 3.6: Data misspecification for the normal.

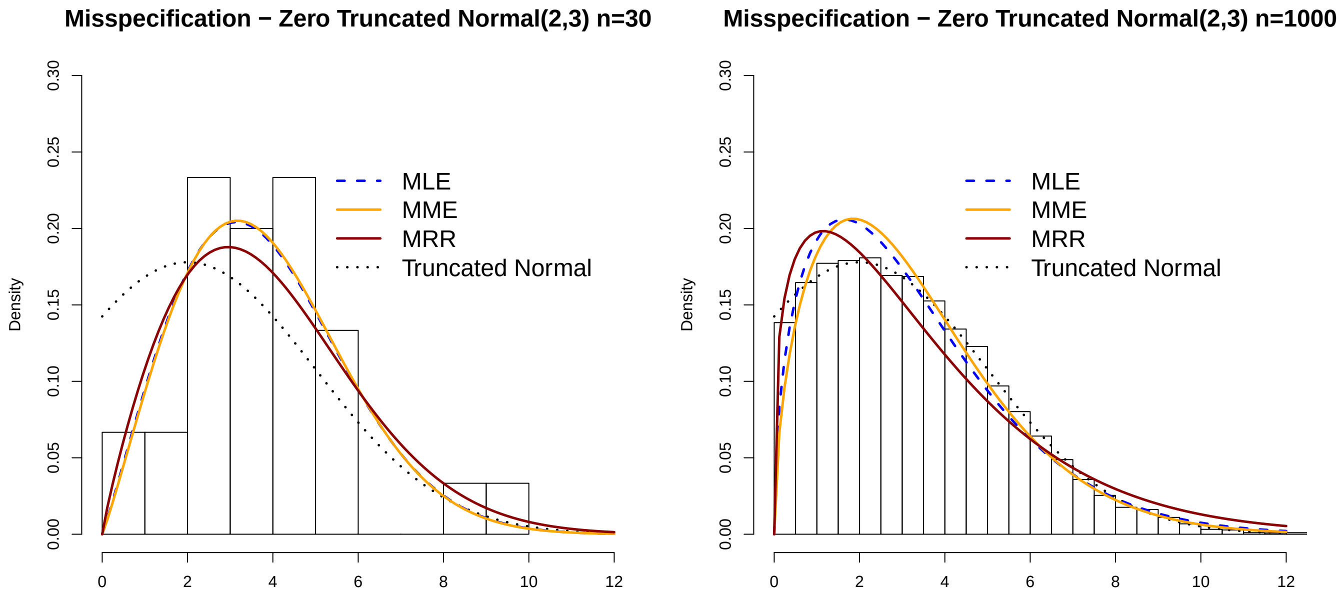


	$n = 30$	$n = 1000$
MLE	0.8484	0.2898
MME	0.7572	0.3330
MRR	0.8375	0.2402
Normal	0.7471	0.8634

Table 3.8: p-values for the KS tests for the normal misspecification. H_0: The data come from a Weibull distribution with corresponding parameter estimates.

Here we see that the best method of estimation for this type of misspecification is the MLE for small data sets and the MME for large sets.

Figure 3.7: Data misspecification for the zero-truncated normal.



Misspecification − Zero Truncated Normal(2,3) n=30

Misspecification − Zero Truncated Normal(2,3) n=1000

	$n = 30$	$n = 1000$
MLE	0.7675	1.3389E-10
MME	0.7924	6.6811E-07
MRR	0.6688	0.0000
TruncNorm	0.0371	0.4912

Table 3.9: p-values for the KS tests for the truncated normal misspecification. H_0: The data come from a Weibull distribution with corresponding parameter estimates.

In this case we notice that our KS tests were significant in all the large sample size Weibull estimations, although the MME is best. I conclude that this is trivial because larger sample sizes that come from a truncated normal are not very likely to be misspecified as Weibull data. It is in small sample sizes that I would show more concern. The MME parameter estimates have the highest p-value with the MLE parameter estimate's p-value slightly lower.

In each case, assuming the Weibull is the underlying distribution of each of these data sets is a reasonable misspecification. In general, the MLE estimates are the best fit for the misspecified data. This is a confirmation of the flexibility of the Weibull distribution as discussed in the introduction. While the fits above are not perfect, it is apparent that the Weibull is capable in parametrizing various types of data.

3.6 Summary

Overall, the maximum likelihood estimator was the best according to the mean square error. This is what we expected because of the asymptotic properties of the maximum likelihood estimator. Although the median rank regression method was worst according to the mean square error, because of its simplicity it is a very good estimator for finding the parameters. Both method of moments and maximum likelihood estimator required iterative algorithms, which was not necessary for the median rank regression method. Misspecification should raise no problems as seen in the previous section. The Weibull density's flexibility allows the modelling various types of data.

CHAPTER 4

DATA APPLICATION

4.1 Description of Data Set

In order to test our methods, I chose to use the T.W. Daniels Experimental Forest (TWDEF) dataset from the Department of Soil Physics at Utah State University (Utah State Universtity—Department of Soil Physics 2009–2010). The data contains the wind speed distributions for five seperate sites in the experimental forest. The set includes data from the following sites: aspen, conifer, grass, sage, and the summit. All non-missing data are the wind speeds, recorded every half hour at each site. Wind energy data are appropriately modelled using the Weibull distribution (Weisser 2003; Seguro and Lambert 2000; Celik 2003). The data set is large, and therefore we presume that the MLE method will likely be the most accurate as seen in our simulation studies.

4.2 Issues with the Wind Data

One of the main issues contained in this data is the fact that there are data equal to zero. Because the Weibull distribution is only for data that is greater than zero, this caused several problems for our parameter estimation, specifically for the MLE and MRR estimates. The solutions are as follows:

1. Add small values to each zero and find the estimates for the parameters using those values. In Figures 4.1, 4.2, and 4.3, these values are denoted in the key by “0.01, 0.0001, and 0.000001.”

2. The data are separated into bins (i.e. just like they are separated before creating a histogram) and the estimates are found for the binned data. For my data, I rounded

each value up to the nearest hundredth, including zero up to one one-hundredth. I found this to be a common practice among wind energy analysts.

3. Delete the data valued at zero and find the estimates for our wind data. This last solution is shown to be equivalent to the parameter estimation for a mixture model of the point density at zero and the Weibull for the MLEs.

4.3 Maximum Likelihood Estimation for a Mixture Model

Here we find the MLE for the mixture model for a point density at zero and a Weibull density. The zero point density is as follows: $f(x) = I(x = 0)$. Also, let n be the total number of data and let m be the number of non-zero data points. Note that x_j below is the j^{th} non-zero wind speed measure.

$$\begin{aligned} g(x|\gamma, \beta, \pi) &= f(x)(1-\pi) + Weibull(\gamma, \beta)\pi \\ \Rightarrow \prod_{i=1}^{n} [g(x_i|\gamma, \beta)] &= \prod_{i=1}^{n} [f(x_i)(1-\pi) + Weibull(x_i|\gamma, \beta)\pi] \end{aligned}$$

When $x_i = 0$, $g(x_i|\gamma, \beta, \pi) = (1-\pi)$ and when $x_i \neq 0$, then $g(x_i|\gamma, \beta, \pi) = Weibull(x_i|\gamma, \beta)\pi$. Therefore we can write the equation as:

$$\prod_{i=1}^{n} [g(x_i|\gamma, \beta)] = (1-\pi)^{(n-m)}\pi^m \prod_{j=1}^{m} Weibull(x_j|\gamma, \beta)$$

Once this has been done, the MLE can be found for π by maximizing the log likelihood.

$$\begin{aligned}
\ell(\mathbf{x}|\gamma,\beta,\pi) &= \log\{(1-\pi)^{(n-m)}\pi^m \prod_{j=1}^{m} Weibull(x_j|\gamma,\beta)\} \\
&= (n-m)\log(1-\pi) + m\log\pi + \sum_{j=1}^{m} \log Weibull(x_j|\gamma,\beta) \\
\Rightarrow \frac{d\ell}{d\pi} &= \frac{(n-m)}{(1-\pi)}(-1) + \frac{m}{\pi} \\
\Rightarrow 0 &= \frac{(m-n)}{(1-\hat{\pi})} + \frac{m}{\hat{\pi}} \\
\Rightarrow 0 &= m\hat{\pi} - n\hat{\pi} + m - m\hat{\pi} \\
\Rightarrow 0 &= -n\hat{\pi} + m \Rightarrow \hat{\pi} = \frac{m}{n}
\end{aligned}$$

Notice that the same can be done with the parameters γ, and β. As you can see from $\ell(\mathbf{x}|\gamma,\beta,\pi)$, the MLEs for the remaining estimates will be the same as the MLE for the non-zero data. Therefore, the MLEs for the mixture model are the same as the estimates that are found in my method where we "delete" the zeros from the data.

4.4 Application Results

In the analyses of these data, we find that all three methods predict the following parameter estimates assuming the Weibull distribution.

Figure 4.1: Here is a histogram of the Grass Site data, with overlaid Weibull curves for the parameters chosen by our Maximum Likelihood Methods.

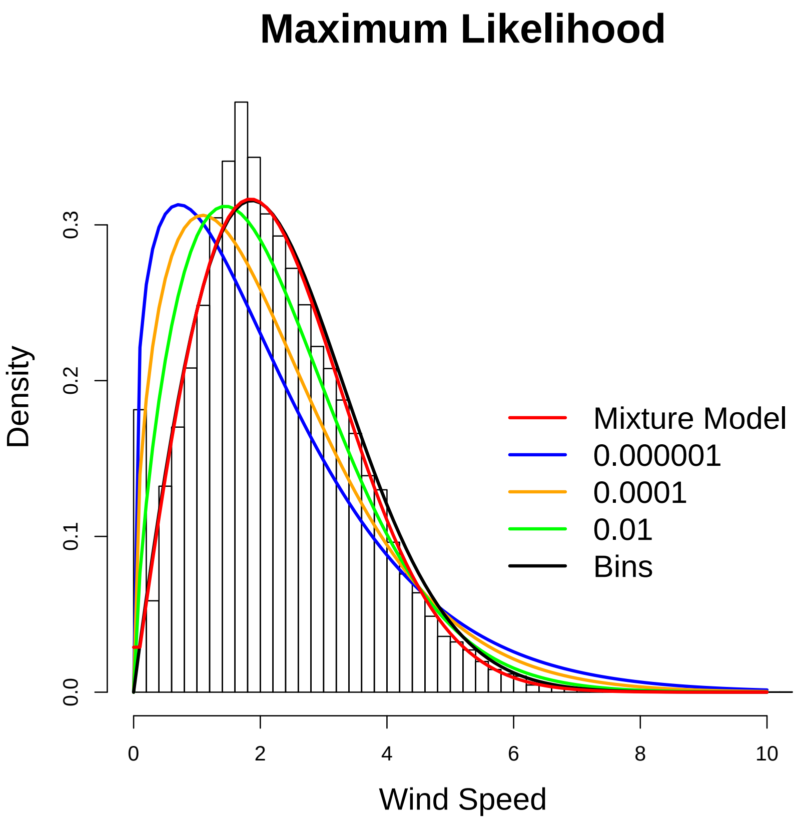


Table 4.1: MLEs for grass site data.

MLE method	γ	β
Mixture	1.99034	2.62306
Deleted	1.99034	2.62306
Binned	1.96030	2.68286
+ 0.000001	1.27802	2.35741
+ 0.0001	1.45597	2.41718
+ 0.01	1.67482	2.48765

Figure 4.2: Here is a histogram of the Grass Site data, with overlaid Weibull curves for the parameters chosen by the Method of Moments.

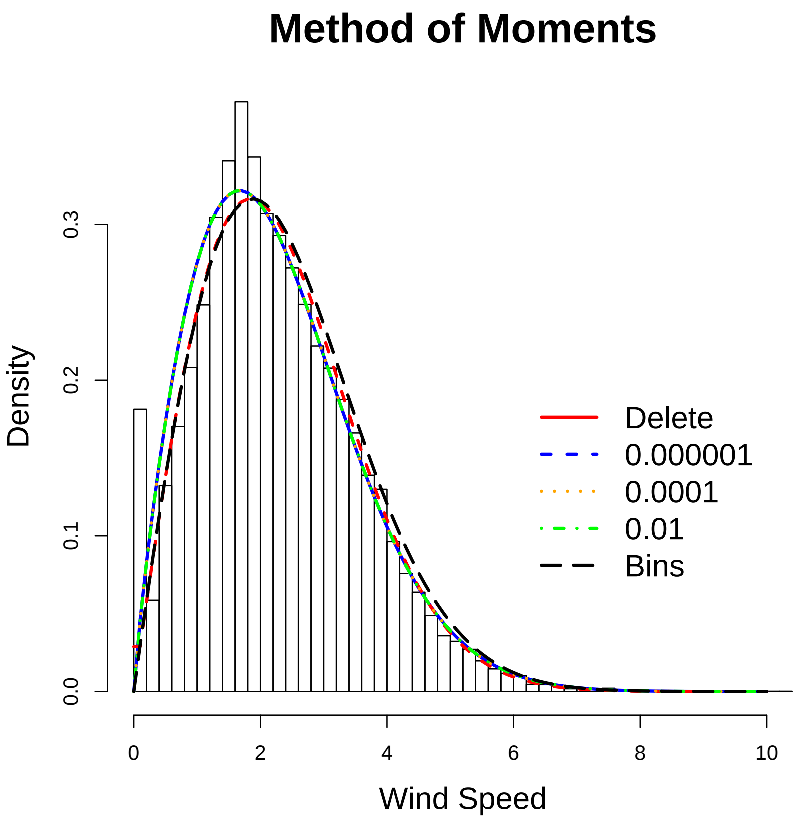


Table 4.2: MMEs for grass site data.

MME method	γ	β
Deleted	1.98974	2.62295
Binned	1.97344	2.68672
+ 0.000001	1.85277	2.54206
+ 0.0001	1.85279	2.54206
+ 0.01	1.85387	2.54244

Figure 4.3: Here is a histogram of the Grass Site data, with overlaid Weibull curves for the parameters chosen by our Median Rank Regression Method.

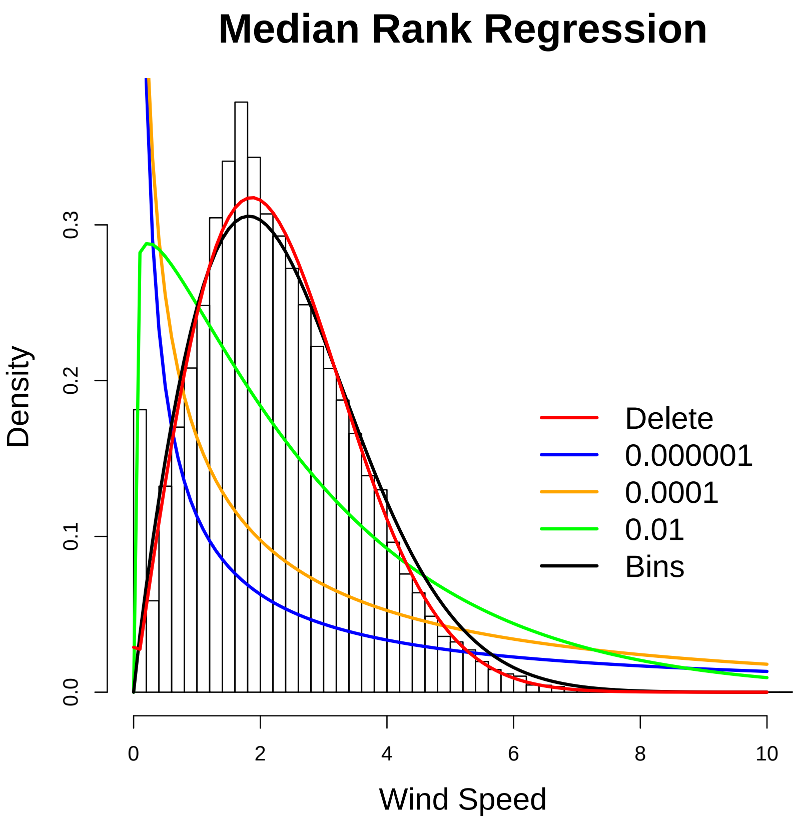


Table 4.3: MRR estimates for grass site data.

MRR method	γ	β
Deleted	2.00652	2.62908
Binned	1.88911	2.70797
+ 0.000001	0.36846	6.21900
+ 0.0001	0.57041	4.07870
+ 0.01	1.07205	2.90100

We can also see that all methods are similar enough to show that each method would be sufficient in determining our parameter estimates. Table 4.4 emphasizes this point because the estimated parameter estimates are within 2 one-hundredths of each other. Graphs with overlaid Weibull curves and tables of parameter estimates for the remaining sites can be found in appendix A.

4.5 Summary

If we look at the resulting parameter estimates from each data set, we can see that the methods in which we replaced the zeros with small numbers gave too much weight to the small values in this data set. This was the case in both sets of the MLE and MRR estimates. The method of moments did not seem to change significantly between each of the methods. Also, looking at the binned data, I noticed that the density of the overlaid curve is slightly higher than the actual data because I am rounding the data up. Finally, for the MLE our mixture model proved to produce the exact estimates as our deleted data method, and the estimate for π was the ratio of the number of non-zero data points to the total number of data points. The values obtained by *nlm* were off by a few ten-millionths. If we look at what appear graphically to be the best estimates for each method, we see that they are all almost the same. This is shown in the table below. Note that the estimate for π in the MLE is compared to the ratio of the number of non-zero data points to the total number of data points in both the MOM and MRR methods.

Table 4.4: The parameter estimates for the MLE mixture and the MME and MRR deleted methods. The deleted methods mimic the mixture model. Notice how close these estimates are to each other.

	γ	β	π
MLE mixture	1.99034	2.62306	0.97118
MME deleted	1.98974	2.62295	0.97118
MRR deleted	2.00652	2.62908	0.97118

A graphical comparison showing each method would not tell us much because of how close the estimates are to each other; the three lines would overlay each other. From this we can see that no matter what the true parameters are, the estimates for each method are all going to have approximately the same accuracy. This is due to the fact that our data set is very large.

BIBLIOGRAPHY

Casella, G., and Berger, R. L. (2002), *Statistical Infrerence* (2nd ed.), Duxbury.

Celik, A. N. (2003), "A Statistical Analysis of Wind Power Density Based on the Weibull and Rayleigh Models at the Southern Region of Turkey," *Renewable Energy*, 29, 593–604, (http://www.elsvier.com/locate/renene).

Coles, S. (2001), *An Introduction to Statistical Modeling of Extreme Values*, Springer - Verlag London Limited.

Collett, D. (2009), *Modelling Survival Data in Medical Research* (2nd ed.), Chapman and Hall/CRC.

Corotis, R. B. (1979), "Statistical Models for Wind Characteristics at Potential Wind Energy Conversion Sites," *United States Department of Energy*, (Work performed under contract No. EY-76-S-06-2342).

Hennessey, Jr., J. P. (1977), "Some Aspects of Wind Power Statistics," *Journal of Applied Meteorology*, 16.

Jones, O., Maillardet, R., and Robinson, A. (2009), *Introduction to Scientific Programming and Simulation Using R*, Chapman and Hall/CRC.

Murthy, D. P., Xie, M., and Jiang, R. (2004), *Weibull Models*, John Wiley and Sons, Inc.

ReliaSoft Corportation (1996–2006), "Estimation of Weibull Parameters," (http://www.weibull.com/LifeDataWeb/).

Seguro, J., and Lambert, T. (2000), "Modern estimation of the parameters of the Weibull wind speed distribution for wind energy analysis," *Journal of Wind Engineering and Industrial Aerodynamics*, 85, 75–84.

