

# Tail Index Estimation for Power Law Distributions in R

**Ranjiva Munasinghe**

MIND Analytics & Management

**Pathum Kossinna**

MIND Analytics & Management

**Dovini Jayasinghe**

MIND Analytics & Management

**Dilanka Wijeratne**

MIND Analytics & Management

---

## Abstract

Power law distributions, in particular Pareto distributions, describe data across diverse areas of study. We have developed a package in R to estimate the tail index for such datasets keeping in mind ease of use, speed (in particular with large datasets) and accuracy. In this short article, we provide a user guide to our package along with the results obtained.

*Keywords:* Power law distributions, fat tailed distributions, heavy tailed distributions, Pareto distributions, tail index, tail estimation, R.

---

## 1. Introduction

Power law distributions account for phenomena and observations across diverse fields including physics, astronomy, biology, economics and the social sciences (Sornette 2004). Newman (2005) cites plenty of specific real-world examples including distribution of wealth, word frequency in a given text, population in a city etc. Taleb (2019) states the most general form of the survival function<sup>1</sup> describing power law distributions as

$$\bar{P}(x) = L(x)x^{-\alpha} \quad (1)$$

where  $L(x)$  is a slowly-varying function, i.e.

$$\lim_{x \rightarrow \infty} \frac{L(\lambda x)}{L(x)} = 1 \quad \forall \lambda > 0$$

The (tail) exponent  $\alpha$  is referred to as the tail index<sup>2</sup>. The main motivation for this

---

<sup>1</sup>The cumulative density function (cdf) is  $P(x)$ , which is the probability that random variable  $X$  is less than  $x$ . The complementary cumulative density function (ccdf) also referred to as the survival function is given by  $\bar{P}(x) = 1 - P(x)$

<sup>2</sup>The tail index is also referred to as shape parameter in the context of Pareto distributions, stability parameter in the context of stable distributions and degrees of freedom in the context of t-distributions.

article and the accompanying R ([R Core Team 2018](#)) package is a study in a forthcoming publication ([Munasinghe et al. 2019](#)) analyzing power law data, in particular that of Pareto distributions, a special class of power laws with probability density function<sup>3</sup> (pdf) defined for  $x \geq x_{\min} > 0$ :

$$p(x) = \frac{\alpha x_{\min}^{\alpha}}{x^{1+\alpha}} \quad (2)$$

Our team decided that it would be better to code our own parameter estimation tools in R with the (main) goal of identifying the tail index  $\alpha$ . We also found some of the existing tail estimation packages are not optimized for speed especially when dealing with a large data set. This resulted in the creation of our R-package **ptsuite**.

In this article our main consideration are Pareto distributions which are pure power law distributions as per the definition in equation (2). With regards to the general definition of power law distributions as per (1) we see there are three scenarios:

1. Right tails, i.e. power law behavior in the limit  $x \rightarrow \infty$ . The Pareto family is in this class.
2. Left tails, i.e. power law behavior in the limit  $x \rightarrow -\infty$
3. Both left and right tails, i.e. power law behavior in limit  $x \rightarrow \pm\infty$

Symmetric stable distributions and Student-T distributions fall into the last class. More general forms of stable distributions include a skew parameter that can result in exclusively left or right tails, for more details please refer to [Nolan \(2017\)](#).

## 2. Overview of Tail Index Estimation Methods

To identify Pareto behavior we can start with a a frequency plot/histogram and try to visually identify a lower bound (c.f.  $x_{\min}$ ) and ‘strict power law’ behavior. We may also do further heuristics such as log-log plots and log-log rank plots and look for a linear relationship to identify power law behavior, though these methods are not fool proof [Nair et al. \(2019\)](#). In [Hubert et al. \(2013\)](#) there is another Pareto test given which looks for linear behavior in the QQ-plots of the log transformed data against standard exponential data<sup>4</sup>. A simple implementation of this method is included in our package. We focus on the tail index estimators given below.

Estimation methods 1 - 5 estimate the tail index working under the assumption of Pareto distributed data. To apply these methods for the general power law form (1), we would look to identify where tail (power law) behavior starts which is not a precise or easy task ([Nair et al. 2019](#)). For a comprehensive list and the background

<sup>3</sup>The pdf is the derivative of the cdf:  $p(x) = \frac{dP(x)}{dx}$

<sup>4</sup>The slope of this line would give the  $\alpha$  estimate.

theory, the interested reader is referred to Hill (1975); Nair *et al.* (2019); Pokorna (2016); Hubert *et al.* (2013). The reference by Fedotenkov (2018) is a useful catalog of Pareto-tail index estimation techniques.

1. Maximum Likelihood Estimation (MLE) (Newman 2005)

The MLE formula leads a biased estimator  $\hat{\alpha}$ :

$$\hat{\alpha} = N \cdot \left[ \sum_{i=1}^N \frac{x_i}{\hat{x}_{\min}} \right]^{-1} \quad (3)$$

Here  $x_i$  represent the data point for  $i = 1, \dots, N$ . The minimum value  $x_{\min}$  is estimated from the data set and hence denoted  $\hat{x}_{\min}$ . As MLE leads to asymptotic normality for the estimator, the error bounds may be given by the standard deviation:

$$\hat{\sigma} = \frac{\sqrt{n+1}}{n} \cdot \hat{\alpha} \quad (4)$$

The estimate (3) can be converted to an unbiased version  $\alpha^*$  as follows Rizzo (2009):

$$\alpha^* = \frac{n-2}{n} \cdot \hat{\alpha}$$

2. Least Squares Estimation (LS) (Zaher *et al.* 2014; Nair *et al.* 2019)

The first step in this method is to sort the data in increasing order. Next for each value  $i$  (of  $N$  data points) we calculate  $y_i$  the number of points greater than the  $i^{\text{th}}$  data point. In this method one seeks to minimize the sum of the squared errors between the rank plot and the logarithm of the cdf. The estimator is given by

$$\hat{\alpha} = \frac{\sum_{i=1}^N \left( \hat{y}_i - \frac{1}{N} \sum_{i=1}^N \hat{y}_i \right) \left( \ln x_i - \frac{1}{N} \sum_{i=1}^N \ln x_i \right)}{\sum_{i=1}^N \left( \ln x_i - \frac{1}{N} \sum_{i=1}^N \ln x_i \right)^2} \quad (5)$$