Utah State Universtity—Department of Soil Physics (2009–2010), "TWDEF Site Data," (http://twdef.usu.edu/view.html).

Weisser, D. (2003), "A Wind Energy Analysis of Grenada: an Estimation Using the 'Weibull' Density Function," *Renewable Energy*, 28, 1803–1812, (http://www.elsvier.com/locate/renene).

Wu, S.-J. (2002), "Estimations of the Parameters of the Weibull Distribution with Progressively Censored Data," *Journal of The Japan Statistical Society*, 32, 155–163.

Zhang, L., Xie, M., and Tang, L. (2007), "A study of two estimation approaches for parameters of Weibull distribution based on WPP," *Reliability Engineering and System Safety*, 92, 360–368.

APPENDICES

APPENDIX A

ADDITIONAL GRAPHS

A.1 Graphs of Normality

Maximum Likelihood Method

Figure A.1: **Normally Distributed Parameter Estimates:** Both are distributions of the parameters for the maximum likelihood method, with $\gamma = 5$, $\beta = 90$, and $n = 20$.

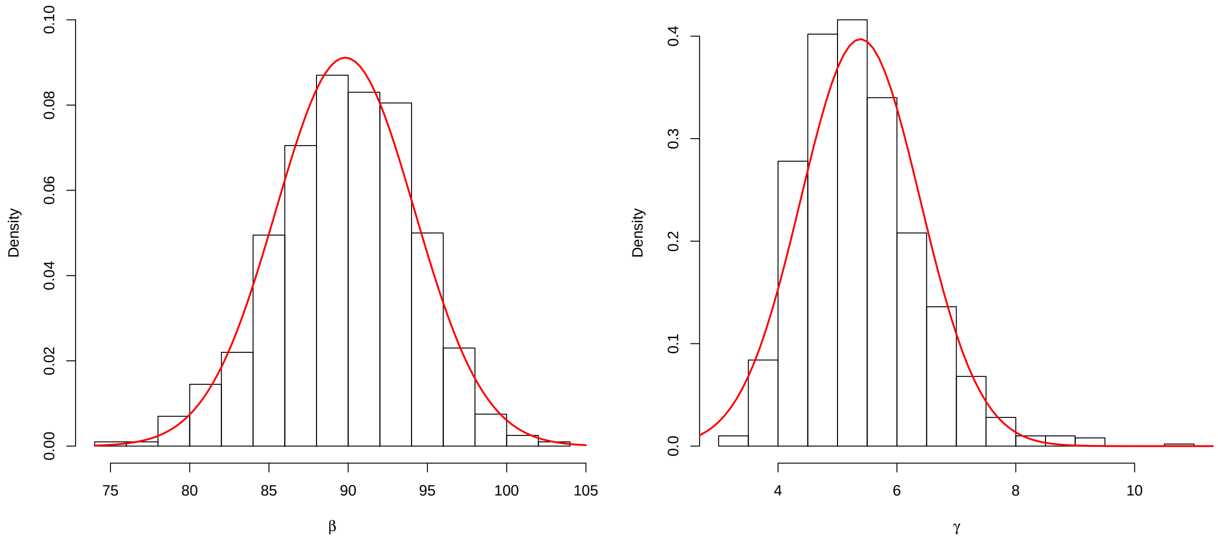


Figure A.2: **Normally Distributed Parameter Estimates:** Both are distributions of the parameters for the maximum likelihood method, with $\gamma = 5$, $\beta = 90$, and $n = 100$.

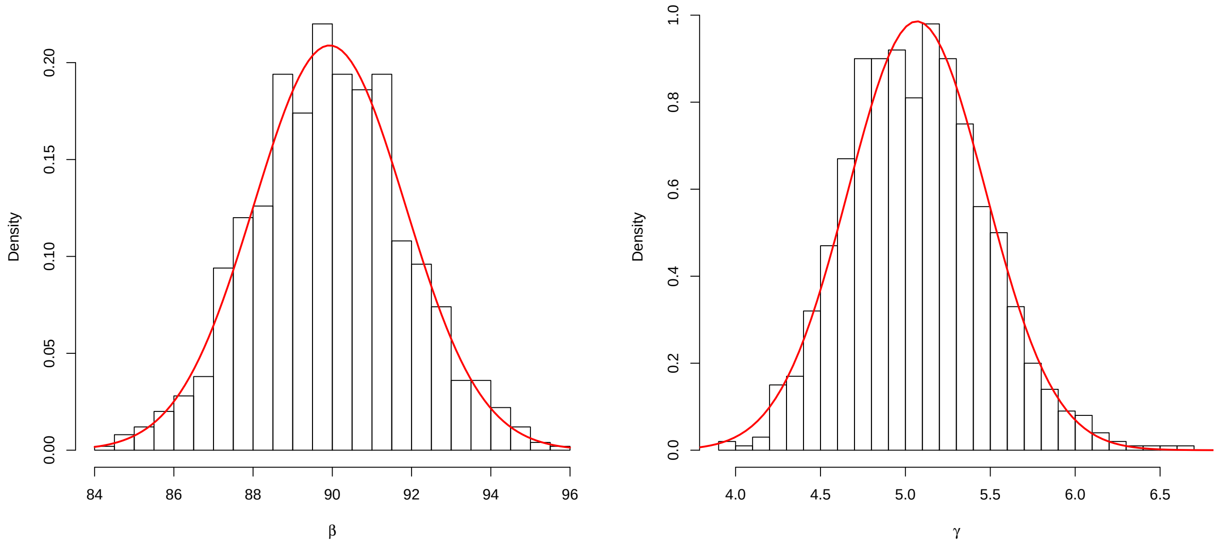


Figure A.3: **Normally Distributed Parameter Estimates:** Both are distributions of the parameters for the maximum likelihood method, with $\gamma = 5$, $\beta = 90$, and $n = 10000$.

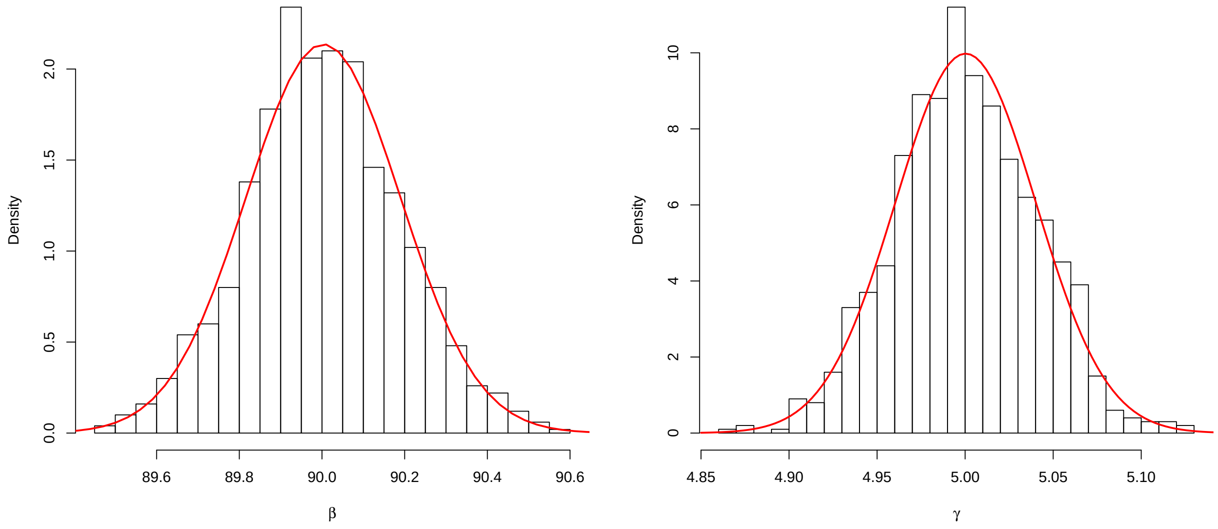


Figure A.4: **Normally Distributed Parameter Estimates:** Both are distributions of the parameters for the maximum likelihood method, with $\gamma = 2$, $\beta = 2.5$, and $n = 20$.

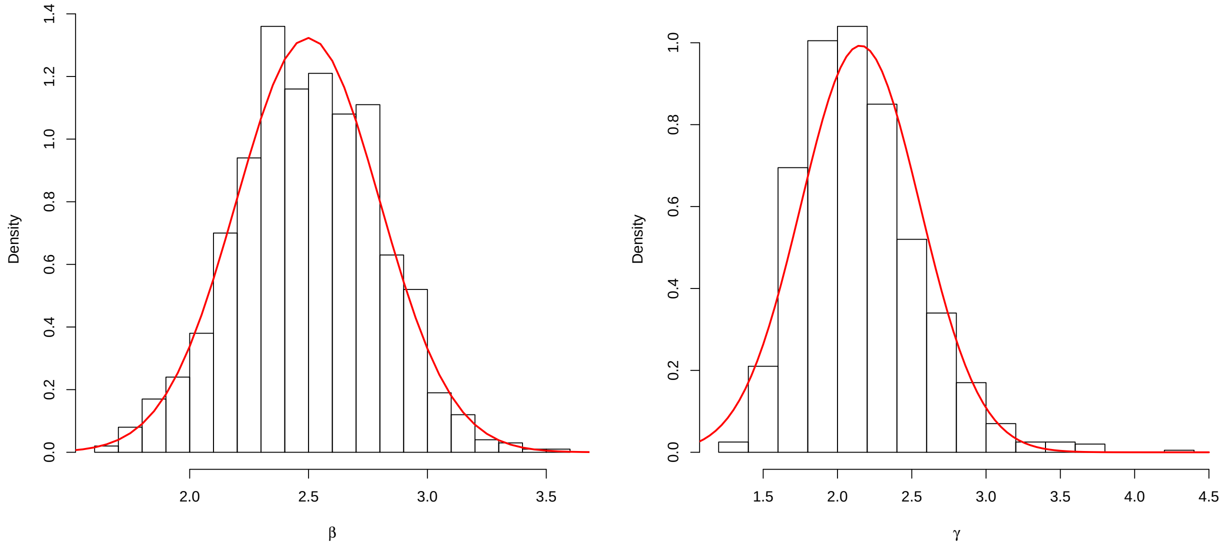


Figure A.5: **Normally Distributed Parameter Estimates:** Both are distributions of the parameters for the maximum likelihood method, with $\gamma = 2$, $\beta = 2.5$, and $n = 100$.

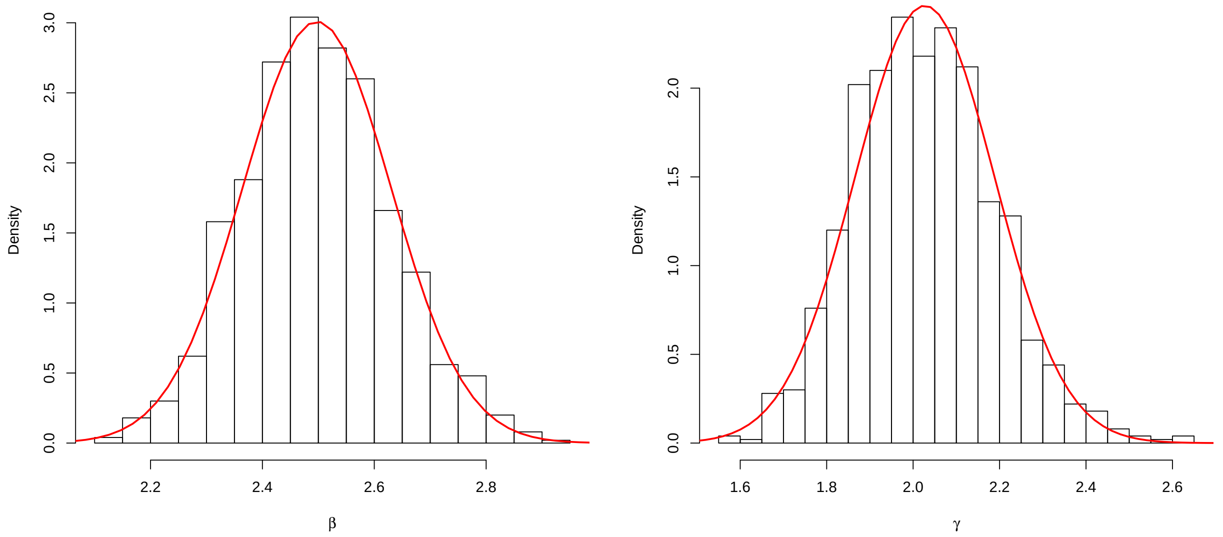
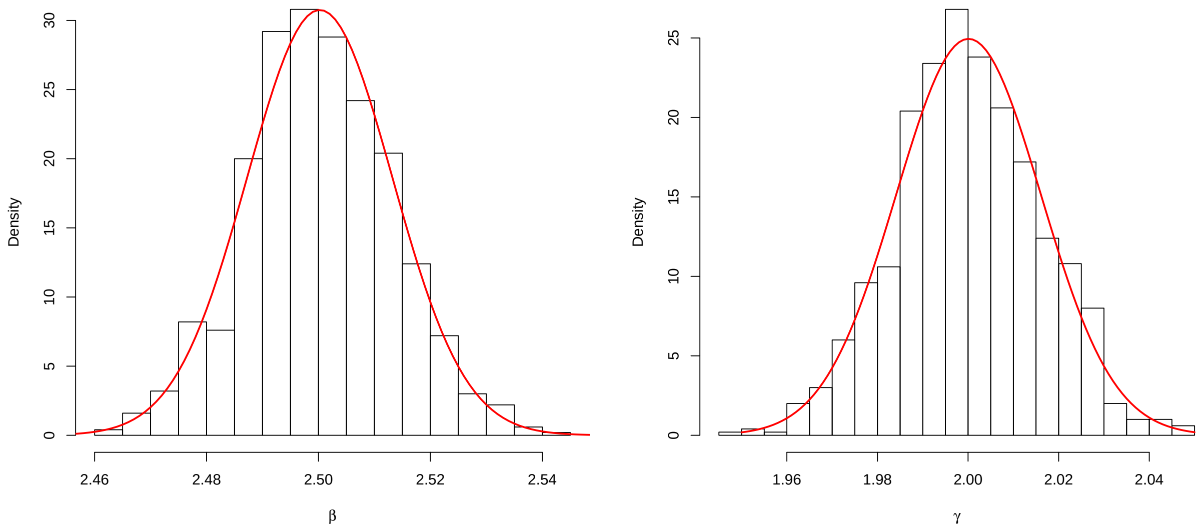


Figure A.6: **Normally Distributed Parameter Estimates:** Both are distributions of the parameters for the maximum likelihood method, with $\gamma = 2$, $\beta = 2.5$, and $n = 10000$.



Method of Moments

Figure A.7: **Normally Distributed Parameter Estimates:** Both are distributions of the parameters for the method of moments, with $\gamma = 5$, $\beta = 90$, and $n = 20$.

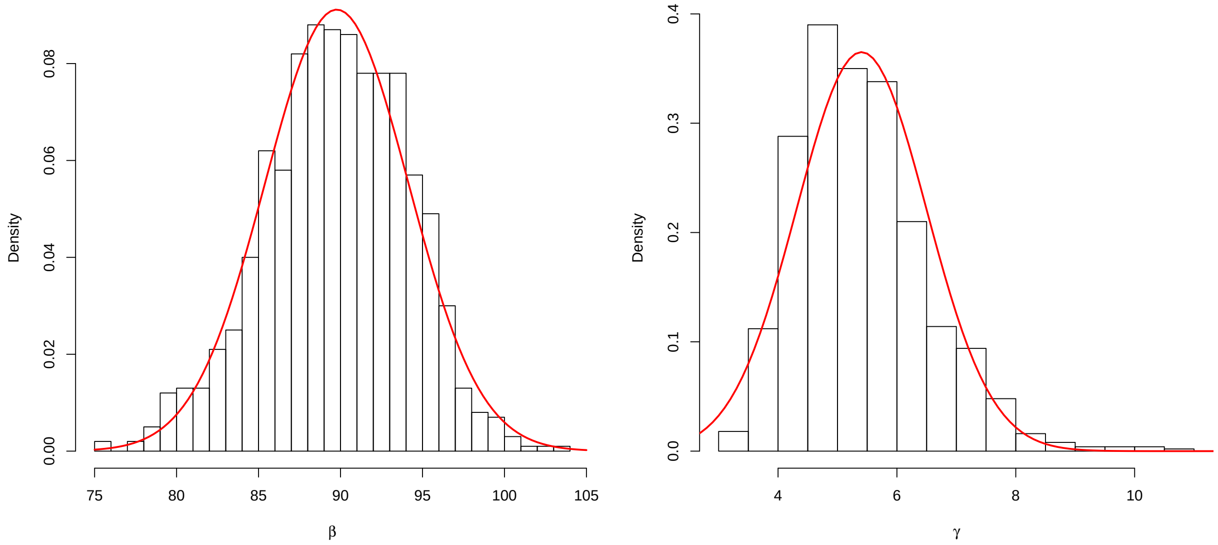


Figure A.8: **Normally Distributed Parameter Estimates:** Both are distributions of the parameters for the method of moments, with $\gamma = 5$, $\beta = 90$, and $n = 100$.

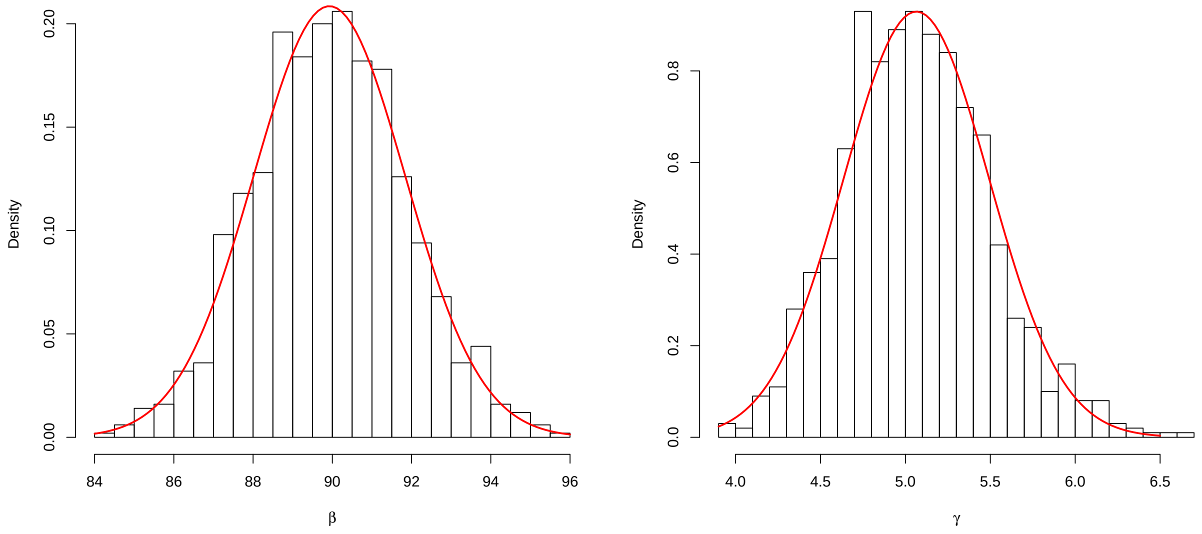


Figure A.9: **Normally Distributed Parameter Estimates:** Both are distributions of the parameters for the method of moments, with $\gamma = 5$, $\beta = 90$, and $n = 10000$.

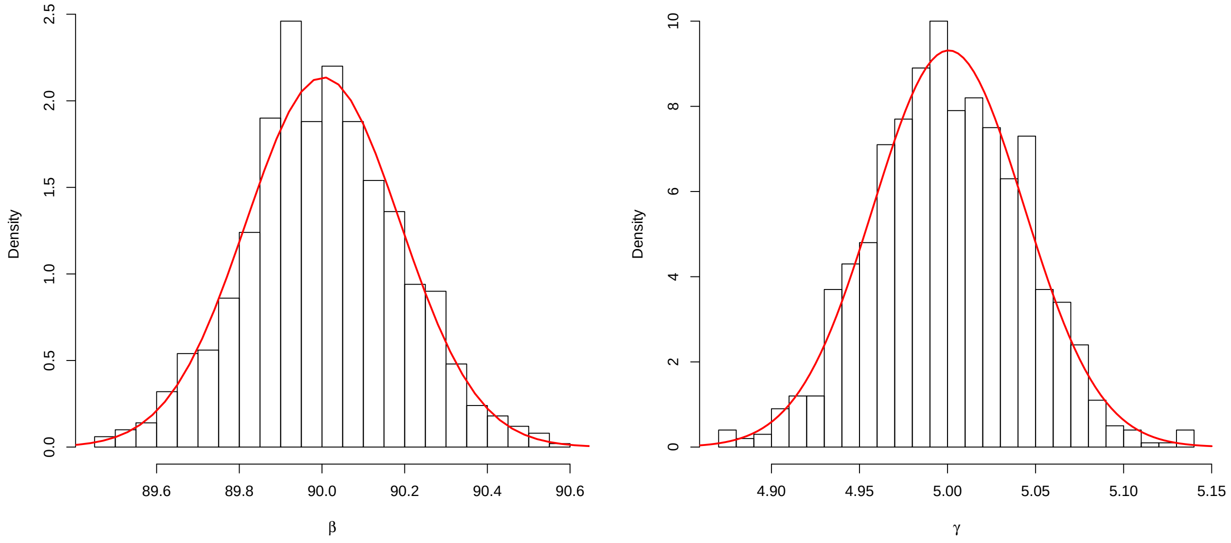


Figure A.10: **Normally Distributed Parameter Estimates:** Both are distributions of the parameters for the method of moments, with $\gamma = 2$, $\beta = 2.5$, and $n = 20$.

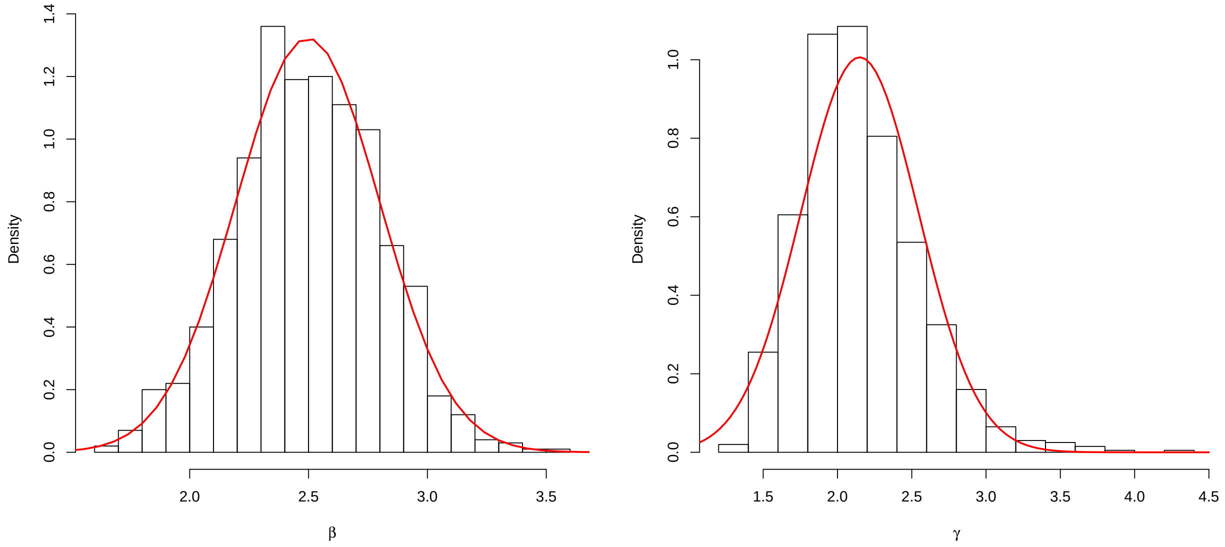


Figure A.11: **Normally Distributed Parameter Estimates:** Both are distributions of the parameters for the method of moments, with $\gamma = 2$, $\beta = 2.5$, and $n = 100$.

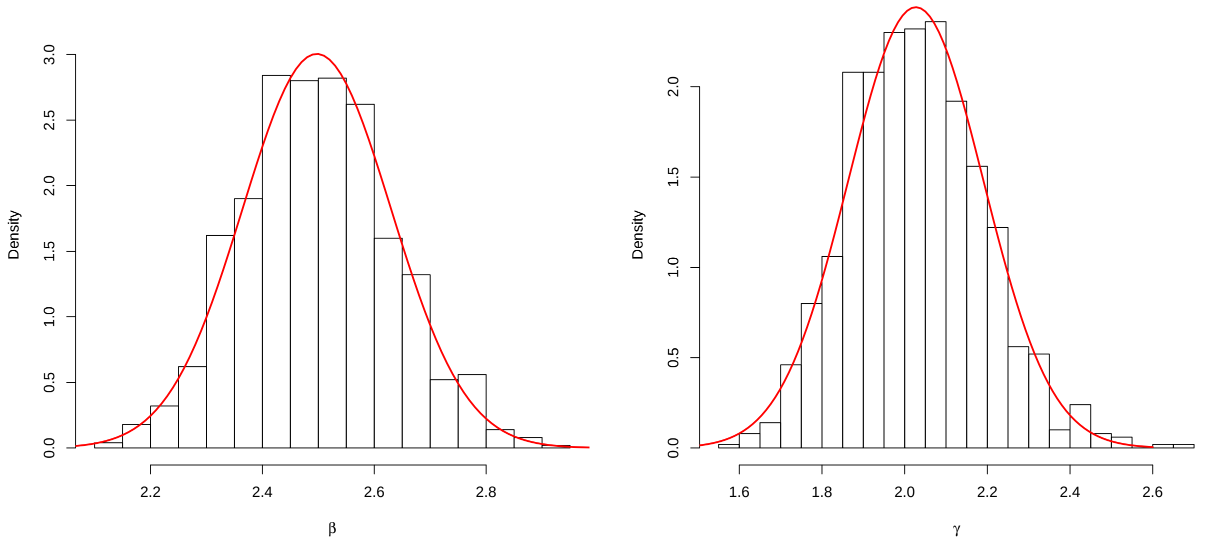
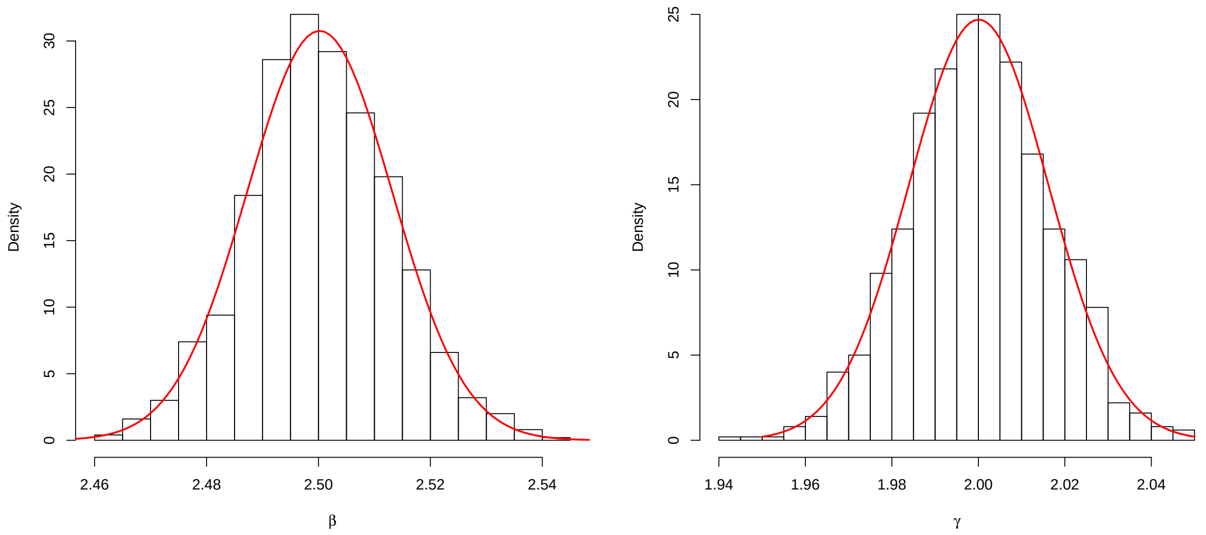


Figure A.12: **Normally Distributed Parameter Estimates:** Both are distributions of the parameters for the method of moments, with $\gamma = 2$, $\beta = 2.5$, and $n = 10000$.



Median Rank Regression

Figure A.13: **Normally Distributed Parameter Estimates:** Both are distributions of the parameters for the median rank regression method, with $\gamma = 5$, $\beta = 90$, and $n = 20$.

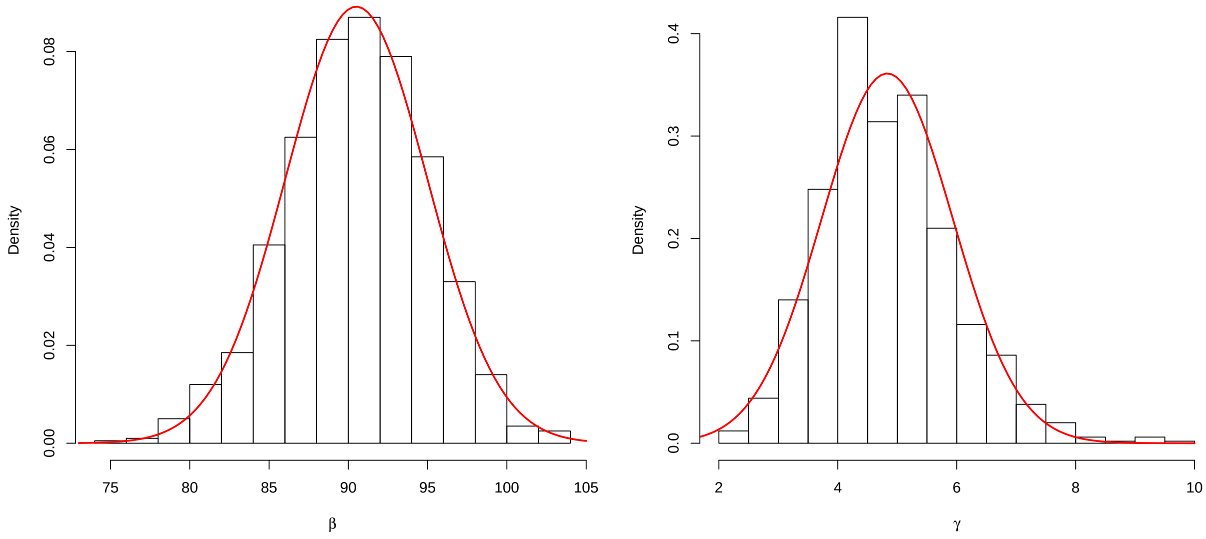


Figure A.14: **Normally Distributed Parameter Estimates:** Both are distributions of the parameters for the median rank regression method, with $\gamma = 5$, $\beta = 90$, and $n = 100$.

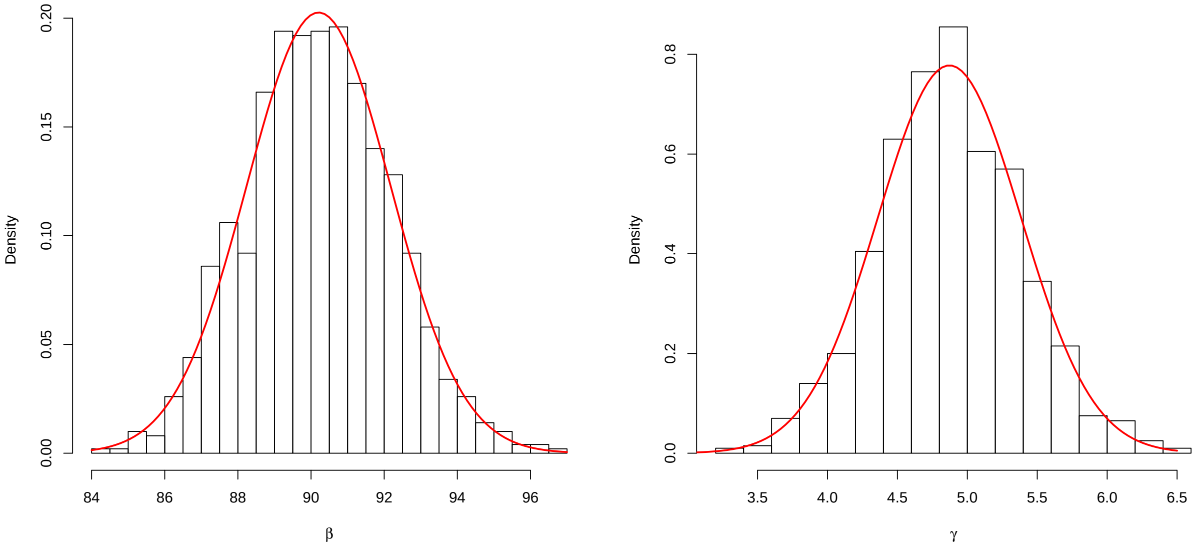


Figure A.15: **Normally Distributed Parameter Estimates:** Both are distributions of the parameters for the median rank regression method, with $\gamma = 5$, $\beta = 90$, and $n = 10000$.

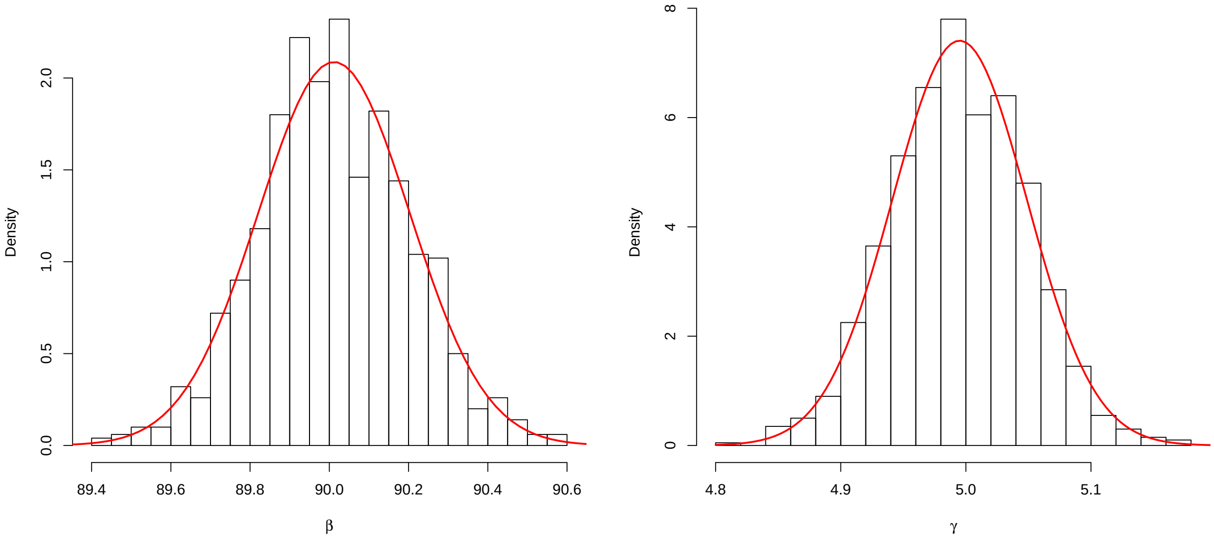


Figure A.16: **Normally Distributed Parameter Estimates:** Both are distributions of the parameters for the median rank regression method, with $\gamma = 2$, $\beta = 2.5$, and $n = 20$.

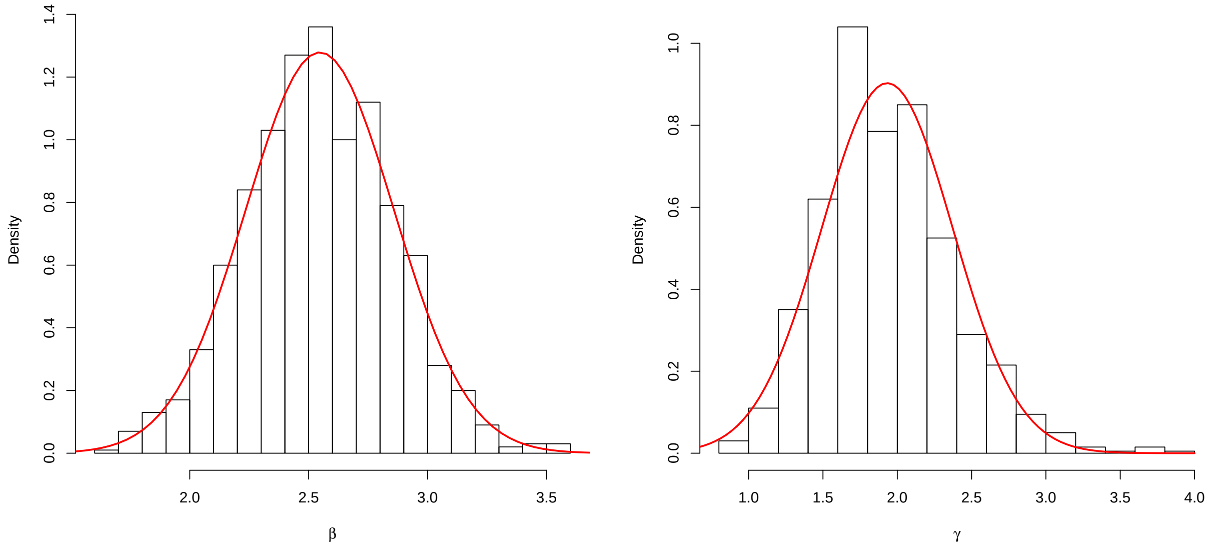


Figure A.17: **Normally Distributed Parameter Estimates:** Both are distributions of the parameters for the median rank regression method, with $\gamma = 2$, $\beta = 2.5$, and $n = 100$.