3. Weighted Least Squares Estimation (WLS) (Nair *et al.* 2019)

The formulation is the same as for the LS method, except that the sum of squared errors is weighted. The choice of weight  $w_i$  is defined to be

$$w_i = \left[ \ln \left( \frac{x_i}{\hat{x}_{\min}} \right) \right]^{-1} \quad (6)$$

For this choice of weight the WLS method is closely related to the MLE, in that they will be converge in the large  $N$ -limit. The tail index estimate is then given by

$$\hat{\alpha} = \frac{-\sum_{i=1}^N \ln(\hat{y}_i/N)}{\sum_{i=1}^N \ln(x_i/\hat{x}_{\min})} \quad (7)$$

If there are no ties in the sorted data we may directly write:

$$\hat{\alpha} = \frac{-\sum_{i=1}^N \ln[(N+1-i)/N]}{\sum_{i=1}^N \ln(x_i/\hat{x}_{\min})} \quad (8)$$

4. Percentile Method (PM) (Bhatti *et al.* 2018)

The PM estimator for the tail index is given by

$$\hat{\alpha} = \frac{\ln 3}{\ln(P_{75}^*) - \ln(P_{25}^*)} \quad (9)$$

Here  $P_q^*$  is the  $q^{th}$  percentile of the data set. We also test a modified version using the median which we refer to here as the Modified Percentile Method (MPM). This is given by

$$\hat{\alpha} = \frac{\ln 2}{\ln(P_{75}^*) - \ln(P_{50}^*)} \quad (10)$$

The final modification we test makes use of the geometric mean and is written here in the shorthand Geometric Mean Percentile Method (GMPPM):

$$\hat{\alpha} = \frac{1 - \ln 4}{\frac{1}{N} \sum_{i=1}^N \ln x_i - \ln(P_{75}^*)} \quad (11)$$

There are further variations on PM estimation in Bhatti *et al.* (2018).

## 5. Method of Moments (MoM) (Rytgaard 1990),

The MoM estimator is derived by equating the sample mean to the theoretical mean of the Pareto distribution.

$$\hat{\alpha} = \frac{\sum_{i=1}^N x_i}{\sum_{i=1}^N x_i - N \cdot \hat{x}_{\min}} \quad (12)$$

The major drawback with this method is that if  $\alpha \leq 1$  the estimator will not converge to the true value - a Pareto distribution with  $\alpha \leq 1$  has infinite mean. (For this estimator  $\hat{\alpha} \rightarrow 1$  when  $\alpha \leq 1$  (Brazauskas and Serfling 2000)) It is always better to use this method in conjunction with another to check the validity of the estimate.

6. Hill Estimator (HE) (Nair *et al.* 2019; Pokorna 2016; Hill 1975)

The HE method is a type of MLE method (Hill 1975; Hubert *et al.* 2013; Fedotenkov 2018) which can be used for general power law setting (1) and one would need to specify where the tail starts. In particular the HE formula requires the specification of the tuning parameter (starting point)  $k$ . Identifying where the tail starts is an art in itself and requires a thorough examination of the data. If one chooses the parameter  $k$  to be too large the variance of the estimator increases; on the other hand if it is too low the bias increases (Fedotenkov 2018). The data first needs to be sorted in increasing order and then we may write:

$$\hat{\alpha}(k) = k \cdot \left[ \sum_{j=1}^k \ln \frac{x_{N-j+1}}{x_{N-k}} \right]^{-1} \quad (13)$$

We look to address the following with our estimator package:

1. **Accuracy** We examine how close the tail index estimates are on known tail index values corresponding to generated Pareto data. In particular we will also look at the size of the sample and the effect it has on accuracy.
2. **Speed** We want to examine how long the code takes to run. The effect of the sample size on the speed will also be examined.

### 3. Results for Accuracy

We begin by noting that we test the HE method separately as we have to supply an extra parameter, namely the tuning parameter  $k$ . To test each of the other six methods we do the following:

1. Simulate Pareto data for 4 different values of  $\alpha$ : 0.5, 1.5, 2.2, 5. Pareto data is generated via

$$x_i = \frac{x_{\min}}{u_i^{1/\alpha}}$$

where  $u_i$  is a standard uniform random number. We work with  $x_{\min} = 2$  for all the tests.

2. For each  $\alpha$  value we generate 3 different sample sizes:  $10^3, 10^5, 10^6$  to test the effect of sample size on the accuracy.
3. For each sample size we generate four different samples (by changing the random number generator seed).

In total we run  $288 = 6 \times 4 \times 3 \times 4$  tests. To economize on space we present a sample of the results (Table 1) with the complete set of results given in Appendix A. The  $\hat{\alpha}$  error given is calculated as follows:

$$\text{Error \%} = \frac{|\alpha - \hat{\alpha}|}{\alpha} \times 100\%$$

Method	Sample Size	$\hat{\alpha}$	$\hat{x}_{\min}$	$\hat{\alpha}$ error
MLE	1000	2.2592007	2.000197	2.69%
MLE	100000	2.1915330	2.000005	0.38%
MLE	1000000	2.1989831	2.000001	0.05%
LS	1000	2.2079391	2.000197	0.36%
LS	100000	2.1871973	2.000005	0.58%
LS	1000000	2.1998009	2.000001	0.01%
MoM	1000	2.2992300	2.000197	4.51%
MoM	100000	2.1897800	2.000005	0.46%
MoM	1000000	2.1998500	2.000001	0.01%
PM	1000	2.2464994	2.000197	2.11%
PM	100000	2.1965665	2.000005	0.16%
PM	1000000	2.1982344	2.000001	0.08%
MPM	1000	2.1701057	2.000197	1.36%
MPM	100000	2.1880345	2.000005	0.54%
MPM	1000000	2.1984458	2.000001	0.07%
GMPM	1000	2.2628257	2.000197	2.86%
GMPM	100000	2.2038589	2.000005	0.18%
GMPM	1000000	2.1994165	2.000001	0.03%
WLS	1000	2.2493215	2.000197	2.24%
WLS	100000	2.1913868	2.000005	0.39%
WLS	1000000	2.1989658	2.000001	0.05%

Table 1: Sample results for  $\alpha = 2.2$  (using sample 3 from each run).