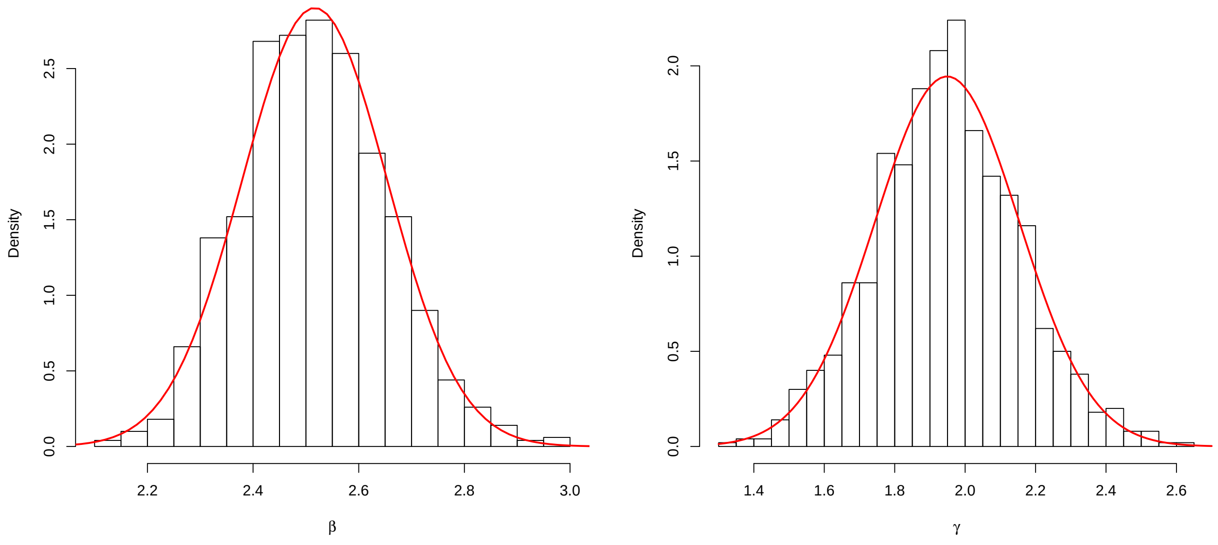
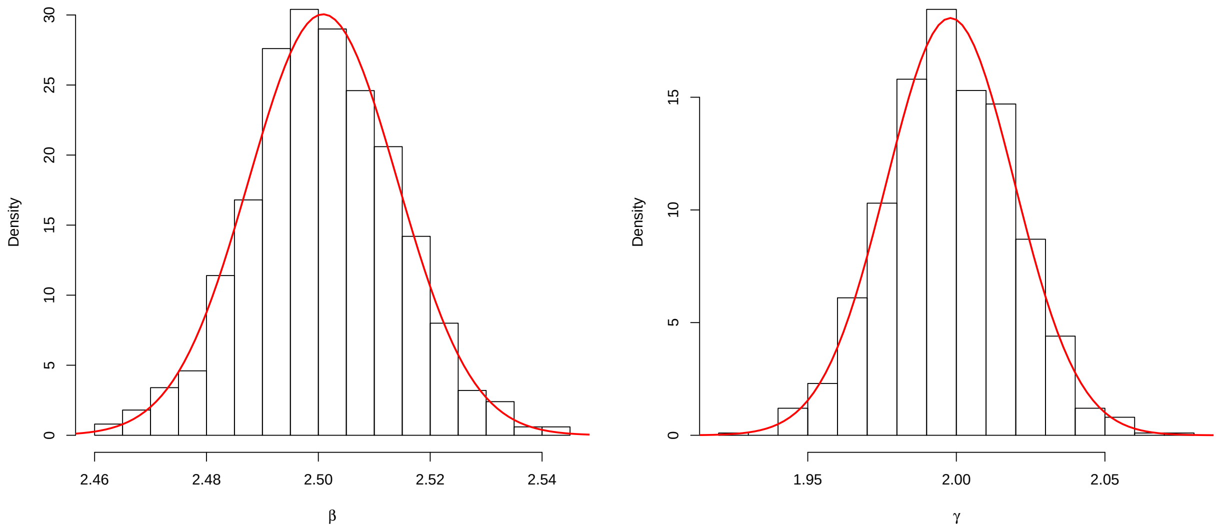


Figure A.18: **Normally Distributed Parameter Estimates:** Both are distributions of the parameters for the median rank regression method, with $\gamma = 2$, $\beta = 2.5$, and $n = 10000$.



MLE method	γ	β
Mixture	1.98968	1.31464
Deleted	1.98968	1.31464
Binned	1.94784	1.36516
Add 0.000001	0.77568	1.02308
Add 0.0001	1.00373	1.08067
Add 0.01	1.37274	1.15694

Table A.1: MLEs for the aspen site.

MME method	γ	β
Deleted	2.01276	1.31787
Binned	1.94667	1.36504
+ 0.000001	1.68236	1.20933
+ 0.0001	1.68241	1.20934
+ 0.01	1.68710	1.21042

Table A.2: MMEs for the aspen site.

MRR method	γ	β
Deleted	1.90188	1.33728
Binned	1.90098	1.37734
+ 0.000001	0.26718	3.02624
+ 0.0001	0.41547	1.98037
+ 0.01	0.84196	1.38628

Table A.3: MRR estimates for the aspen site.

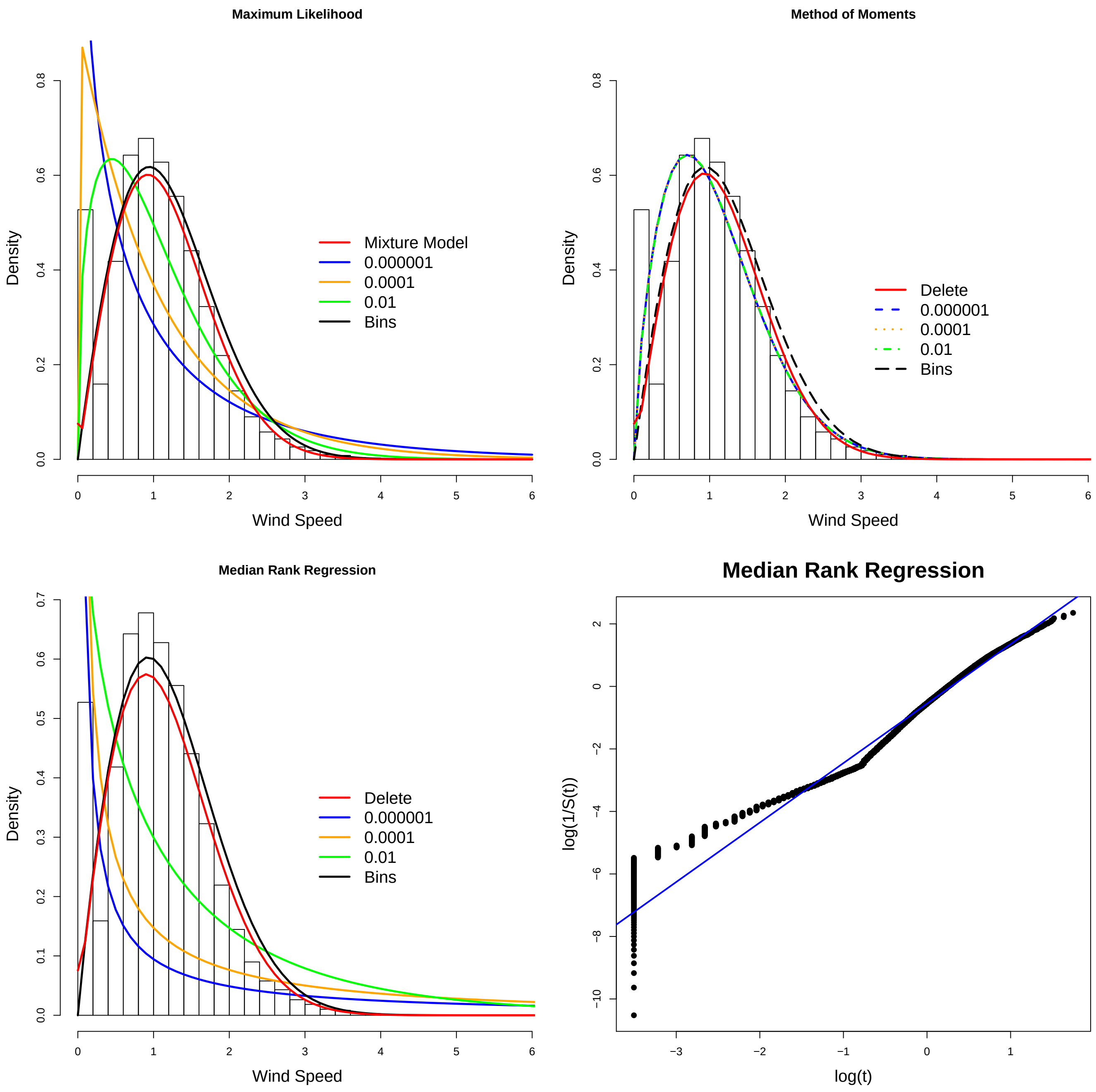


Figure A.19: Graphical summary of the parameter estimates for the aspen site. The method of moments appears to be most "robust" in estimation for all methods. In all methods, the deleted/mixture and binned data parameter estimates are most accurate.

MLE method	γ	β
Mixture	1.91104	0.71525
Deleted	1.91104	0.71525
Binned	1.08684	0.50213
+ 0.000001	0.25449	0.14215
+ 0.0001	0.37822	0.22965
+ 0.01	0.71221	0.38087

Table A.4: MLEs for the conifer site.

MME method	γ	β
Deleted	2.03695	0.72417
Binned	1.24820	0.52303
+ 0.000001	1.11050	0.46508
+ 0.0001	1.11067	0.46513
+ 0.01	1.12787	0.47051

Table A.5: MMEs for the conifer site.

MRR method	γ	β
Deleted	1.47805	0.75570
Binned	0.94479	0.52934
+ 0.000001	0.17376	0.26494
+ 0.0001	0.26972	0.32761
+ 0.01	0.57137	0.42676

Table A.6: MRR estimates for the conifer site.

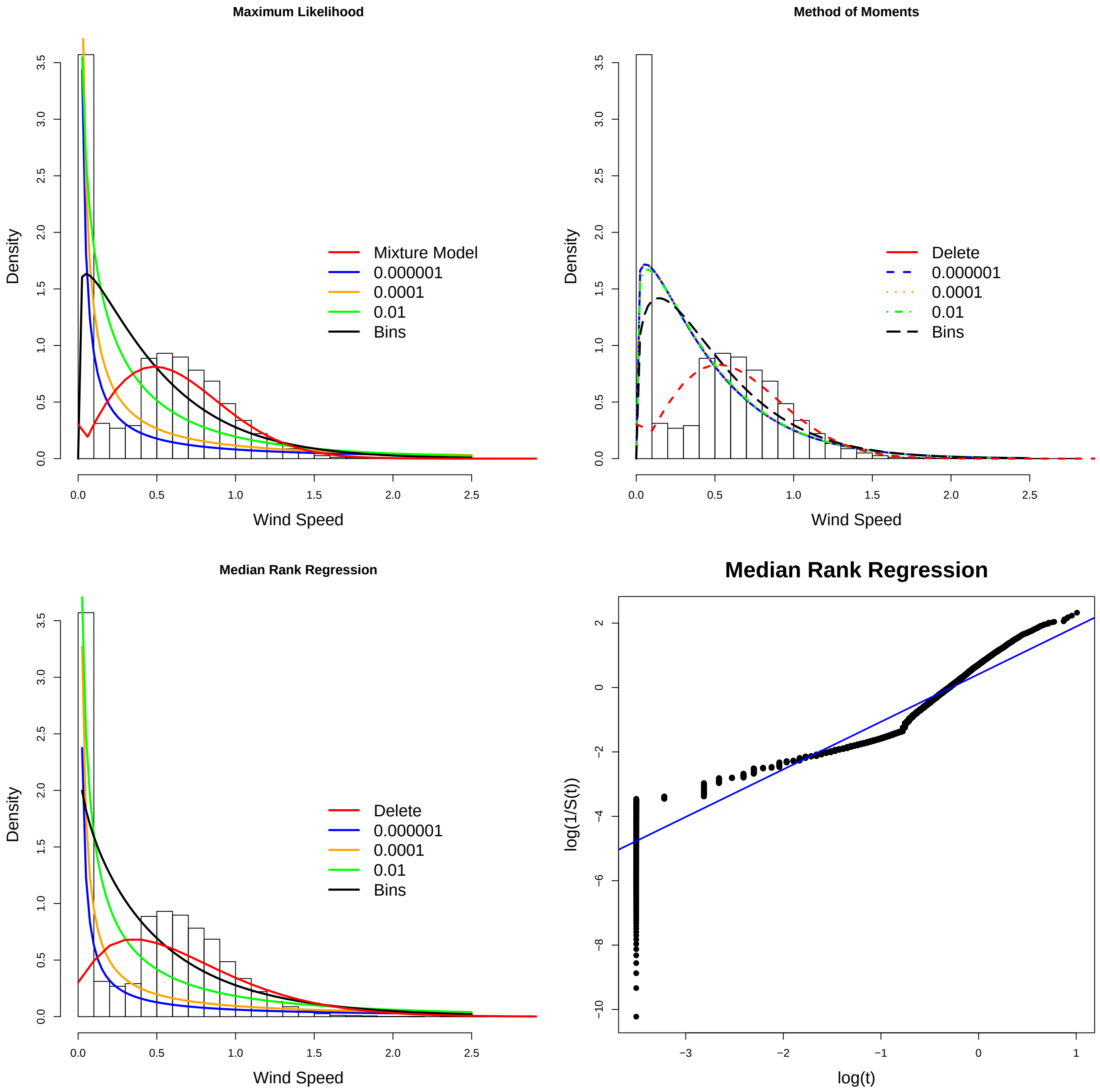


Figure A.20: Graphical summary of the parameter estimates for the conifer site. Notice the high proportion of values close to zero.

MLE method	γ	β
Mixture	1.80668	4.81714
Deleted	1.80668	4.81714
Binned	1.86742	4.98842
+ 0.000001	1.66072	4.70657
+ 0.0001	1.70347	4.73489
+ 0.01	1.74780	4.76414

Table A.7: MLEs for the summit.

MME method	γ	β
Deleted	1.78376	4.80543
Binned	1.84069	4.97420
+ 0.000001	1.76297	4.77663
+ 0.0001	1.76298	4.77663
+ 0.01	1.76307	4.77671

Table A.8: MMEs for the summit.

MRR method	γ	β
Deleted	1.87337	4.80686
Binned	1.98354	4.95250
+ 0.000001	0.75611	6.98805
+ 0.0001	1.07748	5.70484
+ 0.01	1.53393	4.98629

Table A.9: MRR estimates for the summit.

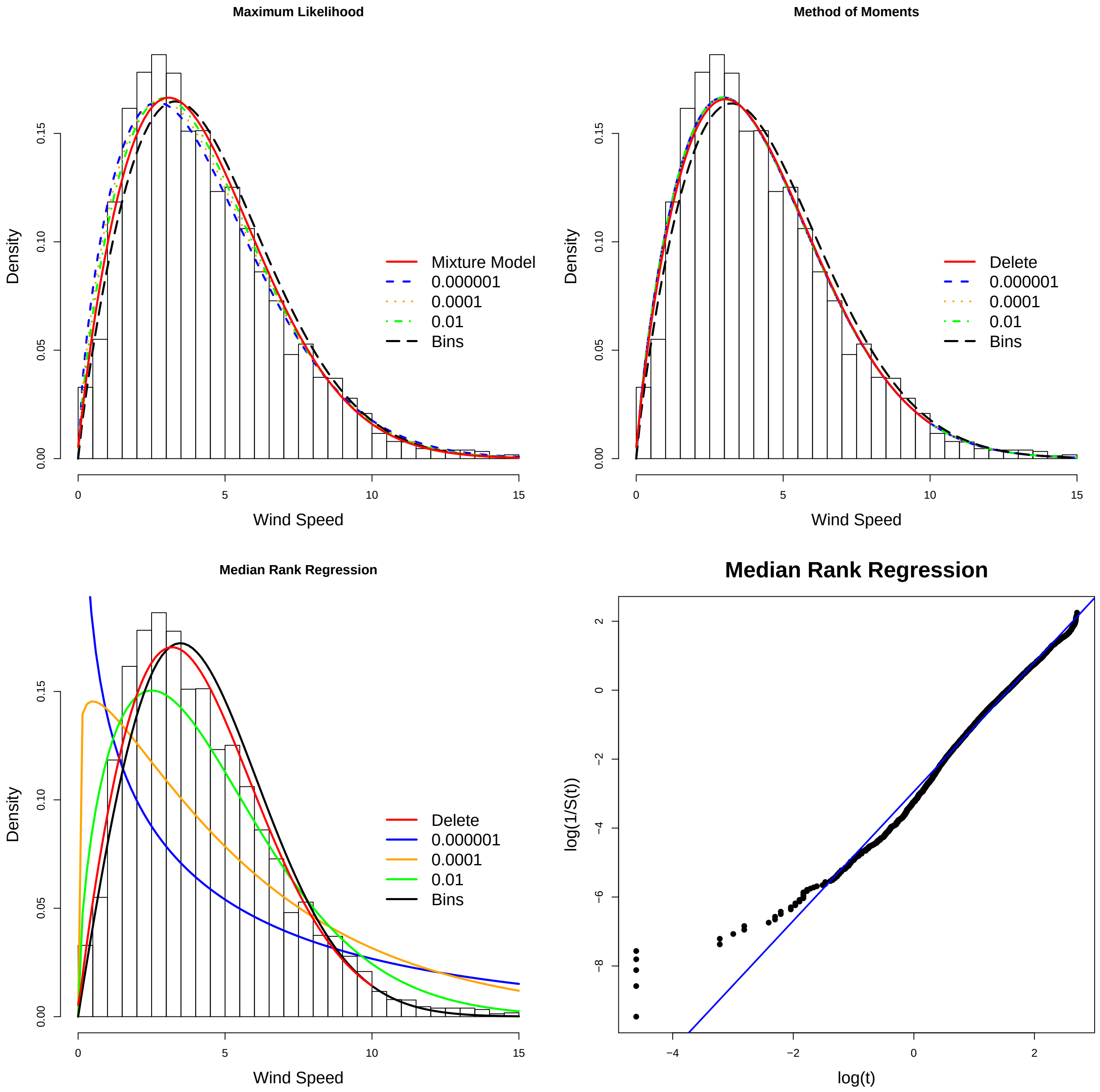


Figure A.21: Graphical summary for the parameter estimates for the summit site. There is evidence here that the MRR method is most sensitive to the zero data.

MLE method	γ	β
Mixture	1.44645	2.32531
Deleted	1.44645	2.32531
Binned	1.49738	2.40691
+ 0.000001	1.00917	2.05087
+ 0.0001	1.12541	2.11278
+ 0.01	1.26391	2.18411

Table A.10: MLEs for the sage site.

MME method	γ	β
Deleted	1.45800	2.33109
Binned	1.48168	2.40055
+ 0.000001	1.38578	2.24106
+ 0.0001	1.38579	2.24107
+ 0.01	1.38644	2.24158

Table A.11: MMEs for the summit.

MRR method	γ	β
Deleted	1.39711	2.34865
Binned	1.51890	2.40479
+ 0.000001	0.38197	4.50466
+ 0.0001	0.57915	3.11101
+ 0.01	0.97749	2.39434

Table A.12: MRR estimates for the sage site.

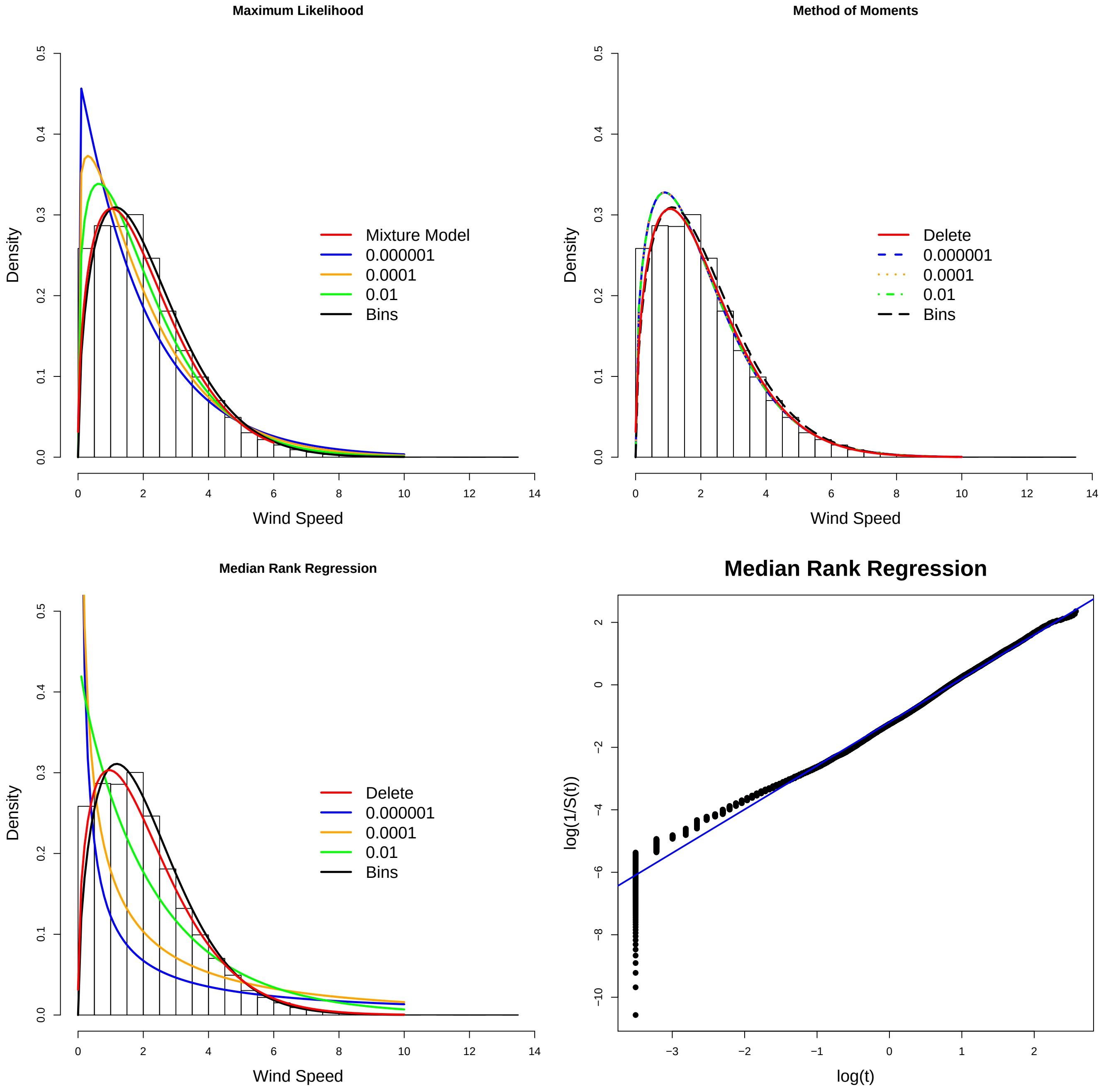


Figure A.22: Graphical summary of the parameter estimates for the sage site. Once again, the MME proves to be most robust when comparing each solution to remove the zeros from our data.

APPENDIX B

C-CODE

B.1 Code for Simulation Studies

```
/*******************
Mark Nielsen
C Project
(Adapted from code by Dr. David Engler)
********************/


#include <stdio.h>
#include <gsl/gsl_rng.h>          // required for random number generation
#include <gsl/gsl_randist.h>      // required for data generation from distributions
#include <gsl/gsl_multimin.h>     // required for use of multimin (optimization)
//#include <gsl/gsl_cdf.h>          // required for CDF values
//#include <gsl/gsl_statistics.h>
//#include <gsl/gsl_sort.h>
#include <gsl/gsl_sf_gamma.h>

#include <math.h>

# define npop 10000    // number of data points
# define nparam 2      // number of parameters to be estimated in function
# define nsim 1000        // number of simulations

gsl_rng * r;  /* global generator */

//This is my Weibull function
double weibull_pdf(double dat,double theta[nparam]) {
  double weibull;
  weibull=theta[0]/(pow(theta[1],theta[0]))*pow(dat,(theta[0]-1))*exp(-pow(dat/theta[1],theta[0]));
  return(weibull);
}

```

```

// ##### DEFINE FUNCTION TO BE MINIMIZED
// params = data
// v = vector of parameters to be identified
double weibull_ll (const gsl_vector *v, void *params)
{
  double *p = (double *)params; // convert data into readable format
  double weibll_val = 0;        // initialize value of neg loglikelihood
  double gamma, beta;
  gamma = gsl_vector_get(v, 0);    // assign values of param vector to specific
      variables
  beta = gsl_vector_get(v, 1);
  if(gamma<=0){
    return(1e10);
  }else if(beta<=0){
    return(1e10);
  }else{
    int i;
    for(i=0;i<npop;i++) {
      weibll_val =
        weibll_val -
            log(gamma/(pow(beta,gamma))) - (gamma-1)*log(p[i]) + pow((p[i]/beta),
                gamma);
    }
        return(weibll_val);
  }
}


//write a program to sort vectors
void sorting(double x[npop]){
  int i;
  int j;
  double temp;
  for (i=0;i<npop;i++){
    for (j=i+1;j<npop;j++){
      if (x[j] < x[i]){
        temp=x[i];
        x[i]=x[j];
        x[j]=temp;
      }
    }
  }
  return;
}

//Secant Method root finding.
double bisect(double lo, double hi, double e, int max, double mu[2])
{
  double mid;
  int i;
  for(i=0;i<50;i++){
  mid = (hi+lo)/2;
  if((gsl_sf_gamma(1+2/hi)/pow(gsl_sf_gamma(1+1/hi),2) - mu[1]/pow(mu[0],2))
       *(gsl_sf_gamma(1+2/mid)/pow(gsl_sf_gamma(1+1/mid),2) - mu[1]/pow(mu[0],2)) <= 0){
    lo=mid;}
  else if((gsl_sf_gamma(1+2/lo)/pow(gsl_sf_gamma(1+1/lo),2) - mu[1]/pow(mu[0],2))
       *(gsl_sf_gamma(1+2/mid)/pow(gsl_sf_gamma(1+1/mid),2) - mu[1]/pow(mu[0],2)) < 0){
    hi=mid;}
  }
  return mid;
}



//Function to find minimum between two numbers.
double min(double val1, double val2){
double minimum;
if(val1<val2){
        minimum=val1;
}else{
        minimum=val2;
}
return(minimum);
}

```

```c
// ##### GENERATE DATA
void gendata (double theta[nparam], double par[npop], FILE *sim_data,int j)
{
  /* ###### set up random number generator ########################## */
  const gsl_rng_type * TT;
  gsl_rng_env_setup();
  TT = gsl_rng_default;
  r = gsl_rng_alloc (TT);
  gsl_rng_set(r,j);  // set seed based on simulation number
  gsl_rng_free (r);  // function frees all the memory associated with the generator r
  /* ############################################################## */
  int i;
  double runif;
  double alpha;

  for(i=0;i<npop;i++) {
    runif= gsl_rng_uniform(r);
    par[i]=theta[1]*pow(-log(runif),1/theta[0]);
    fprintf(sim_data,"%f,",par[i]);
  }
  fprintf(sim_data,"\n");
  return;
}



/*************************Least Squares Regression Function ****************************/
// This is a program in C to conduct least squares regression with
// the following data points.

void regress(double y[npop],double x[npop],double beta[2]){
  int points=npop;
  int i;
  double xbar=0;
  double ybar=0;

  double div=points;
  //compute xbar & ybar
  for(i=0;i<points;i++){
    xbar=xbar+x[i]/div;
    ybar=ybar+y[i]/div;
  }
  double fract[2];
  fract[0]=0;
  fract[1]=0;
  //now multiply to get our beta extimates
  for(i=0;i<points;i++){
    fract[0]=fract[0]+(x[i]-xbar)*y[i];
    fract[1]=fract[1]+pow((x[i]-xbar),2);
  }
  beta[1]=fract[0]/fract[1];
  beta[0]=ybar-beta[1]*xbar;

 return;
}


        /** Median Rank Regression Method **/
void medianrank(double data[npop], double thta[nparam]){
        double MRR[nparam];
    MRR[0]=MRR[1]=0;
        int k;
        double ynew[npop];
        double xnew[npop];
        for(k=0;k<npop;k++){
          //compute y's
          // log(log(1/(1-(i-.3)/(npop+.4))))
          ynew[k]=log(log(1/(1-((k+1)-.3)/(npop+.4))));

          //compute x's
          // log(x[i])
          xnew[k]=log(data[k]);
        }
        //fit the model.
        regress(ynew,xnew,MRR);

        //compute parameters.
        //gamma=c1
        thta[0]=MRR[1];
        //beta=exp(-c0/gamma)
        thta[1]=exp(-MRR[0]/thta[0]);

    return;
}


int
main(void)
{


  /* ###### set up multimin ######################################## */
  const gsl_multimin_fminimizer_type *T =
  gsl_multimin_fminimizer_nmsimplex2;
  gsl_multimin_fminimizer *s = NULL;
  gsl_vector *ss;
  gsl_vector *x;
  gsl_multimin_function minex_func;
  size_t iter;
  int status;
  double size;
  /* ############################################################## */

  double par[npop];
  double pargamma[npop];
  double theta[nparam];
  //theta=2,2.5

  //gamma
  theta[0] = 2;
  //beta
  theta[1] = 2.5;

  int j;
  //for MRR
```

```
double MRRest[nparam];
MRRest[0]=MRRest[1]=0;

FILE *est_mle;
est_mle= fopen(”../simdata/bigB/mle_est1_big.csv”, ”w”);
FILE *est_mrr;
est_mrr = fopen(”../simdata/bigB/mrr_est1_big.csv”, ”w”);
FILE *est_mom;
est_mom = fopen(”../simdata/bigB/mom_est1_big.csv”, ”w”);
FILE *sim_data;
sim_data = fopen(”../simdata/bigB/sim_data1_bigA.csv”, ”w”);

for(j=0;j<nsim;j++) {

  iter=0;
  // generate data
  gendata(theta,par,sim_data,j);

      double parsort[npop];
      int k;
      for(k=0;k<npop;k++){
        parsort[k]=par[k];
      }

      /** Maximum Likelihood Estimation **/

      // Starting values (for optimization)
  x = gsl_vector_alloc (nparam);
  gsl_vector_set (x, 0, 1.0); // set initial gamma value
  gsl_vector_set (x, 1, 2.0); // set initial beta value

  // Set initial step sizes to 1
  ss = gsl_vector_alloc (nparam);
  gsl_vector_set_all (ss, 1.0);

  // Initialize method and iterate
  minex_func.n = nparam;     // number of parameters
  minex_func.f = weibull_ll;     // function of interest
  minex_func.params = parsort;  // data

  s = gsl_multimin_fminimizer_alloc (T, nparam);
  gsl_multimin_fminimizer_set (s, &minex_func, x, ss);

  do
    {
      status = gsl_multimin_fminimizer_iterate(s);
      if (status)
        break;
      size = gsl_multimin_fminimizer_size (s);
      status = gsl_multimin_test_size (size, 1e-5);
      if (status == GSL_SUCCESS)
        {
          fprintf(est_mle,"%f, %f\n",gsl_vector_get (s->x, 0),gsl_vector_get (s->x, 1));
        }
    }
  while (status == GSL_CONTINUE && iter < 100);
  gsl_vector_free(x);
  gsl_vector_free(ss);
  gsl_multimin_fminimizer_free (s);


  /** Median Rank Regression Method **/

      //sort the data.
      sorting(parsort);

      medianrank(parsort,MRRest);

      //print to file.
      fprintf(est_mrr,"%f, %f\n",MRRest[0],MRRest[1]);

      /** Method of Moments Estimation **/

      double MOMest[nparam];
      MOMest[0]=MOMest[1]=1;
      //define moments.
      double mu[2];
```

```
        mu[0]=mu[1]=0;
        for(k=0;k<npop;k++){
          double n=npop;
          // mu1=E(x)
          mu[0]=mu[0]+parsort[k]/n;
          // mu2=E(xˆ2)
          mu[1]=mu[1]+pow(parsort[k],2)/n;
        }


        //Use root finding to estimate gamma
        double max=1000;
        double error=1e−100;
    MOMest[0]=bisect(.1,101.3,error,max,mu);

        //Solve for beta
        // beta=mu1/GAMMA(1+1/gamma)
        MOMest[1]=mu[0] / gsl_sf_gamma(1+1/MOMest[0]);
        //print to file.
    fprintf(est_mom,”%f, %f\n”,MOMest[0],MOMest[1]);
  }

  fclose(est_mom);
  fclose(est_mrr);
  fclose(est_mle);
  fclose(sim_data);
  return;
}
```

B.2 Code to Generate from Gamma for Misspecification Studies

```
/******************
Mark Nielsen
C Project
******************/


#include <stdio.h>
#include <gsl/gsl_rng.h>          // required for random number generation
#include <gsl/gsl_randist.h>      // required for data generation from distributions
```

```
#include <gsl/gsl_multimin.h>      // required for use of multimin (optimization)
//#include <gsl/gsl_cdf.h>           // required for CDF values
//#include <gsl/gsl_statistics.h>
//#include <gsl/gsl_sort.h>
#include <gsl/gsl_sf_gamma.h>

#include <math.h>

# define npop 1000     // number of data points
# define nparam 2      // number of parameters to be estimated in function
# define nsim 1       // number of simulations

gsl_rng * r;  /* global generator */

//This is my Weibull function
double weibull_pdf(double dat,double theta[nparam]) {
  double weibull;
  weibull=theta[0]/(pow(theta[1],theta[0]))*pow(dat,(theta[0]-1))*exp(-pow(dat/theta[1],theta[0]));
  return(weibull);
}


// ##### DEFINE FUNCTION TO BE MINIMIZED
// params = data
// v = vector of parameters to be identified
double weibull_ll (const gsl_vector *v, void *params)
{
  double *p = (double *)params; // convert data into readable format
  double weibll_val = 0;         // initialize value of neg loglikelihood
  double gamma, beta;
  gamma = gsl_vector_get(v, 0);     // assign values of param vector to specific variables
  beta = gsl_vector_get(v, 1);
  if(gamma<=0){
    return(1e10);
  }else if(beta<=0){
    return(1e10);
  }else{
    int i;
    for(i=0;i<npop;i++) {
      weibll_val =
        weibll_val -
            log(gamma/(pow(beta,gamma))) - (gamma-1)*log(p[i]) + pow((p[i]/beta),gamma);
    }
        return(weibll_val);
  }
}


//write a program to sort vectors
void sorting(double x[npop]){
  int i;
  int j;
  double temp;
  for (i=0;i<npop;i++){
    for (j=i+1;j<npop;j++){
      if (x[j] < x[i]){
        temp=x[i];
        x[i]=x[j];
        x[j]=temp;
      }
    }
  }
  return;
}

//Secant Method root finding.
double bisect(double lo, double hi, double e, int max, double mu[2])
{
  double mid;
  int i;
  for(i=0;i<50;i++){
  mid = (hi+lo)/2;
  if((gsl_sf_gamma(1+2/hi)/pow(gsl_sf_gamma(1+1/hi),2) - mu[1]/pow(mu[0],2))
       *(gsl_sf_gamma(1+2/mid)/pow(gsl_sf_gamma(1+1/mid),2) - mu[1]/pow(mu[0],2)) <= 0){
```

```
    lo=mid;}
  else if((gsl_sf_gamma(1+2/lo)/pow(gsl_sf_gamma(1+1/lo),2) - mu[1]/pow(mu[0],2))
       *(gsl_sf_gamma(1+2/mid)/pow(gsl_sf_gamma(1+1/mid),2) - mu[1]/pow(mu[0],2)) < 0){
    hi=mid;}
  }
  return mid;
}



//Function to find minimum between two numbers.
double min(double val1, double val2){
double minimum;
if(val1<val2){
        minimum=val1;
}else{
        minimum=val2;
}
return(minimum);
}

// ##### GENERATE DATA for Gamma(n,1)
void gammadata (double n, double par[npop], FILE *sim_data2,int j)
{
  /* ###### set up random number generator ########################## */
  const gsl_rng_type * TT;
  gsl_rng_env_setup();
  TT = gsl_rng_default;
  r = gsl_rng_alloc (TT);
  gsl_rng_set(r,j);  // set seed based on simulation number
  gsl_rng_free (r);  // function frees all the memory associated with the generator r
  /* ################################################################ */
  int i,k;
  double runif;
  double alpha;

  for(i=0;i<npop;i++) {
    par[i]=0;
    for(k=0;k<n;k++){
    runif= gsl_rng_uniform(r);
    par[i]=par[i]-log(runif);
        }
    fprintf(sim_data2,”%f,”,par[i]);
  }
  fprintf(sim_data2,”\n”);
  return;
}


/*************************Least Squares Regression Function *****************************/
// This is a program in C to conduct least squares regression with
// the following data points.

void regress(double y[npop],double x[npop],double beta[2]){
  int points=npop;
  int i;
  double xbar=0;
  double ybar=0;

  double div=points;
  //compute xbar & ybar
  for(i=0;i<points;i++){
    xbar=xbar+x[i]/div;
    ybar=ybar+y[i]/div;
  }
  double fract[2];
  fract[0]=0;
  fract[1]=0;
  //now multiply to get our beta extimates
  for(i=0;i<points;i++){
    fract[0]=fract[0]+(x[i]-xbar)*y[i];
    fract[1]=fract[1]+pow((x[i]-xbar),2);
  }
  beta[1]=fract[0]/fract[1];
  beta[0]=ybar-beta[1]*xbar;
```

```

 return;
}


        /** Median Rank Regression Method **/
void medianrank(double data[npop], double thta[nparam]){
        double MRR[nparam];
    MRR[0]=MRR[1]=0;
        int k;
        double ynew[npop];
        double xnew[npop];
        for(k=0;k<npop;k++){
          //compute y's
          // log(log(1/(1-(i-.3)/(npop+.4))))
          ynew[k]=log(log(1/(1-((k+1)-.3)/(npop+.4))));

          //compute x's
          // log(x[i])
          xnew[k]=log(data[k]);
        }
        //fit the model.
        regress(ynew,xnew,MRR);

        //compute parameters.
        //gamma=c1
        thta[0]=MRR[1];
        //beta=exp(-c0/gamma)
        thta[1]=exp(-MRR[0]/thta[0]);

    return;
}


int
main(void)
{


```

```
/* ###### set up multimin ####################################### */
const gsl_multimin_fminimizer_type *T =
gsl_multimin_fminimizer_nmsimplex2;
gsl_multimin_fminimizer *s = NULL;
gsl_vector *ss;
gsl_vector *x;
gsl_multimin_function minex_func;
size_t iter;
int status;
double size;
/* ############################################################ */

double par[npop];
double pargamma[npop];
double theta[nparam];

int j;
//for MRR
double MRRest[nparam];
MRRest[0]=MRRest[1]=0;

FILE *est_mle;
est_mle= fopen(”../simdata/gamma/mle_est.csv”, ”w”);
FILE *est_mrr;
est_mrr = fopen(”../simdata/gamma/mrr_est.csv”, ”w”);
FILE *est_mom;
est_mom = fopen(”../simdata/gamma/mom_est.csv”, ”w”);
FILE *sim_data2;
sim_data2 = fopen(”../simdata/gamma/sim_data2.csv”, ”w”);

for(j=0;j<nsim;j++) {

  iter=0;
  // generate data
  //gendata(theta,par,sim_data,j);
      int n=4;
      gammadata(n,par,sim_data2,j);

      double parsort[npop];
```

```
    int k;
    for(k=0;k<npop;k++){
      parsort[k]=par[k];
    }

    /** Maximum Likelihood Estimation **/

    // Starting values (for optimization)
x = gsl_vector_alloc (nparam);
gsl_vector_set (x, 0, 1.0); // set initial gamma value
gsl_vector_set (x, 1, 2.0); // set initial beta value

// Set initial step sizes to 1
ss = gsl_vector_alloc (nparam);
gsl_vector_set_all (ss, 1.0);

// Initialize method and iterate
minex_func.n = nparam;     // number of parameters
minex_func.f = weibull_ll;     // function of interest
minex_func.params = parsort;  // data

s = gsl_multimin_fminimizer_alloc (T, nparam);
gsl_multimin_fminimizer_set (s, &minex_func, x, ss);

do
  {
    status = gsl_multimin_fminimizer_iterate(s);
    if (status)
      break;
    size = gsl_multimin_fminimizer_size (s);
    status = gsl_multimin_test_size (size, 1e-5);
    if (status == GSL_SUCCESS)
      {
        fprintf(est_mle,”%f, %f\n”,gsl_vector_get (s->x, 0),gsl_vector_get (s->x
            , 1));
      }
  }
while (status == GSL_CONTINUE && iter < 100);
gsl_vector_free(x);
```

```
    gsl_vector_free(ss);
    gsl_multimin_fminimizer_free (s);


    /** Median Rank Regression Method **/

        //sort the data.
        sorting(parsort);

        medianrank(parsort,MRRest);

        //print to file.
        fprintf(est_mrr,”%f, %f\n”,MRRest[0],MRRest[1]);

        /** Method of Moments Estimation **/

        double MOMest[nparam];
        MOMest[0]=MOMest[1]=1;
        //define moments.
        double mu[2];
        mu[0]=mu[1]=0;
        for(k=0;k<npop;k++){
          double n=npop;
          // mu1=E(x)
          mu[0]=mu[0]+parsort[k]/n;
          // mu2=E(x^2)
          mu[1]=mu[1]+pow(parsort[k],2)/n;
        }


        //Use root finding to estimate gamma
        double max=1000;
        double error=1e-100;
    MOMest[0]=bisect(.1,101.3,error,max,mu);

        //Solve for beta
        // beta=mu1/GAMMA(1+1/gamma)
        MOMest[1]=mu[0] / gsl_sf_gamma(1+1/MOMest[0]);
        //print to file.
        fprintf(est_mom,"%f, %f\n",MOMest[0],MOMest[1]);
    }

    fclose(est_mom);
    fclose(est_mrr);
    fclose(est_mle);
    fclose(sim_data2);
    return;
}
```