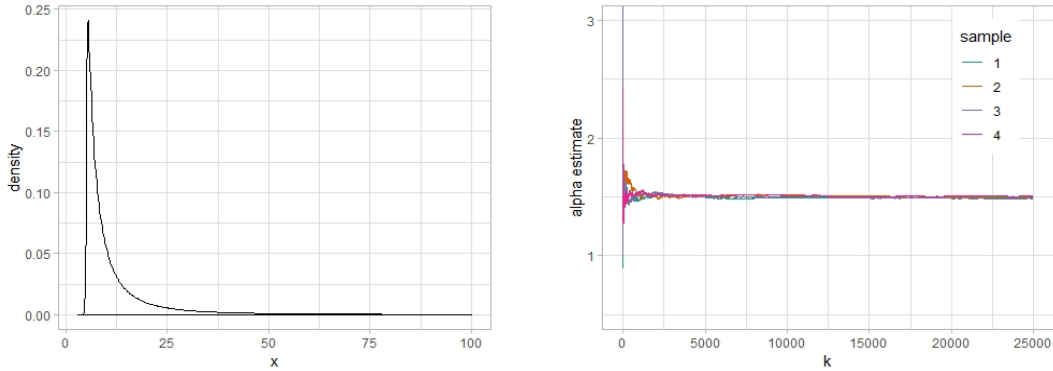
### 3.1. Accuracy of Hill's Estimator - Results

The HE method differs from the other methods in this paper as one needs to specify a cut-off point from which to consider the “tail” portion of the distribution. This is denoted by the  $k$  value which is the  $k^{th}$  largest observation of the data set. In other words, it can be thought of as the number of observations of the tail to be considered in estimating the shape parameter (tail index). We may also specify a value as a starting point of the tail estimation area instead of a ranked order.

### 3.2. Hill's Estimator for Strictly Pareto data

In this section we look at the impact of  $k$  when HE is used on strictly Pareto data. We consider the same random Pareto samples (with  $\alpha = 0.5, 1.5, 2.2$  and  $5.0$ ) used for the previous estimators but only consider  $10^3, 10^5$  and  $10^6$  sample sizes along with  $k$  set at the  $25^{th}, 50^{th}$  and  $75^{th}$  percentiles for each sample size. (Refer Appendix B for the full set of results.)

We also generate four samples from the Pareto distribution with  $\alpha = 1.5$  and  $\hat{x}_{\min} = 5$ . Figure 1b shows how the HE varies with  $k$ . Since the data supplied is strictly Pareto, increasing  $k$  (from 1 to  $n = 100,000$ ) increases the accuracy of the estimate and we can see the estimate stabilizing around  $\hat{\alpha} = 1.5$ .



(a) Density plot of Pareto distribution.

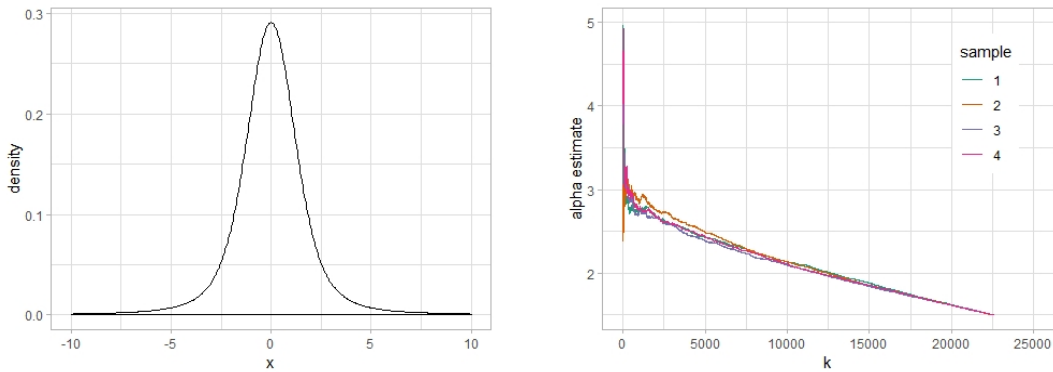
(b) Hill's estimate for  $\alpha$  with varying  $k$ .

Figure 1: Plot of pdf of Pareto distribution ( $\alpha = 1.5, \hat{x}_{\min} = 5$ ) and Hill's estimate for varying  $k$ .

### 3.3. Hill's Estimator for Non-Pareto data

To further test the accuracy of the HE method and understand the relationship of the value of  $k$  in estimating  $\alpha$ , we vary  $k$  in the presence of non-Pareto power law data. We consider data sampled from a i) t-distribution and ii) stable distribution respectively, and estimate the tail-index.

For this purpose we first generate four samples from the t-distribution with degrees of freedom (d.f.) equal to 3. The tail index  $\alpha$  is estimated by varying  $k$  from 1 to  $n$  (where  $n = 100,000$ ). It can be seen from Figure 2b that using a smaller value of  $k$  results in a more accurate estimation of the tail index - we feel this is because a smaller value of  $k$  avoids including observations from the “body” of the distribution. We should note however the earlier point about the bias of the estimator for small  $k$ .



(a) Density plot of t-distribution.

(b) Hill's estimate for  $\alpha$  with varying  $k$ .

Figure 2: Plot of pdf of t-distribution (d.f. = 3) and Hill's estimate for varying  $k$ .