APPENDIX C

R-CODE

C.1 Code to Generate from Normals for Misspecification Studies

```
###########
# Mark Nielsen
# Final Project

library(spuRs)
library(xtable)
library(msm)

#########################################################
####repeat this last part for misspecified gamma(4,0);

miss=read.csv(”../simdata/gamma/sim_data2.csv”,header=F)
miss=as.numeric(miss[-1001])

mle.miss=read.csv(”../simdata/gamma/mle_est.csv”,header=F)
mom.miss=read.csv(”../simdata/gamma/mom_est.csv”,header=F)
mrr.miss=read.csv(”../simdata/gamma/mrr_est.csv”,header=F)

#KS tests
lg.gamma=c(
ks.test(miss,'pweibull',mle.miss[1,1],mle.miss[1,2])$p.value,#MLE is best!
ks.test(miss,'pweibull',mom.miss[1,1],mom.miss[1,2])$p.value,
ks.test(miss,'pweibull',mrr.miss[1,1],mrr.miss[1,2])$p.value,
ks.test(miss,'pgamma',4,1)$p.value)

sm.miss=read.csv(”../simdata/gamma/sm_sim_data2.csv”,header=F)
sm.miss=as.numeric(sm.miss[-31])

sm.mle.miss=read.csv(”../simdata/gamma/sm_mle_est.csv”,header=F)
sm.mom.miss=read.csv(”../simdata/gamma/sm_mom_est.csv”,header=F)
sm.mrr.miss=read.csv(”../simdata/gamma/sm_mrr_est.csv”,header=F)

```

```
#KS tests
sm.gamma=c(
ks.test(sm.miss,'pweibull',sm.mle.miss[1,1],sm.mle.miss[1,2])$p.value,
ks.test(sm.miss,'pweibull',sm.mom.miss[1,1],sm.mom.miss[1,2])$p.value,
ks.test(sm.miss,'pweibull',sm.mrr.miss[1,1],sm.mrr.miss[1,2])$p.value,#MRR is best!
ks.test(sm.miss,'pgamma',4,1)$p.value)

names(lg.gamma)=names(sm.gamma)=c('MLE','MME','MRR','Gamma')
xtable(cbind(sm.gamma,lg.gamma),digits=4)

#avg.miss=rbind(mean(mle.miss),mean(mom.miss),mean(mrr.miss))
#var.miss=rbind(diag(var(mle.miss)),diag(var(mom.miss)),diag(var(mrr.miss)))

#Graphical Summaries
pdf("../Graphs/misspec.pdf",width=14)
par(mfrow=c(1,2))

hist(sm.miss,freq=F,breaks=5, xlab='',ylim=c(0,.25),xlim=c(0,16), main='Misspecification - Gamma(4,1) n=30',cex.main=1.5)
curve(dweibull(x,sm.mrr.miss[1,1],sm.mrr.miss[1,2]),from=0,to=15,add=T,col='darkred',lwd=2.5,lty=2)
curve(dweibull(x,sm.mle.miss[1,1],sm.mle.miss[1,2]),from=0,to=15,add=T,col='blue',lwd=2.5,lty=1)
curve(dweibull(x,sm.mom.miss[1,1],sm.mom.miss[1,2]),from=0,to=15,add=T,col='orange',lwd=2.5,lty=2)
curve(dgamma(x,4,1),from=0,to=15,add=T,col='black',lwd=2.5,lty=3)
legend(6.5,.2,c("MLE","MME","MRR","Gamma"),col=c('blue','orange','darkred','black'),lwd=c(2.5,2.5,2.5,2.5), lty=c(1,2,2,3),bty='n',cex=1.5)

hist(miss,freq=F,breaks=20, xlab='',ylim=c(0,.25),xlim=c(0,16), main='Misspecification - Gamma(4,1) n=1000',cex.main=1.5)
curve(dweibull(x,mrr.miss[1,1],mrr.miss[1,2]),from=0,to=15,add=T,col='darkred',lwd=2.5,lty=2)
curve(dweibull(x,mle.miss[1,1],mle.miss[1,2]),from=0,to=15,add=T,col='blue',lwd=2.5,lty=1)
curve(dweibull(x,mom.miss[1,1],mom.miss[1,2]),from=0,to=15,add=T,col='orange',lwd=2.5,lty=2)
curve(dgamma(x,4,1),from=0,to=15,add=T,col='black',lwd=2.5,lty=3)
```

```
legend(7,.2,c(”MLE”,”MME”,”MRR”,”Gamma”),col=c('blue','orange','darkred','black'),
    lwd=c(2.5,2.5,2.5,2.5), lty=c(1,2,2,3),bty='n',cex=1.5)
#dev.off()
dev.off()
parsort=sort(miss)
source('sourcecode.r')
MRRest=medianrank(parsort)
thta=MRRest[[4]]

# plot how this is estimated
#pdf(”../Graphs/MRRgamma.pdf”)
#plot(log(parsort),MRRest[[3]], main='Median Rank Regression',
#   xlab='log(t)', ylab='log(1/S(t))',pch=19,cex.main=2,cex.lab=1.5)
#abline(thta[1],thta[2],lty=1,lwd=2,col='blue')
#text(3,-1,expression(y == psi[0] + psi[1]*x),col='blue',cex=2)
#dev.off()

# what about for a truncated normal?
set.seed(100)
miss2=rtnorm(1000,5,1,lower=0)
sm.miss2=rtnorm(30,5,1,lower=0)


#/** Median Rank Regression Method **/
#//sort the data.
parsort2=sort(miss2)
sm.parsort2=sort(sm.miss2)
MRRest2=medianrank(parsort2)
sm.MRRest2=medianrank(sm.parsort2)
thta2=MRRest2[[4]]
sm.thta2=sm.MRRest2[[4]]
MRR2=MRRest2[[1]]
sm.MRR2=sm.MRRest2[[1]]

#/** Method of Moments Estimation **/
MOMest2=rep(1,2)
sm.MOMest2=rep(1,2)
#//define moments.
mu2=rep(0,2)
```

```
sm.mu2=rep(0,2)
#// mu1=E(x)
mu2[1]=sum(miss2)/length(miss2)
sm.mu2[1]=sum(sm.miss2)/length(sm.miss2)
#// mu2=E(x^2)
mu2[2]=sum(miss2^2)/length(miss2)
sm.mu2[2]=sum(sm.miss2^2)/length(sm.miss2)
#//Use root finding to estimate gamma
MOMest2[1]=bisection(.06, 10.2,mu2, tol = 1e-6)
sm.MOMest2[1]=bisection(.06,10.2,sm.mu2,tol=1e-6)
#//Solve for beta
#// beta=mu1/GAMMA(1+1/gamma)
MOMest2[2]=mu2[1] / gamma(1+1/MOMest2[1])
sm.MOMest2[2]=sm.mu2[1]/gamma(1+1/sm.MOMest2[1])

#/** Maximum Likelihood Estimation **/
MLEest2=nlm(weibull.ll,p=c(5,5), dat=miss2)$estimate
sm.MLEest2=nlm(weibull.ll,p=c(5,5), dat=sm.miss2)$estimate

pdf('../Graphs/miss_norm.pdf',width=14)
par(mfrow=c(1,2))
hist(sm.miss2, ylim=c(0,.6),xlim=c(2,10),freq=F, breaks=4, xlab='', main='
    Misspecification - Normal(5,1) n=30',cex.main=1.5)
curve(dnorm(x,5,1)/(1-pnorm(0,5,1)), from=0, to=10, add=T, col='black',lwd=2.5,lty
    =3)
curve(dweibull(x,sm.MLEest2[1],sm.MLEest2[2]), from=0, to=10, add=T, col='blue',lwd
    =2.5)
curve(dweibull(x,sm.MOMest2[1],sm.MOMest2[2]), from=0, to=10, add=T, col='orange',
    lwd=2.5)
curve(dweibull(x,sm.MRR2[1],sm.MRR2[2]), from=0, to=10, add=T, col='darkred',lwd
    =2.5,lty=2)
legend(7,.45,c("MLE","MME","MRR","Normal"),col=c('blue','orange','darkred','black'),
    lwd=2.5, lty=c(1,1,2,3), bty='n',cex=1.5)

hist(miss2, ylim=c(0,.45),xlim=c(2,10),freq=F, breaks=20, xlab='', main='
    Misspecification - Normal(5,1) n=1000',cex.main=1.5)
curve(dnorm(x,5,1)/(1-pnorm(0,5,1)), from=0, to=10, add=T, col='black',lwd=2.5,lty
    =3)
curve(dweibull(x,MLEest2[1],MLEest2[2]), from=0, to=10, add=T, col='blue',lwd=2.5)
curve(dweibull(x,MOMest2[1],MOMest2[2]), from=0, to=10, add=T, col='orange',lwd=2.5)
curve(dweibull(x,MRR2[1],MRR2[2]), from=0, to=10, add=T, col='darkred',lwd=2.5,lty=2)
legend(7,.35,c("MLE","MME","MRR","Normal"),col=c('blue','orange','darkred','black'),lwd=2.5, lty=c(1,1,2,3), bty='n',cex=1.5)
dev.off()


#KS tests
lg.norm=c(
ks.test(miss2,'pweibull',MLEest2[1],MLEest2[2])$p.value,
ks.test(miss2,'pweibull',MOMest2[1],MOMest2[2])$p.value,#MOM is best!
ks.test(miss2,'pweibull',MRR2[1],MRR2[2])$p.value,
ks.test(miss2,'ptnorm',5,1,0)$p.value)
sm.norm=c(
ks.test(sm.miss2,'pweibull',sm.MLEest2[1],sm.MLEest2[2])$p.value,#MLE is best!
ks.test(sm.miss2,'pweibull',sm.MOMest2[1],sm.MOMest2[2])$p.value,
ks.test(sm.miss2,'pweibull',sm.MRR2[1],sm.MRR2[2])$p.value,
ks.test(sm.miss2,'ptnorm',5,1,0)$p.value)
names(lg.norm)=names(sm.norm)=c('MLE','MME','MRR','Normal')
xtable(cbind(sm.norm,lg.norm),digits=4)

#another truncated normal
set.seed(1234)
miss3=rtnorm(10000,2,3,lower=0)
sm.miss3=rtnorm(30,2,3,lower=0)

#/** Median Rank Regression Method **/
#//sort the data.
parsort3=sort(miss3)
sm.parsort3=sort(sm.miss3)
MRRest3=medianrank(parsort3)
sm.MRRest3=medianrank(sm.parsort3)
thta3=MRRest3[[4]]
sm.thta3=sm.MRRest3[[4]]
MRR3=MRRest3[[1]]
sm.MRR3=sm.MRRest3[[1]]

#/** Method of Moments Estimation **/
```

```
MOMest3=rep(1,2)
sm.MOMest3=rep(1,2)
#//define moments.
mu3=rep(0,2)
sm.mu3=rep(0,2)
#// mu1=E(x)
mu3[1]=sum(miss3)/length(miss3)
sm.mu3[1]=mean(sm.miss3)
#// mu2=E(x^2)
mu3[2]=sum(miss3^2)/length(miss3)
sm.mu3[2]=sum(sm.miss3^2)/length(sm.miss3)
#//Use root finding to estimate gamma
MOMest3[1]=bisection(.06, 10.2,mu3, tol = 1e-6)
sm.MOMest3[1]=bisection(.06,10.2,sm.mu3,tol=1e-6)
#//Solve for beta
#// beta=mu1/GAMMA(1+1/gamma)
MOMest3[2]=mu3[1] / gamma(1+1/MOMest3[1])
sm.MOMest3[2]=sm.mu3[1]/gamma(1+1/sm.MOMest3[1])

#/** Maximum Likelihood Estimation **/
MLEest3=nlm(weibull.ll,p=c(5,5), dat=miss3)$estimate
sm.MLEest3=nlm(weibull.ll,p=c(5,5),dat=sm.miss3)$estimate


pdf('../Graphs/miss_tnorm.pdf',width=14)
par(mfrow=c(1,2))
hist(sm.miss3, ylim=c(0,.3),xlim=c(0,12),freq=F,breaks=10, xlab='', main='Misspecification - Zero Truncated Normal(2,3) n=30',cex.main=1.5)
curve(dnorm(x,2,3)/(1-pnorm(0,2,3)), from=0, to=12, add=T, col='black',lwd=2.5,lty=3)
curve(dweibull(x,sm.MLEest3[1],sm.MLEest3[2]), from=0, to=12, add=T, col='blue',lwd=2.5,lty=2)
curve(dweibull(x,sm.MOMest3[1],sm.MOMest3[2]), from=0, to=12, add=T, col='orange',lwd=2.5)
curve(dweibull(x,sm.MRR3[1],sm.MRR3[2]), from=0, to=12, add=T, col='darkred',lwd=2.5,lty=1)
legend(5,.25,c("MLE","MME","MRR","Truncated Normal"),col=c('blue','orange','darkred','black'),lwd=2.5, lty=c(2,1,1,3), bty='n',cex=1.5)

```

```r
hist(miss3, ylim=c(0,.3),xlim=c(0,12),freq=F,breaks=20, xlab='', main='Misspecification - Zero Truncated Normal(2,3) n=1000',cex.main=1.5)
curve(dnorm(x,2,3)/(1-pnorm(0,2,3)), from=0, to=12, add=T, col='black',lwd=2.5,lty=3)
curve(dweibull(x,MLEest3[1],MLEest3[2]), from=0, to=12, add=T, col='blue',lwd=2.5,lty=2)
curve(dweibull(x,MOMest3[1],MOMest3[2]), from=0, to=12, add=T, col='orange',lwd=2.5)
curve(dweibull(x,MRR3[1],MRR3[2]), from=0, to=12, add=T, col='darkred',lwd=2.5,lty=1)
legend(4,.25,c("MLE","MME","MRR","Truncated Normal"),col=c('blue','orange','darkred','black'),lwd=2.5, lty=c(2,1,1,3), bty='n',cex=1.5)
dev.off()


#Kolmogorov-Smirnov Test
lg.trnorm=c(
ks.test(miss3,'pweibull',MLEest3[1],MLEest3[2])$p.value,
ks.test(miss3,'pweibull',MOMest3[1],MOMest3[2])$p.value,#MOM is best!
ks.test(miss3,'pweibull',MRR3[1],MRR3[2])$p.value,
ks.test(miss3,'ptnorm',2,3,0)$p.value)

sm.trnorm=c(
ks.test(sm.miss3,'pweibull',sm.MLEest3[1],sm.MLEest3[2])$p.value,
ks.test(sm.miss3,'pweibull',sm.MOMest3[1],sm.MOMest3[2])$p.value,#MOM is best!
ks.test(sm.miss3,'pweibull',sm.MRR3[1],sm.MRR3[2])$p.value,
ks.test(sm.miss3,'ptnorm',2,3,0)$p.value)
names(lg.trnorm)=names(sm.trnorm)=c('MLE','MME','MRR','TruncNorm')
xtable(cbind(sm.trnorm,lg.trnorm),digits=-4)


xtable(rbind(sm.gamma,lg.gamma,sm.norm,lg.norm,sm.trnorm,lg.trnorm))
```

C.2 Code to compute MSE

```r
###########
# Mark Nielsen
# Masters Project
library(spuRs)
library(xtable)