Next we simulate four samples from a symmetric stable distribution<sup>5</sup> with stability parameter<sup>6</sup> (tail index)  $\alpha=1.5$  and vary  $k$  to estimate  $\alpha$ . The results of this simulation are similar to that of t-distribution in that a smaller value of  $k$  leads to a closer estimate of the true  $\alpha$ . The results are as follows:

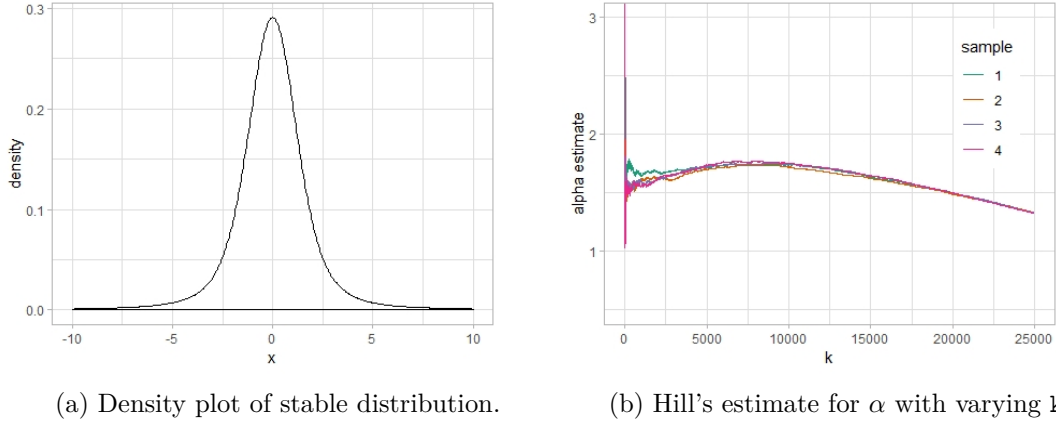


Figure 3: Plot of pdf of stable distribution ( $\alpha = 1.5$ ) and Hill's estimate for varying  $k$ .

## 4. Results for Speed

In this section we look at the performance of the functions available in the **ptsuite** package. All estimator functions have been coded in C++ to ensure the highest possible performance. In the initial comparison we compare the speed of the functions available in the package (using the package **microbenchmark** (Mersmann 2018)) to each other but do not include Hill's Estimator as it restricts its domain to a subset of the sample. The results of this comparison can be seen in Table 2.

The rank allocated here is the rank given to the particular function when compared to the other functions for a similar sample size.

We note here that the most computationally intensive functions are the LS and the WLS methods with the WLS method consuming a very large time for progressively larger datasets. It is seen that the PM and the MPM are consistently the fastest methods regardless of the sample size (but we remind the reader that the results of the percentile methods are also the least accurate for small sample sizes).

<sup>5</sup>A symmetric stable distribution was generated using  $\frac{\sin(\alpha V)}{(\cos(V))^{1/\alpha}} \left( \frac{\cos((1-\alpha)V)}{W} \right)^{(1-\alpha)/\alpha}$  where  $V \sim \text{Unif}[-\pi/2, \pi/2]$  and  $W \sim \text{Exp}(1)$  (Glasserman 2004, p.149).

<sup>6</sup>A symmetric stable distribution with stability parameter  $\alpha$  has the following property  $\lim_{x \rightarrow \pm\infty} p(x) \sim |x|^{1+\alpha}$ .



Method	Sample Size	Minimum ( $\mu s$ )	Median ( $\mu s$ )	Maximum ( $\mu s$ )	Rank
MLE	1000	23.61	24.46	57.74	3
MLE	10000	219.88	223.01	354.70	3
MLE	100000	2,171.16	2,226.06	2,627.13	3
MLE	1000000	21,939.19	22,337.42	23,294.00	3
MLE	10000000	221,566.50	230,232.30	250,296.00	3
LS	1000	54.61	56.32	85.33	7
LS	10000	543.86	592.21	680.96	7
LS	100000	5,558.61	5,817.88	18,289.77	7
LS	1000000	58,810.86	59,627.22	73,150.27	6
LS	10000000	627,379.60	679,164.70	753,455.90	6
WLS	1000	50.92	52.62	97.85	6
WLS	10000	516.84	539.02	750.08	6
WLS	100000	5,195.66	5,382.26	5,784.75	6
WLS	1000000	77,044.60	78,201.58	103,841.82	7
WLS	10000000	31,236,312.10	35,342,699.70	37,041,751.00	7
PM	1000	6.83	7.40	71.40	1
PM	10000	70.83	71.97	103.54	1
PM	100000	801.00	816.50	1,363.06	2
PM	1000000	9,372.44	9,660.87	11,815.25	1
PM	10000000	130,214.10	135,611.10	744,835.20	1
MPM	1000	7.11	7.54	81.64	2
MPM	10000	70.54	72.25	151.89	2
MPM	100000	801.57	814.65	1,105.35	1
MPM	1000000	9,348.55	9,664.28	24,424.95	2
MPM	10000000	130,062.50	135,871.10	188,462.30	2
GMPM	1000	29.01	30.15	112.07	5
GMPM	10000	292.69	295.40	661.33	5
GMPM	100000	3,028.48	3,071.72	4,194.99	5
GMPM	1000000	31,910.11	32,412.01	89,015.44	5
GMPM	10000000	355,401.90	371,501.40	434,037.60	5
MoM	1000	25.60	27.02	182.90	4
MoM	10000	238.93	244.34	343.61	4
MoM	100000	2,364.59	2,419.48	2,813.72	4
MoM	1000000	24,000.85	24,314.30	66,785.83	4
MoM	10000000	241,939.30	252,236.90	279,492.80	4

Table 2: Function timing results using **microbenchmark** for various sample sizes.

## 5. Using the Package

The **ptsuite** is built with ease of use in mind. Here we go over the use of each of the functions provided in the package.