```

```
#setwd("C:/Users/Owner/Documents/My Dropbox/Classes/STAT 624/")


#comparison of pdfs
x=seq(from=0,to=2.5,length.out=100)
pdf('../Graphs/Weibull_PDF.pdf')
plot(x,dweibull(x,.5,1),type='l',col='blue',lwd=2.5,ylab='Density',main=expression(paste(gamma,' - Shape parameter'),sep=''),xlab='',cex.main=2)
points(x,dweibull(x,1,1),type='l',col='red',lwd=2.5)
points(x,dweibull(x,1.5,1),type='l',col='black',lwd=2.5)
points(x,dweibull(x,5,1),type='l',col='forestgreen',lwd=2.5)
legend(1.25,2.5,c(expression(paste(beta == 1,',', gamma == 0.5)),expression(paste(beta == 1,',', gamma == 1)),expression(paste(beta == 1,',', gamma == 1.5)),expression(paste(beta == 1,',', gamma == 5))),col=c('blue','red','black','forestgreen'),lty=1,lwd=2.5,bty='n',cex=1.5)
dev.off()

#comparison of pdfs
x=seq(from=0,to=2.5,length.out=100)
pdf('../Graphs/Weibull2_PDF.pdf')
plot(x,dweibull(x,2,.5),type='l',col='blue',lwd=2.5,ylab='Density',main=expression(paste(beta,' - Scale parameter'),sep=''),xlab='',cex.main=2)
points(x,dweibull(x,2,1),type='l',col='red',lwd=2.5)
points(x,dweibull(x,2,1.5),type='l',col='black',lwd=2.5)
points(x,dweibull(x,2,5),type='l',col='forestgreen',lwd=2.5)
legend(1.25,1.5,c(expression(paste(beta == 0.5,',', gamma == 2)),expression(paste(beta == 1,'   ', gamma == 2)),expression(paste(beta == 1.5,',', gamma == 2)),expression(paste(beta == 5,'   ', gamma == 2))),col=c('blue','red','black','forestgreen'),lty=1,lwd=2.5,bty='n',cex=1.5)
dev.off()

#pdf of our parameter estimates.
pdf('../Graphs/params.pdf',width=14)
par(mfrow=c(1,2))
x=seq(from=0,to=10,length.out=100)
plot(x,dweibull(x,2,2.5),type='l',col='blue',lwd=2.5,ylab='Density',main=expression(paste(beta == 2,' and ', gamma == 2.5)),xlab='',cex.main=2)
x=seq(from=0,to=150,length.out=100)
```

```
plot(x,dweibull(x,5,90),type='l',col='red',lwd=2.5,ylab='Density',main=expression(
    paste(beta == 5,' and ', gamma == 90)),xlab='',cex.main=2)
dev.off()


###############
# MSE for each Method
#par(2,5)
mle.ty=read.csv("../simdata/tinyA/mle_est1_tiny.csv",header=F)
mle.sm=read.csv("../simdata/smallA/mle_est1_small.csv",header=F)
mle.lg=read.csv("../simdata/bigA/mle_est1_big.csv",header=F)

mom.ty=read.csv("../simdata/tinyA/mom_est1_tiny.csv",header=F)
mom.sm=read.csv("../simdata/smallA/mom_est1_small.csv",header=F)
mom.lg=read.csv("../simdata/bigA/mom_est1_big.csv",header=F)

mrr.ty=read.csv("../simdata/tinyA/mrr_est1_tiny.csv",header=F)
mrr.sm=read.csv("../simdata/smallA/mrr_est1_small.csv",header=F)
mrr.lg=read.csv("../simdata/bigA/mrr_est1_big.csv",header=F)

avg.ty=rbind(mean(mle.ty),mean(mom.ty),mean(mrr.ty))
avg.sm=rbind(mean(mle.sm),mean(mom.sm),mean(mrr.sm))
avg.lg=rbind(mean(mle.lg),mean(mom.lg),mean(mrr.lg))

var.ty=rbind(diag(var(mle.ty)),diag(var(mom.ty)),diag(var(mrr.ty)))
var.sm=rbind(diag(var(mle.sm)),diag(var(mom.sm)),diag(var(mrr.sm)))
var.lg=rbind(diag(var(mle.lg)),diag(var(mom.lg)),diag(var(mrr.lg)))


#par (1,3)
mleb.ty=read.csv("../simdata/tinyB/mle_est1_tiny.csv",header=F)
mleb.sm=read.csv("../simdata/smallB/mle_est1_small.csv",header=F)
mleb.lg=read.csv("../simdata/bigB/mle_est1_big.csv",header=F)

momb.ty=read.csv("../simdata/tinyB/mom_est1_tiny.csv",header=F)
momb.sm=read.csv("../simdata/smallB/mom_est1_small.csv",header=F)
momb.lg=read.csv("../simdata/bigB/mom_est1_big.csv",header=F)

mrrb.ty=read.csv("../simdata/tinyB/mrr_est1_tiny.csv",header=F)
mrrb.sm=read.csv("../simdata/smallB/mrr_est1_small.csv",header=F)
mrrb.lg=read.csv("../simdata/bigB/mrr_est1_big.csv",header=F)

avgb.ty=rbind(mean(mleb.ty),mean(momb.ty),mean(mrrb.ty))
avgb.sm=rbind(mean(mleb.sm),mean(momb.sm),mean(mrrb.sm))
avgb.lg=rbind(mean(mleb.lg),mean(momb.lg),mean(mrrb.lg))

varb.ty=rbind(diag(var(mleb.ty)),diag(var(momb.ty)),diag(var(mrrb.ty)))
varb.sm=rbind(diag(var(mleb.sm)),diag(var(momb.sm)),diag(var(mrrb.sm)))
varb.lg=rbind(diag(var(mleb.lg)),diag(var(momb.lg)),diag(var(mrrb.lg)))

bias=cbind((avg.ty[,1]-5),
           (avg.sm[,1]-5),
           (avg.lg[,1]-5),
           (avg.ty[,2]-90),
           (avg.sm[,2]-90),
           (avg.lg[,2]-90),
           (avgb.ty[,1]-2),
           (avgb.sm[,1]-2),
           (avgb.lg[,1]-2),
           (avgb.ty[,2]-2.5),
           (avgb.sm[,2]-2.5),
           (avgb.lg[,2]-2.5))
colnames(bias)=c("ty.gamma5","sm.gamma5","lg.gamma5","ty.beta90","sm.beta90","lg.beta90","ty.gamma1","sm.gamma1","lg.gamma1","ty.beta3","sm.beta3","lg.beta3")
xtable(bias,digits=5)

var=cbind(var.ty[,1],
          var.sm[,1],
          var.lg[,1],
          var.ty[,2],
          var.sm[,2],
          var.lg[,2],
          varb.ty[,1],
          varb.sm[,1],
          varb.lg[,1],
          varb.ty[,2],
          varb.sm[,2],
          varb.lg[,2])
```

```
colnames(var)=c(”ty.gamma5”,”sm.gamma5”,”lg.gamma5”,”ty.beta90”,”sm.beta90”,”lg.beta90”,”ty.gamma1”,”sm.gamma1”,”lg.gamma1”,”ty.beta3”,”sm.beta3”,”lg.beta3”)
xtable(var,digits=5)

mse=cbind((avg.ty[,1]-5)^2+var.ty[,1],
          (avg.ty[,2]-90)^2+var.ty[,2],
          (avg.sm[,1]-5)^2+var.sm[,1],
          (avg.sm[,2]-90)^2+var.sm[,2],
          (avg.lg[,1]-5)^2+var.lg[,1],
          (avg.lg[,2]-90)^2+var.lg[,2],
          (avgb.ty[,1]-2)^2+varb.ty[,1],
          (avgb.ty[,2]-2.5)^2+varb.ty[,2],
          (avgb.sm[,1]-2)^2+varb.sm[,1],
          (avgb.sm[,2]-2.5)^2+varb.sm[,2],
          (avgb.lg[,1]-2)^2+varb.lg[,1],
          (avgb.lg[,2]-2.5)^2+varb.lg[,2])

mse2=cbind((avg.ty[,1]-5)^2+var.ty[,1],
          (avg.sm[,1]-5)^2+var.sm[,1],
          (avg.lg[,1]-5)^2+var.lg[,1],
          (avg.ty[,2]-90)^2+var.ty[,2],
          (avg.sm[,2]-90)^2+var.sm[,2],
          (avg.lg[,2]-90)^2+var.lg[,2],
          (avgb.ty[,1]-2)^2+varb.ty[,1],
          (avgb.sm[,1]-2)^2+varb.sm[,1],
          (avgb.lg[,1]-2)^2+varb.lg[,1],
          (avgb.ty[,2]-2.5)^2+varb.ty[,2],
          (avgb.sm[,2]-2.5)^2+varb.sm[,2],
          (avgb.lg[,2]-2.5)^2+varb.lg[,2])
colnames(mse2)=c(”ty.gamma5”,”sm.gamma5”,”lg.gamma5”,”ty.beta90”,”sm.beta90”,”lg.beta90”,”ty.gamma1”,”sm.gamma1”,”lg.gamma1”,”ty.beta3”,”sm.beta3”,”lg.beta3”)
xtable(mse2,digits=5)

#GRAPH FOR 10, 100, 1000, 10000, 100000
mle10A=read.csv(”../simdata/A/mle_10.csv”,header=F)
mle1000A=read.csv(”../simdata/A/mle_1000.csv”,header=F)
mle100000A=read.csv(”../simdata/A/mle_100000.csv”,header=F)

mom10A=read.csv(”../simdata/A/mom_10.csv”,header=F)
```

```
mom1000A=read.csv("../simdata/A/mom_1000.csv",header=F)
mom100000A=read.csv("../simdata/A/mom_100000.csv",header=F)

mrr10A=read.csv("../simdata/A/mrr_10.csv",header=F)
mrr1000A=read.csv("../simdata/A/mrr_1000.csv",header=F)
mrr100000A=read.csv("../simdata/A/mrr_100000.csv",header=F)

msefunc=function(dat,param){
  (mean(dat)-param)^2+var(dat)
}

mse.mle=rbind(c(msefunc(mle10A[,1], 5),mse[1,1],mse[1,3],msefunc(mle1000A[,1], 5),mse[1,5]),
              c(msefunc(mle10A[,2],90),mse[1,2],mse[1,4],msefunc(mle1000A[,2],90),mse[1,6]))

mse.mom=rbind(c(msefunc(mom10A[,1], 5),mse[2,1],mse[2,3],msefunc(mom1000A[,1], 5),mse[2,5]),
              c(msefunc(mom10A[,2],90),mse[2,2],mse[2,4],msefunc(mom1000A[,2],90),mse[2,6]))

mse.mrr=rbind(c(msefunc(mrr10A[,1], 5),mse[3,1],mse[3,3],msefunc(mrr1000A[,1], 5),mse[3,5]),
              c(msefunc(mrr10A[,2],90),mse[3,2],mse[3,4],msefunc(mrr1000A[,2],90),mse[3,6]))

pdf('../Graphs/mseA.pdf',width=14)
par(mfrow=c(1,2))
plot(c(10,20,100,1000,10000),mse.mom[1,],xaxt='n',type='b',log="x",xlab='Sample Size',ylab='Mean Square Error',main=expression(gamma==5),col='orange',lwd=3,pch=20,lty=1,cex.main=2)
points(c(10,20,100,1000,10000),mse.mrr[1,],type='b',col='darkred',lwd=3,pch=20,lty=2)
points(c(10,20,100,1000,10000),mse.mle[1,],type='b',col='blue',lwd=3,pch=20,lty=3)
legend('topright',c('MLE','MME','MRR'), lty=c(3,1,2), pch=20,lwd=2,col=c('blue','orange','darkred'),cex=1.5)
axis(1,c(10,20,100,1000,10000))

```

```
plot(c(10,20,100,1000,10000),mse.mrr[2,],xaxt='n',type='b',log="x",xlab='Sample Size',ylab='Mean Square Error',main=expression(beta==90),col='darkred',lwd=3,pch=20,lty=2,cex.main=2)
points(c(10,20,100,1000,10000),mse.mom[2,],type='b',col='orange',lwd=3,pch=20,lty=1)
points(c(10,20,100,1000,10000),mse.mle[2,],type='b',col='blue',lwd=3,pch=20,lty=3)
legend('topright',c('MLE','MME','MRR'), lty=c(3,1,2), pch=20,lwd=2,col=c('blue','orange','darkred'),cex=1.5)
axis(1,c(10,20,100,1000,10000))
dev.off()

#GRBPH FOR 10, 100, 1000, 10000, 100000
mle10B=read.csv("../simdata/B/mle_10.csv",header=F)
mle1000B=read.csv("../simdata/B/mle_1000.csv",header=F)
mle100000B=read.csv("../simdata/B/mle_100000.csv",header=F)

mom10B=read.csv("../simdata/B/mom_10.csv",header=F)
mom1000B=read.csv("../simdata/B/mom_1000.csv",header=F)
mom100000B=read.csv("../simdata/B/mom_100000.csv",header=F)

mrr10B=read.csv("../simdata/B/mrr_10.csv",header=F)
mrr1000B=read.csv("../simdata/B/mrr_1000.csv",header=F)
mrr100000B=read.csv("../simdata/B/mrr_100000.csv",header=F)

msefunc=function(dat,param){
  (mean(dat)-param)^2+var(dat)
}

mse.mleb=rbind(c(msefunc(mle10B[,1],  2),mse[1,7],mse[1,  9],msefunc(mle1000B[,1],2),mse[1,11]),
               c(msefunc(mle10B[,2],2.5),mse[1,8],mse[1,10],msefunc(mle1000B[,2],2.5),mse[1,12]))

mse.momb=rbind(c(msefunc(mom10B[,1],  2),mse[2,7],mse[2,  9],msefunc(mom1000B[,1],2),mse[2,11]),
               c(msefunc(mom10B[,2],2.5),mse[2,8],mse[2,10],msefunc(mom1000B[,2],2.5),mse[2,12]))

mse.mrrb=rbind(c(msefunc(mrr10B[,1],  2),mse[3,7],mse[3,  9],msefunc(mrr1000B[,1],2),mse[3,11]),
                    c(msefunc(mrr10B[,2],2.5),mse[3,8],mse[3,10],msefunc(mrr1000B[,2],2.5),mse[3,12]))

pdf(’../Graphs/mseB.pdf’,width=14)
par(mfrow=c(1,2))
plot(c(10,20,100,1000,10000),mse.momb[1,],ylim=c(0,.65),xaxt=’n’,type=’b’,log=”x”,xlab=’Sample Size’,ylab=’Mean Square Error’,main=expression(gamma==2),col=’orange’,lwd=3,pch=20,lty=1,cex.main=2)
points(c(10,20,100,1000,10000),mse.mrrb[1,],type=’b’,col=’darkred’,lwd=3,pch=20,lty=2)
points(c(10,20,100,1000,10000),mse.mleb[1,],type=’b’,col=’blue’,lwd=3,pch=20,lty=3)
legend(’topright’,c(’MLE’,’MME’,’MRR’), lty=c(3,1,2), pch=20,lwd=2,col=c(’blue’,’orange’,’darkred’),cex=1.5)
axis(1,c(10,20,100,1000,10000))

plot(c(10,20,100,1000,10000),mse.mrrb[2,],xaxt=’n’,type=’b’,log=”x”,xlab=’Sample Size’,ylab=’Mean Square Error’,main=expression(beta==2.5),col=’darkred’,lwd=3,pch=20,lty=2,cex.main=2)
points(c(10,20,100,1000,10000),mse.momb[2,],type=’b’,col=’orange’,lwd=3,pch=20,lty=1)
points(c(10,20,100,1000,10000),mse.mleb[2,],type=’b’,col=’blue’,lwd=3,pch=20,lty=3)
legend(’topright’,c(’MLE’,’MME’,’MRR’), lty=c(3,1,2), pch=20,lwd=2,col=c(’blue’,’orange’,’darkred’),cex=1.5)
axis(1,c(10,20,100,1000,10000))
dev.off()




#MLE
pdf(”../Graphs/norm_mle_big.pdf”,width=14)
par(mfrow=c(1,2))
hist(mle.lg[,2],freq=F,main=’’,xlab=expression(beta),breaks=20)
curve(dnorm(x,avg.lg[1,2],sqrt(var.lg[1,2])),from=88,to=91,add=T,col=’red’,lwd=2)
hist(mle.lg[,1],freq=F,main=’’,xlab=expression(gamma),breaks=20)
curve(dnorm(x,avg.lg[1,1],sqrt(var.lg[1,1])),from=4.85,to=5.15,add=T,col=’red’,lwd=2)
dev.off()

```

```
pdf("../Graphs/norm_mleb_big.pdf",width=14)
par(mfrow=c(1,2))
hist(mleb.lg[,2],freq=F,main='',xlab=expression(beta),breaks=20)
curve(dnorm(x,avgb.lg[1,2],sqrt(varb.lg[1,2])),from=2.45,to=2.55,add=T,col='red',lwd=2)
hist(mleb.lg[,1],freq=F,main='',xlab=expression(gamma),breaks=20)
curve(dnorm(x,avgb.lg[1,1],sqrt(varb.lg[1,1])),from=1.95,to=2.05,add=T,col='red',lwd=2)
dev.off()

pdf("../Graphs/norm_mleb_small.pdf",width=14)
par(mfrow=c(1,2))
hist(mleb.sm[,2],freq=F,main='',xlab=expression(beta),breaks=20)
curve(dnorm(x,avgb.sm[1,2],sqrt(varb.sm[1,2])),from=2,to=4.1,add=T,col='red',lwd=2)
hist(mleb.sm[,1],freq=F,main='',xlab=expression(gamma),breaks=20)
curve(dnorm(x,avgb.sm[1,1],sqrt(varb.sm[1,1])),from=1,to=3,add=T,col='red',lwd=2)
dev.off()

pdf("../Graphs/norm_mle_small.pdf",width=14)
par(mfrow=c(1,2))
hist(mle.sm[,2],freq=F,main='',xlab=expression(beta),breaks=20)
curve(dnorm(x,avg.sm[1,2],sqrt(var.sm[1,2])),from=84,to=96,add=T,col='red',lwd=2)
hist(mle.sm[,1],freq=F,main='',xlab=expression(gamma),breaks=20)
curve(dnorm(x,avg.sm[1,1],sqrt(var.sm[1,1])),from=3.5,to=7,add=T,col='red',lwd=2)
dev.off()

pdf("../Graphs/norm_mleb_tiny.pdf",width=14)
par(mfrow=c(1,2))
hist(mleb.ty[,2],freq=F,main='',xlab=expression(beta),breaks=20)
curve(dnorm(x,avgb.ty[1,2],sqrt(varb.ty[1,2])),from=1,to=6,add=T,col='red',lwd=2)
hist(mleb.ty[,1],freq=F,main='',xlab=expression(gamma),breaks=20)
curve(dnorm(x,avgb.ty[1,1],sqrt(varb.ty[1,1])),from=.5,to=4.5,add=T,col='red',lwd=2)
dev.off()

pdf("../Graphs/norm_mle_tiny.pdf",width=14)
par(mfrow=c(1,2))
hist(mle.ty[,2],freq=F,main='',ylim=c(0,.1),xlab=expression(beta),breaks=20)
curve(dnorm(x,avg.ty[1,2],sqrt(var.ty[1,2])),from=74,to=105,add=T,col='red',lwd=2)
hist(mle.ty[,1],freq=F,main='',xlab=expression(gamma),breaks=20)
```

```
curve(dnorm(x,avg.ty[1,1],sqrt(var.ty[1,1])),from=2,to=12,add=T,col='red',lwd=2)
dev.off()

#MOM
pdf("../Graphs/norm_mom_big.pdf",width=14)
par(mfrow=c(1,2))
hist(mom.lg[,2],freq=F,main='',xlab=expression(beta),breaks=20)
curve(dnorm(x,avg.lg[2,2],sqrt(var.lg[2,2])),from=88,to=91,add=T,col='red',lwd=2)
hist(mom.lg[,1],freq=F,main='',xlab=expression(gamma),breaks=20)
curve(dnorm(x,avg.lg[2,1],sqrt(var.lg[2,1])),from=4.85,to=5.15,add=T,col='red',lwd=2)
dev.off()

pdf("../Graphs/norm_momb_big.pdf",width=14)
par(mfrow=c(1,2))
hist(momb.lg[,2],freq=F,main='',xlab=expression(beta),breaks=20)
curve(dnorm(x,avgb.lg[2,2],sqrt(varb.lg[2,2])),from=2.45,to=2.55,add=T,col='red',lwd=2)
hist(momb.lg[,1],freq=F,main='',xlab=expression(gamma),breaks=20)
curve(dnorm(x,avgb.lg[2,1],sqrt(varb.lg[2,1])),from=1.95,to=2.05,add=T,col='red',lwd=2)
dev.off()

pdf("../Graphs/norm_momb_small.pdf",width=14)
par(mfrow=c(1,2))
hist(momb.sm[,2],freq=F,main='',ylim=c(0,3.25),xlab=expression(beta),breaks=20)
curve(dnorm(x,avgb.sm[2,2],sqrt(varb.sm[2,2])),from=2,to=3,add=T,col='red',lwd=2)
hist(momb.sm[,1],freq=F,main='',xlab=expression(gamma),breaks=20)
curve(dnorm(x,avgb.sm[2,1],sqrt(varb.sm[2,1])),from=1.5,to=2.6,add=T,col='red',lwd=2)
dev.off()

pdf("../Graphs/norm_mom_small.pdf",width=14)
par(mfrow=c(1,2))
hist(mom.sm[,2],freq=F,main='',xlab=expression(beta),breaks=20)
curve(dnorm(x,avg.sm[2,2],sqrt(var.sm[2,2])),from=84,to=96,add=T,col='red',lwd=2)
hist(mom.sm[,1],freq=F,main='',xlab=expression(gamma),breaks=20)
curve(dnorm(x,avg.sm[2,1],sqrt(var.sm[2,1])),from=3.9,to=6.5,add=T,col='red',lwd=2)
dev.off()
```

```
pdf("../Graphs/norm_momb_tiny.pdf",width=14)
par(mfrow=c(1,2))
hist(momb.ty[,2],freq=F,main='',xlab=expression(beta),breaks=20)
curve(dnorm(x,avgb.ty[2,2],sqrt(varb.ty[2,2])),from=0,to=6,add=T,col='red',lwd=2)
hist(momb.ty[,1],freq=F,main='',xlab=expression(gamma),breaks=20)
curve(dnorm(x,avgb.ty[2,1],sqrt(varb.ty[2,1])),from=1,to=4.5,add=T,col='red',lwd=2)
dev.off()

pdf("../Graphs/norm_mom_tiny.pdf",width=14)
par(mfrow=c(1,2))
hist(mom.ty[,2],freq=F,main='',xlab=expression(beta),breaks=20)
curve(dnorm(x,avg.ty[2,2],sqrt(var.ty[2,2])),from=75,to=105,add=T,col='red',lwd=2)
hist(mom.ty[,1],freq=F,main='',xlab=expression(gamma),breaks=20)
curve(dnorm(x,avg.ty[2,1],sqrt(var.ty[2,1])),from=2,to=12,add=T,col='red',lwd=2)
dev.off()


#MRR
pdf("../Graphs/norm_mrr_big.pdf",width=14)
par(mfrow=c(1,2))
hist(mrr.lg[,2],freq=F,main='',xlab=expression(beta),breaks=20)
curve(dnorm(x,avg.lg[3,2],sqrt(var.lg[3,2])),from=89,to=91,add=T,col='red',lwd=2)
hist(mrr.lg[,1],freq=F,main='',xlab=expression(gamma),breaks=20)
curve(dnorm(x,avg.lg[3,1],sqrt(var.lg[3,1])),from=4.8,to=5.2,add=T,col='red',lwd=2)
dev.off()

pdf("../Graphs/norm_mrrb_big.pdf",width=14)
par(mfrow=c(1,2))
hist(mrrb.lg[,2],freq=F,main='',xlab=expression(beta),breaks=20)
curve(dnorm(x,avgb.lg[3,2],sqrt(varb.lg[3,2])),from=2.45,to=2.55,add=T,col='red',lwd=2)
hist(mrrb.lg[,1],freq=F,main='',xlab=expression(gamma),breaks=20)
curve(dnorm(x,avgb.lg[3,1],sqrt(varb.lg[3,1])),from=1.9,to=2.1,add=T,col='red',lwd=2)
dev.off()

pdf("../Graphs/norm_mrrb_small.pdf",width=14)
par(mfrow=c(1,2))
```

```
hist(mrrb.sm[,2],freq=F,main='',xlab=expression(beta),breaks=20)
curve(dnorm(x,avgb.sm[3,2],sqrt(varb.sm[3,2])),from=2,to=3.5,add=T,col='red',lwd=2)
hist(mrrb.sm[,1],freq=F,main='',xlab=expression(gamma),breaks=20)
curve(dnorm(x,avgb.sm[3,1],sqrt(varb.sm[3,1])),from=1.3,to=2.7,add=T,col='red',lwd=2)
dev.off()

pdf("../Graphs/norm_mrr_small.pdf",width=14)
par(mfrow=c(1,2))
hist(mrr.sm[,2],freq=F,main='',xlab=expression(beta),breaks=20)
curve(dnorm(x,avg.sm[3,2],sqrt(var.sm[3,2])),from=84,to=97,add=T,col='red',lwd=2)
hist(mrr.sm[,1],freq=F,main='',xlab=expression(gamma),breaks=20)
curve(dnorm(x,avg.sm[3,1],sqrt(var.sm[3,1])),from=3,to=6.5,add=T,col='red',lwd=2)
dev.off()

pdf("../Graphs/norm_mrrb_tiny.pdf",width=14)
par(mfrow=c(1,2))
hist(mrrb.ty[,2],freq=F,main='',xlab=expression(beta),breaks=20)
curve(dnorm(x,avgb.ty[3,2],sqrt(varb.ty[3,2])),from=1,to=4.5,add=T,col='red',lwd=2)
hist(mrrb.ty[,1],freq=F,main='',xlab=expression(gamma),breaks=20)
curve(dnorm(x,avgb.ty[3,1],sqrt(varb.ty[3,1])),from=.25,to=4,add=T,col='red',lwd=2)
dev.off()

pdf("../Graphs/norm_mrr_tiny.pdf",width=14)
par(mfrow=c(1,2))
hist(mrr.ty[,2],freq=F,main='',xlab=expression(beta),breaks=20)
curve(dnorm(x,avg.ty[3,2],sqrt(var.ty[3,2])),from=73,to=105,add=T,col='red',lwd=2)
hist(mrr.ty[,1],freq=F,main='',xlab=expression(gamma),breaks=20)
curve(dnorm(x,avg.ty[3,1],sqrt(var.ty[3,1])),from=1.5,to=10,add=T,col='red',lwd=2)
dev.off()
```