### 5.1. The `generate_pareto` function

This function is able to generate random Pareto distributed data with the specified **shape** and **scale** parameters. The function has been written to be similar in type to the popular **runif** and **rexp** type of functions for generating data from a particular distribution.

The full call of the function is:

```
generate_pareto(sampleSize, shape, scale)
```

For example to generate a sample of size 100,000 with  $\alpha$  (shape parameter) = 1.2 and

$x_{min}$  (scale parameter) = 3 and store it in the variable `data`, the following code could be used producing the output as displayed below:

```
R> set.seed(1234)
R> d <- generate_pareto(100000, 1.2, 3)
R> head(d)

[1] 18.364397  4.454392  4.533604  4.447960  3.398763  4.349730
```

## 5.2. The `pareto_qq_test` function

The `pareto_qq_test` function can be used as a first step to identify whether the data is Pareto distributed before estimating the tail index. If most of the data points appear to be distributed along a line, it is possible that the data may be Pareto. Conversely, if most of the data are distributed non-linearly, then the data is most probably not Pareto distributed.

This function<sup>7</sup> plots the quantiles of the standard exponential distribution on the x-axis and the log values of the provided data on the y-axis. If Pareto data was supplied, a log transformation of this data would result in an exponential distribution with mean  $\alpha^{-1}$ . These data points would then show up on the QQ-plot as line with slope  $\alpha^{-1}$ .

**Note:** This is a heuristic test where one can look for data being distributed along a straight line indicating a possible Pareto distribution.

The full call of the function is:

```
pareto_qq_test(dat)
```

Demonstration of this function is carried out using a Pareto distributed sample of size 100,000 with  $\alpha$  (shape parameter) = 1.2 and  $x_{min}$  (scale parameter) = 3, and using an exponentially distributed sample of size 100,000 with  $\lambda$  (reciprocal of mean) = 5 as an example of non-Pareto data.

Q-Q Plot constructed for Pareto distributed data:

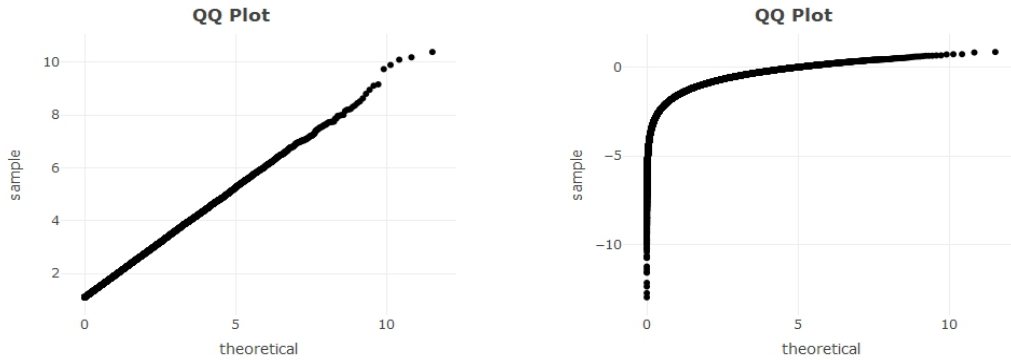
```
R> set.seed(1234)
R> d <- generate_pareto(100000, 1.2, 3)
R> pareto_qq_test(d)
```

Q-Q Plot constructed for exponentially distributed data:

```
R> set.seed(1234)
R> exp_data <- rexp(100000, 5)
R> pareto_qq_test(exp_data)
```

---

<sup>7</sup>For added interactivity this package makes use of the `plotly` package (Sievert 2018) to generate the Q-Q plots if it is available in the user's R library. If unavailable, it defaults to the R base plot.



(a) Q-Q Plot for Pareto distributed data. (b) Q-Q Plot for exponentially distributed data.

Figure 4: Outputs of `pareto_qq_test` function.

### 5.3. The `pareto_test` function

The `pareto_test` function can be used to identify whether the data is Pareto distributed (?). The test generates a p-value corresponding to the actual distribution of the data and is tested for significance. In the case of Pareto data, the p-value should be greater than the pre-determined significance level (generally taken as 0.05). In addition to using the function on Pareto data, we tested the function on selected non-Pareto (generated) data sets to ensure the test would reject those as Pareto. The tests were conducted on generated data sets of various sizes. The results obtained are shown below in Table 3.<sup>8</sup>

Distribution	Sample Size	p-value
Pareto (shape = 1.2, scale = 3)	100	0.5056804
	1000	0.1595162
	100000	0.849023
Exponential (rate = 5)	100	$4.76 \times 10^{-43}$
	1000	0
	100000	0
Binomial (n = 20, p = 0.6)	100	$1.06 \times 10^{-44}$
	1000	0
	100000	0
Half-normal	100	$1.37 \times 10^{-69}$
	1000	0
	100000	0

Table 3: Results of `pareto_test` function for various sample sizes.

The full call of the function is:

```
pareto_test(dat)
```

<sup>8</sup>In the table, half-normal distribution is taken to be the positive tail of the standard normal distribution excluding zeros.

Demonstration of this function is carried out using a Pareto distributed sample of size 100,000 with  $\alpha$  (shape parameter) = 1.2 and  $x_{min}$  (scale parameter) = 3, and using an exponentially distributed sample of size 100,000 with  $\lambda$  (reciprocal of mean) = 5 as an example of non-Pareto data.

```
R> set.seed(1234)
R> d <- generate_pareto(100000, 1.2, 3)
R> pareto_test(d)
```

```
$`p-value`
[1] 0.8604162
```

```
R> set.seed(1234)
R> exp_data <- rexp(100000, 5)
R> pareto_test(exp_data)
```

```
$`p-value`
[1] 0
```

#### 5.4. The `alpha_mle` function

This function can be used to estimate the shape parameter ( $\alpha$ ) using the Maximum Likelihood Estimator method - please refer equation (3). It can be used to obtain biased and unbiased estimates of the shape and scale parameters as well as the confidence interval for the shape parameter for the biased estimates.

The full call of the function is:

```
alpha_mle(dat, biased = TRUE, significance = NULL)
```

To demonstrate the various possibilities of this function we first generate Pareto data with  $\alpha = 1.2$  and  $x_{min} = 3$  and assign it to the variable `data`. `alpha_mle` is then used to obtain:

1. Biased Estimates for  $\alpha$  and  $x_{min}$
2. Unbiased Estimates for  $\alpha$  and  $x_{min}$
3. Biased Estimates for  $\alpha$  and  $x_{min}$  and the 95% confidence interval for  $\alpha$

R implementation of the above mentioned demonstration is given below.

Generating data from a Pareto Distribution:

```
R> set.seed(1234)
R> d <- generate_pareto(100000, 1.2, 3)
```

Obtaining the biased maximum likelihood estimator for  $\alpha$  and  $x_{min}$ :

```
R> alpha_mle(d)
```

```
$shape
[1] 1.201384
```

```
$scale
[1] 3.000005
```

Obtaining the unbiased maximum likelihood estimator for  $\alpha$  and  $x_{min}$ :

```
R> alpha_mle(d, FALSE)
```

```
$shape
[1] 1.201359
```

```
$scale
[1] 2.999998
```

Obtaining the biased maximum likelihood estimator for  $\alpha$  and  $x_{min}$  and the corresponding 95% confidence interval for  $\alpha$ :

```
R> alpha_mle(d, TRUE, 0.05)
```

```
$shape
[1] 1.201384
```

```
$lower_bound
[1] 1.193937
```

```
$upper_bound
[1] 1.20883
```

```
$scale
[1] 3.000005
```

### 5.5. The alpha\_hills function

The Hill's Estimator - please refer equation (13) - is particularly useful in the fact that it allows the specification of  $k$  as either the number of observations to be included in the tail or the minimum value to be considered in the tail of the distribution. When  $k=n$ , the Hill's Estimator returns the same estimate as `alpha_mle` with a warning notifying the user.

The full call of the function is:

```
alpha_hills(dat, k, value = FALSE)
```

We demonstrate the usage of the Hill's Estimator by generating Pareto data with  $\alpha=1.2$  and  $x_{min}=3$ , assigning it to the variable `data` and then using `alpha_hills`

R implementation of the above mentioned demonstration is given below with the respective outputs:

```
R> set.seed(1234)
R> d <- generate_pareto(100000, 1.2, 3)
R> alpha_hills(d, 8000, FALSE)
```

```
$shape
[1] 1.198717
```

```
$scale
[1] 24.54499
```

```
R> alpha_hills(d, 5000, TRUE)
```

```
$shape
[1] 1.125696
```

```
$scale
[1] 5002.76
```

```
R> alpha_hills(d, 100000, FALSE)
```

```
$shape
[1] 1.201384
```

```
$scale
[1] 3.000005
```

Warning message:

```
In alpha_hills(d, 1e+05, FALSE) :
Setting k as the number of observations makes it equivalent to the MLE
(alpha_mle function).
```

## 5.6. The `alpha_ls` function

This function can be used to estimate the shape parameter ( $\alpha$ ) using the Least Squares Estimator method - please refer equation (5).

The full call of the function is:

```
alpha_ls(dat)
```

We demonstrate the usage of the Least Squares Estimator by generating Pareto data with  $\alpha=1.2$  and  $x_{min}=3$ , assigning it to the variable `data` and then using `alpha_ls`.

R implementation of the above mentioned demonstration is given below with the respective output:

```
R> set.seed(1234)
R> d <- generate_pareto(100000, 1.2, 3)
R> alpha_ls(d)
```

```
$shape
[1] 1.20067
```

```
$scale
[1] 3.000005
```

### 5.7. The `alpha_percentile` function

This function can be used to estimate the shape parameter ( $\alpha$ ) using the Percentile Estimator method - please refer equation (9).

The full call of the function is:

```
alpha_percentile(dat)
```

We demonstrate the usage of the Percentile Estimator by generating Pareto data with  $\alpha=1.2$  and  $x_{min}=3$ , assigning it to the variable `data` and then using `alpha_percentile`.

R implementation of the above mentioned demonstration is given below with the respective output:

```
R> set.seed(1234)
R> d <- generate_pareto(100000, 1.2, 3)
R> alpha_percentile(d)
```

```
$shape
[1] 1.20048
```

```
$scale
[1] 3.000005
```

### 5.8. The `alpha_modified_percentile` function

This function can be used to estimate the shape parameter ( $\alpha$ ) using the Modified Percentile Estimator method - please refer equation (10).

The full call of the function is:

```
alpha_modified_percentile(dat)
```

We demonstrate the usage of the Modified Percentile Estimator by generating Pareto data with  $\alpha=1.2$  and  $x_{min}=3$ , assigning it to the variable `data` and then using `alpha_modified_percentile`.

R implementation of the above mentioned demonstration is given below with the respective output:

```
R> set.seed(1234)
R> d <- generate_pareto(100000, 1.2, 3)
R> alpha_modified_percentile(d)
```

```
$shape
[1] 1.196936
```

```
$scale
[1] 3.000005
```

### 5.9. The `alpha_geometric_percentile` function

This function can be used to estimate the shape parameter ( $\alpha$ ) using the Geometric Percentile Estimator method - please refer equation (11).

The full call of the function is:

```
alpha_geometric_percentile(dat)
```

We demonstrate the usage of the Geometric Percentile Estimator by generating Pareto data with  $\alpha=1.2$  and  $x_{min}=3$ , assigning it to the variable `data` and then using `alpha_geometric_percentile`.

R implementation of the above mentioned demonstration is given below with the respective output:

```
R> set.seed(1234)
R> d <- generate_pareto(100000, 1.2, 3)
R> alpha_geometric_percentile(d)
```

```
$shape
[1] 1.195801
```

```
$scale
[1] 3.000005
```



### 5.10. The `alpha_wls` function

This function can be used to estimate the shape parameter ( $\alpha$ ) using the Weighted Least Squares Estimator method - please refer to equations (7) and (8).