C.3 Code to read in wind data

```
############
# Mark Nielsen
# Masters project
# creating data sets with only variables of interest.


#setwd('/home/owner/Desktop/Dropbox/Classes/MASTERS PROJECT')
```

```

ab09=read.csv('../winddata/ab_2009.csv',header=T)
ab10=read.csv('../winddata/ab_2010.csv',header=T)
ca09=read.csv('../winddata/ca_2009.csv',header=T)
ca10=read.csv('../winddata/ca_2010.csv',header=T)
gc09=read.csv('../winddata/gc_2009.csv',header=T)
gc10=read.csv('../winddata/gc_2010.csv',header=T)
sb09=read.csv('../winddata/sb_2009.csv',header=T)
sb10=read.csv('../winddata/sb_2010.csv',header=T)
rtmet09=read.csv('../winddata/rtmet_2009.csv',header=T)
rtmet10=read.csv('../winddata/rtmet_2010.csv',header=T)

ab=rbind(ab09[,c(1,80)],ab10[,c(1,80)])
ab$WindS_ms_Ave[ab[,2]==-8999]=NA
hist(ab[,2],freq=F,breaks=40)
ab.nm=ab[!is.na(ab[,2]),]
plot(density(ab.nm[,2]))

ca=rbind(ca09[,c(1,80)],ca10[,c(1,80)])
ca$WindS_ms_Ave[ca[,2]==-8999]=NA
hist(ca[,2],freq=F,breaks=40)
ca.nm=ca[!is.na(ca[,2]),]
plot(density(ca.nm[,2]))

gc=rbind(gc09[,c(1,80)],gc10[,c(1,80)])
gc$WindS_ms_Ave[gc[,2]==-8999]=NA
hist(gc[,2],freq=F,breaks=40)
gc.nm=gc[!is.na(gc[,2]),]
plot(density(gc.nm[,2]))

sb=rbind(sb09[,c(1,80)],sb10[,c(1,80)])
sb$WindS_ms_Ave[sb[,2]==-8999]=NA
hist(sb[,2],freq=F,breaks=40)
sb.nm=sb[!is.na(sb[,2]),]
plot(density(sb.nm[,2]))
curve(dweibull(x,1.4,2.2),from=0, to=14, add=T, col=2)

rtmet=rbind(rtmet09[,c(1,13)],rtmet10[,c(1,13)])
rtmet$WindS_ms_Ave[rtmet[,2]==-9999]=NA
```

```
hist(rtmet[,2],freq=F,breaks=40)
rtmet.nm=rtmet[!is.na(rtmet[,2]),]
plot(density(rtmet.nm[,2]))

dim(ab)[1]
#28285
dim(ca)[1]
#28284
dim(gc)[1]
#28283
dim(sb)[1]
#28283
dim(rtmet)[1]
#9158

write.csv(ab,'../winddata/ab_wind.csv',row.names=F)
write.csv(ca,'../winddata/ca_wind.csv',row.names=F)
write.csv(gc,'../winddata/gc_wind.csv',row.names=F)
write.csv(sb,'../winddata/sb_wind.csv',row.names=F)
write.csv(rtmet,'../winddata/rtmet_wind.csv',row.names=F)
```

C.4 Code for parameter estimation

```
library(xtable)

gc=read.csv('../winddata/gc_wind.csv',header=T)
par=gc[(!is.na(gc[,2])),2]
par[par==0]=.000001
par2=gc[(!is.na(gc[,2]) & gc[,2]!=0),2]
par3=gc[(!is.na(gc[,2])),2]
par3[par3==0]=.0001
par4=gc[(!is.na(gc[,2])),2]
par4[par4==0]=.01

#change the data to bin format (i.e. frequencies of wspeed from 0-.25, .25-.5, etc.)
frq=ceiling(4*gc[!is.na(gc[,2]),2])/4
frq[frq==0]=.25

n1=length(par)
n2=length(par2)
```

```
ratio=n2/n1

source('sourcecode.r')

#/** Maximum Likelihood Estimation **/
MLEest=nlm(weibull.ll,p=c(1,3), dat=par)$estimate
MLEest2=nlm(weibull.ll,p=c(1,3), dat=par2)$estimate
MLEest3=nlm(weibull.ll,p=c(1,3), dat=par3)$estimate
MLEest4=nlm(weibull.ll,p=c(1,3), dat=par4)$estimate
# here are the frequecy formatted data...
MLEest5=nlm(weibull.ll,p=c(1,3), dat=frq)$estimate

#/** Maximum Likelihood Estimation with mixture distribution **/
par1=gc[(!is.na(gc[,2])),2]
MLEmix=nlm(mix.ll,p=c(3,3,.9),dat=par1)$estimate

#plot the curves and see which one fits best...
pdf('../Graphs/gc_MLE.pdf')
hist(gc[,2],freq=F,breaks=40,  main='Maximum Likelihood', xlab='Wind Speed',cex.lab=1.5,cex.main=2)
curve(dweibull(x,MLEest[1],MLEest[2]),from=0,to=10,add=T,col='blue',lwd=2.5)
curve(dweibull(x,MLEest3[1],MLEest3[2]),from=0,to=10,add=T,col='orange',lwd=2.5)
curve(dweibull(x,MLEest4[1],MLEest4[2]),from=0,to=10,add=T,col='green',lwd=2.5)
curve(dweibull(x,MLEest5[1],MLEest5[2]),from=0,to=10,add=T,col='black',lwd=2.5)
curve(ifelse(x==0,1-MLEmix[3],0)+MLEmix[3]*dweibull(x,MLEmix[1],MLEmix[2]),from=0,to=10,add=T,col='red',lwd=2.5)
legend(5.5,.2,c("Mixture Model","0.000001","0.0001","0.01","Bins"),col=c('red','blue','orange','green','black'),lty=c(1,1,1,1,1),lwd=2.5,cex=1.5, bty='n')
dev.off()

tblMLE=rbind(MLEmix[1:2],MLEest2,MLEest5,MLEest,MLEest3,MLEest4)
colnames(tblMLE)=c("gamma","beta")
rownames(tblMLE)=c("MLEmix","MLEdel","MLEbin","MLE000001","MLE0001","MLE01")
xtable(tblMLE,digits=5)


#/** Median Rank Regression Method **/

#//sort the data.
```

```
parsort=sort(par)
parsort2=sort(par2)
parsort3=sort(par3)
parsort4=sort(par4)
parsort5=sort(frq)

MRRest=medianrank(parsort)
MRR=MRRest[[1]]
MRR2=medianrank(parsort2)
grph=MRR2[[3]]
MRR2=MRR2[[1]]
thta=medianrank(parsort2)[[4]]
MRR3=medianrank(parsort3)[[1]]
MRR4=medianrank(parsort4)[[1]]
MRR5=medianrank(parsort5)[[1]]

#plot the curves and see which one fits best...
pdf('../Graphs/gc_MRR.pdf')
hist(gc[,2],freq=F,breaks=40, main='Median Rank Regression', xlab='Wind Speed',cex.lab=1.5,cex.main=2)
curve(dweibull(x,MRR[1],MRR[2]),from=0,to=10,add=T,col='blue',lwd=2.5)
curve(dweibull(x,MRR3[1],MRR3[2]),from=0,to=10,add=T,col='orange',lwd=2.5)
curve(dweibull(x,MRR4[1],MRR4[2]),from=0,to=10,add=T,col='green',lwd=2.5)
curve(dweibull(x,MRR5[1],MRR5[2]),from=0,to=10,add=T,col='black',lwd=2.5)
curve(ifelse(x==0,1-ratio,0)+ratio*dweibull(x,MRR2[1],MRR2[2]),from=0,to=10,add=T,col='red',lwd=2.5)
legend(6,.2,c("Delete","0.000001","0.0001","0.01","Bins"),col=c('red','blue','orange','green','black'),lty=c(1,1,1),lwd=2.5,cex=1.5, bty='n')
dev.off()

# plot how this is estimated
pdf("../Graphs/MRR_gcline.pdf")
plot(log(parsort2),grph, main='Median Rank Regression',
   xlab='log(t)', ylab='log(1/S(t))',pch=19,cex.main=2,cex.lab=1.5)
abline(thta[1],thta[2],lty=1,lwd=2,col='blue')
text(1,-4,expression(y == psi[0] + psi[1]*x),col='blue',cex=2)
dev.off()

tblMRR=rbind(MRR2,MRR5,MRR,MRR3,MRR4)
```

```
colnames(tblMRR)=c(”gamma”,”beta”)
rownames(tblMRR)=c(”MRRdel”,”MRRbin”,”MRR000001”,”MRR0001”,”MRR01”)
xtable(tblMRR, digits=5)


#/** Method of Moments Estimation **/
MOMest=rep(1,2);
MOMest2=rep(1,2);
MOMest3=rep(1,2);
MOMest4=rep(1,2)
MOMest5=rep(1,2)

#//define moments.
mu=rep(0,2)
mu2=rep(0,2)
mu3=rep(0,2)
mu4=rep(0,2)
mu5=rep(0,2)

#// mu1=E(x)
mu[1]=sum(par)/length(par)
mu2[1]=sum(par2)/length(par2)
mu3[1]=sum(par3)/length(par3)
mu4[1]=sum(par4)/length(par4)
mu5[1]=sum(frq)/length(frq)

#// mu2=E(x^2)
mu[2]=sum(par^2)/length(par)
mu2[2]=sum(par2^2)/length(par2)
mu3[2]=sum(par3^2)/length(par3)
mu4[2]=sum(par4^2)/length(par4)
mu5[2]=sum(frq^2)/length(frq)

#//Use root finding to estimate gamma
MOMest[1]=bisection(.06, 10.2,mu, tol = 1e-6)
MOMest2[1]=bisection(.06, 10.2,mu2, tol = 1e-6)
MOMest3[1]=bisection(.06, 10.2,mu3, tol = 1e-6)
MOMest4[1]=bisection(.06, 10.2,mu4, tol = 1e-6)
MOMest5[1]=bisection(.06, 10.2,mu5, tol = 1e-6)
```

```

#//Solve for beta
#// beta=mu1/GAMMA(1+1/gamma)
MOMest[2]=mu[1] / gamma(1+1/MOMest[1])
MOMest2[2]=mu2[1] / gamma(1+1/MOMest2[1])
MOMest3[2]=mu3[1] / gamma(1+1/MOMest3[1])
MOMest4[2]=mu4[1] / gamma(1+1/MOMest4[1])
MOMest5[2]=mu5[1] / gamma(1+1/MOMest5[1])

tblMOM=rbind(MOMest2,MOMest5,MOMest,MOMest3,MOMest4)
colnames(tblMOM)=c(”gamma”,”beta”)
rownames(tblMOM)=c(”MOMdel”,”MOMbin”,”MOM000001”,”MOM0001”,”MOM01”)
xtable(tblMOM, digits=5)

#plot the curves and see which one fits best...
pdf('../Graphs/gc_MOM.pdf')
hist(gc[,2],freq=F,breaks=40, main='Method of Moments', xlab='Wind Speed',cex.lab=1.5,cex.main=2)
curve(dweibull(x,MOMest[1],MOMest[2]),from=0,to=10,add=T,col='blue',lwd=2.5)
curve(dweibull(x,MOMest3[1],MOMest3[2]),from=0,to=10,add=T,col='orange',lwd=2.5,lty=3)
curve(dweibull(x,MOMest4[1],MOMest4[2]),from=0,to=10, add=T,col='green',lwd=2.5, lty=4)
curve(dweibull(x,MOMest5[1],MOMest5[2]),from=0,to=10,add=T,col='black',lwd=2.5, lty=5)
curve(ifelse(x==0,1-ratio,0)+ratio*dweibull(x,MOMest2[1],MOMest2[2]),from=0,to=10,add=T,col='red',lwd=2.5,lty=2)
legend(6,.2,c(”Delete”,”0.000001”,”0.0001”,”0.01”,”Bins”),col=c('red','blue','orange','green','black'),lty=c(1,2,3,4,5),lwd=2.5,cex=1.5, bty='n')
dev.off()

tbl=rbind(MLEmix,c(MOMest2,ratio),c(MRR2,ratio))
colnames(tbl)=c(”gamma”,”beta”,”pi”)
rownames(tbl)=c(”MLEmix”,”MOMdel”,”MRRdel”)
xtable(tbl,digits=5)
```

C.5 Source code

```

##### DEFINE FUNCTION TO BE MINIMIZED
```

```
# dat = data
# theta = vector of parameters to be identified
weibull.ll=function(theta,dat)
{
  if(theta[1]<=0){
    return(1e10)
  }else if(theta[2]<=0){
    return(1e10)
  }else{
    f = -length(dat)*log(theta[1]/theta[2]^theta[1]) - (theta[1]-1)*sum(log(dat)) + sum((dat/theta[2])^theta[1])
    return(f)
  }
}


##### Define function to be minimized with mixture distn.
# dat = data
# theta = vector of parameters to be 'ID'ed
mix.ll=function(theta,dat)
{
  n=length(dat)
  m=length(dat[dat!=0])
  if(theta[1]<=0){
    return(1e10)
  }else if(theta[2]<=0){
    return(1e10)
  }else if(theta[3]<0){
    return(1e10)
  }else if(theta[3]>1){
    return(1e10)
  }else{
    f = (n-m)*log(1-theta[3])+m*log(theta[3])+m*log(theta[1]/theta[2]^theta[1]) + (theta[1]-1)*sum(log(dat[dat!=0])) - sum((dat[dat!=0]/theta[2])^theta[1])
    return(-f)
  }
}


```

```
#      /** Median Rank Regression Method **/
medianrank=function(dat){
  ynew=rep(0,length(dat))
  for(k in 1:length(dat)){
  #//compute y's
  #// log(log(1/(1-(i-.3)/(npop+.4))))
  ynew[k]=log(log(1/(1-(k-.3)/(length(dat)+.4))))
}
  #//compute x's
  #// log(x[i])
  xnew=log(dat)

  #//fit the model.
  #//compute xbar & ybar
  beta=coefficients(lm(ynew~xnew))
  #//compute parameters.
  #//gamma=c1
  thta=rep(0,2)
  thta[1]=beta[2]
  #//beta=exp(-c0/gamma)
  thta[2]=exp(-beta[1]/thta[1])
  return(list(thta,xnew,ynew,beta))
}



#//Use root finding to estimate gamma
bisection=function (x.l, x.r, mu, tol = 1e-09)
{
    if (x.l >= x.r) {
        cat("error: x.l >= x.r \n")
        return(NULL)
    }
    f.l <- gamma(1+2/x.l)/gamma(1+1/x.l)^2-mu[2]/mu[1]^2
    f.r <- gamma(1+2/x.r)/gamma(1+1/x.r)^2-mu[2]/mu[1]^2
    if (f.l * f.r > 0) {
        cat("error: ftn(x.l) * ftn(x.r) > 0 \n")
        return(NULL)
    }
    n <- 0
    while ((x.r - x.l) > tol) {
        x.m <- (x.l + x.r)/2
        f.m <- gamma(1.+2/x.m)/(gamma(1.+1/x.m))^2-mu[2]/mu[1]^2
        if (f.l * f.m < 0) {
            x.r <- x.m
            f.r <- f.m
        }
        else {
            x.l <- x.m
            f.l <- f.m
        }
        n <- n + 1
        cat("at iteration", n, "the root lies between", x.l,
            "and", x.r, "\n")
    }
    return((x.l + x.r)/2)
}
```