The full call of the function is:

```
alpha_wls(dat)
```

We demonstrate the usage of the Weighted Least Squares Estimator by generating Pareto data with  $\alpha=1.2$  and  $x_{min}=3$ , assigning it to the variable `data` and then using `alpha_wls`.

R implementation of the above mentioned demonstration is given below with the respective output:

```
R> set.seed(1234)
R> d <- generate_pareto(100000, 1.2, 3)
R> alpha_wls(d)
```

```
$shape
[1] 1.201303
```

```
$scale
[1] 3.000005
```

### 5.11. The `alpha_moment` function

This function can be used to estimate the shape parameter ( $\alpha$ ) using the Moment Estimator method - please refer equation (12).

The full call of the function is:

```
alpha_moment(dat)
```

We demonstrate the usage of the Moment Estimator by generating Pareto data with  $\alpha=1.2$  and  $x_{min}=3$ , assigning it to the variable `data` and then using `alpha_moment`.

R implementation of the above mentioned demonstration is given below with the respective output:

```
R> set.seed(1234)
R> d <- generate_pareto(100000, 1.2, 3)
R> alpha_moment(d)
```

```
$shape
[1] 1.241772
```

```
$scale
[1] 3.000005
```

### 5.12. The `generate_all_estimates` function

This function can be used to obtain estimates of the shape parameter ( $\alpha$ ) using all estimators except the HE.

The full call of the function is:

```
generate_all_estimates(dat)
```

We demonstrate the usage of the function by generating Pareto data with  $\alpha=1.2$  and  $x_{min}=3$ , assigning it to the variable `data` and then using `generate_all_estimates`.

R implementation of the above mentioned demonstration is given below with the respective output:

```
R> set.seed(1234)
R> d <- generate_pareto(100000, 1.2, 3)
R> generate_all_estimates(d)
```

	Method.of.Estimation	Shape.Parameter	Scale.Parameter
1	Maximum Likelihood Estimate	1.201384	3.000005
2	Least Squares	1.200670	3.000005
3	Method of Moments	1.241772	3.000005
4	Percentiles Method	1.200480	3.000005
5	Modified Percentiles Method	1.196936	3.000005
6	Geometric Percentiles Method	1.195801	3.000005
7	Weighted Least Squares	1.201303	3.000005

## 6. Summary and Future Work

Our R package **ptsuite** can be used in tail index estimation for Pareto distributed data. One may use the `pareto-qq-test` function as a first step to test (heuristically) if the data is Pareto. There are a number of methods which can be called on to estimate the tail index, and it is advisable to use more than one method as a comparison for the estimate. While we were confident of the accuracy of these well-known methods, we were concerned on run-times especially for large data samples. The functions on **ptsuite** performed reasonably well in the speed tests, where we tested data sets with  $10^7$  data points.

For future work we hope to add more tail estimation methods and functions found in the references to this package. In particular we hope to eventually incorporate the vast catalog of estimators in reference [Fedotenkov \(2018\)](#).

## Computational details

The results in this paper were obtained using R 3.5.2. and Microsoft R 3.5.1 (Microsoft and Team 2017)<sup>9</sup> with the **ptsuite** 1.0.0 package. R itself and all packages used are available from the Comprehensive R Archive Network (CRAN) at <https://CRAN.R-project.org/>. Microsoft R may be obtained from <https://mran.microsoft.com/open>.

RStudio V 1.1.423 (RStudio Team 2016) was used as the IDE for all tasks related to package development and generating results. The packages **devtools** (Wickham *et al.* 2018b) and **roxygen2** (Wickham *et al.* 2018a) were used to assist in package documentation and development. The **Rcpp** package (Eddelbuettel and François 2011; Eddelbuettel 2013; Eddelbuettel and Balamuta 2017) was used in the compilation of C++ code used in all estimation functions of the package.

**microbenchmark** 1.4-4, **ggplot2** 3.0.0 (Wickham 2016) and **plotly** 4.8.0 were used in generating the results included in this paper.

Two computers were used in the building of this package and the generation of the results. The technical specifications of the two computers are as follows:

Specification	Laptop	Desktop
<b>CPU</b>	Intel(R) Core(TM) i5-8250U @ 1.60GHz, 1800 Mhz, 4 Core(s), 8 Logical Processor(s)	Intel(R) Core(TM) i7-7700 @ 3.60GHz, 3601 Mhz, 4 Core(s), 8 Logical Processor(s)
<b>RAM</b>	8 GB	32 GB
<b>OS</b>	Windows 10 Enterprise LTSC 10.0.17763 Build 17763	Windows 10 Pro 10.0.17763 Build 17763

## Acknowledgments

We wish to acknowledge Dr. J. Nair for helpful advice and in providing unpublished material for our research and Prof. S. Gulati for sharing knowledge on the Goodness of Fit tests. D. J. would also like to acknowledge Prof. S. Banneheka for a very useful discussion on the WLS method of estimation.

## References

Bhatti SH, Hussain S, Ahmad T, Aslam M, Aftab M, Raza MA (2018). “Efficient estimation of Pareto model: Some modified percentile estimators.” *PLoS ONE*, **13**(5), 1–15.

<sup>9</sup>Microsoft R was used for the benchmark results of the speed and we note here that this version of R appeared to use slightly lower times in running the same tests.

- Brazauskas V, Serfling R (2000). “Robust and Efficient Estimation Of The Tail Index Of A Single-Parameter Pareto Distribution.” *North American Actuarial Journal*, **4**, 12–27.
- Eddelbuettel D (2013). *Seamless R and C++ Integration with Rcpp*. Springer, New York. doi:10.1007/978-1-4614-6868-4. ISBN 978-1-4614-6867-7.
- Eddelbuettel D, Balamuta JJ (2017). “Extending extitR with extitC++: A Brief Introduction to extitRcpp.” *PeerJ Preprints*, **5**, e3188v1. ISSN 2167-9843. doi:10.7287/peerj.preprints.3188v1. URL <https://doi.org/10.7287/peerj.preprints.3188v1>.
- Eddelbuettel D, François R (2011). “Rcpp: Seamless R and C++ Integration.” *Journal of Statistical Software*, **40**(8), 1–18. doi:10.18637/jss.v040.i08. URL <http://www.jstatsoft.org/v40/i08/>.
- Fedotenkov I (2018). “A review of more than one hundred Pareto-tail index estimators.” *MPRA Paper 90072*, University Library of Munich, Germany. URL <https://ideas.repec.org/p/pra/mprapa/90072.html>.
- Glasserman P (2004). *Monte Carlo Methods in Financial Engineering*. Springer, New York, NY, USA.
- Hill B (1975). “A Simple General Approach To Inference About The Tail Of A Distribution.” *The Annals of Statistics*, **3**(5), 1163–1174.
- Hubert M, Dierckx G, Vanpaemel D (2013). “Detecting Influential Data Point For The Hill Estimator In Pareto-Type Distributions.” *Computational Statistics & Data Analysis*, **65**, 13–28.
- Mersmann O (2018). *microbenchmark: Accurate Timing Functions*. R package version 1.4-4, URL <https://CRAN.R-project.org/package=microbenchmark>.
- Microsoft, Team RC (2017). *Microsoft R Open*. Microsoft, Redmond, Washington. URL <https://mran.microsoft.com/>.
- Munasinghe RM, Kossinna P, Wijeratne D, Jayasinghe D (2019). “In Preparation.” Fat Tails and Power Laws in Business Data.
- Nair J, Wierman A, Zwart B (2019). “The Fundamentals Of Heavy Tails: Properties, Emergence, And Identification.” <Http://users.cms.caltech.edu/~adamw/heavytails.html>.
- Newman MEJ (2005). “Power Laws, Pareto Distributions And Zipf’s Law.” *Contemporary Physics*, **46**, 323–351.
- Nolan JP (2017). “Stable Distributions: Models For Heavy Tailed Data.” <Http://fs2.american.edu/jpnolan/www/stable/chap1.pdf>.

- Pokorna M (2016). *Estimation and Application of the Tail Index*. Bachelor's thesis, Charles University in Prague, Faculty of Social Sciences, Institute of Economic Studies.
- R Core Team (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL <https://www.R-project.org/>.
- Rizzo ML (2009). "New Goodness-of-Fit Tests for Pareto Distributions." *ASTIN Bulletin: The Journal Of The IAA*, **39**(2), 691–715.
- RStudio Team (2016). *RStudio: Integrated Development Environment for R*. RStudio, Inc., Boston, MA. URL <http://www.rstudio.com/>.
- Rytgaard M (1990). "Estimation In The Pareto Distribution." *ASTIN Bulletin: The Journal Of The IAA*, **20**(2), 201–216.
- Sievert C (2018). *plotly for R*. URL <https://plotly-book.cpsievert.me>.
- Sornette D (2004). *Critical Phenomena In Natural Sciences, Chaos, Fractals, Selforganization And Disorder: Concepts And Tools*. Springer Series in Synergetics.
- Taleb NN (2019). *The Statistical Consequences Of Fat Tails*. Stem Academic Publishing.
- Wickham H (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. ISBN 978-3-319-24277-4. URL <http://ggplot2.org>.
- Wickham H, Danenberg P, Eugster M (2018a). *roxygen2: In-Line Documentation for R*. R package version 6.1.1, URL <https://CRAN.R-project.org/package=roxygen2>.
- Wickham H, Hester J, Chang W (2018b). *devtools: Tools to Make Developing R Packages Easier*. R package version 2.0.1, URL <https://CRAN.R-project.org/package=devtools>.
- Zaher HM, El-Sheik AA, El-Magd NATA (2014). "Estimation of Pareto Parameters Using a Fuzzy Least-Squares Method and Other Known Techniques with a Comparison." *British Journal of Mathematics & Computer Science*, **4**(14), 2067–2088.

## A. Complete Tables of Estimates

Method	Sample Size	Sample	$\hat{\alpha}$	$\hat{x}_{\min}$	$\hat{\alpha}$ error
MLE	1000	1	5.0652128	2.000028	1.30%
		2	4.8768226	2.000292	2.46%
		3	5.1345472	2.000087	2.69%
		4	4.7497352	2.000301	5.01%
	100000	1	4.9810971	2.000021	0.38%
		2	5.0083374	2.000011	0.17%
		3	4.9807569	2.000002	0.38%
		4	5.0034707	2.000001	0.07%
	1000000	1	4.9993293	2.000001	0.01%
		2	5.0041973	2.000000	0.08%
		3	4.9976888	2.000000	0.05%
		4	4.9974922	2.000000	0.05%
LS	1000	1	5.0476293	2.000028	0.95%
		2	4.7059973	2.000292	5.88%
		3	5.0180434	2.000087	0.36%
		4	4.5705256	2.000301	8.59%
	100000	1	4.9668545	2.000021	0.66%
		2	4.9963066	2.000011	0.07%
		3	4.9709030	2.000002	0.58%
		4	5.0056427	2.000001	0.11%
	1000000	1	5.0012983	2.000001	0.03%
		2	5.0005891	2.000000	0.01%
		3	4.9995475	2.000000	0.01%
		4	4.9986945	2.000000	0.03%
MoM	1000	1	5.1021780	2.000028	2.04%
		2	4.8865050	2.000292	2.27%
		3	5.1537270	2.000087	3.07%
		4	4.7433810	2.000301	5.13%
	100000	1	4.9795270	2.000021	0.41%
		2	5.0083190	2.000011	0.17%
		3	4.9799920	2.000002	0.40%
		4	5.0053840	2.000001	0.11%
	1000000	1	4.9999360	2.000001	0.00%
		2	5.0037260	2.000000	0.07%
		3	4.9984560	2.000000	0.03%
		4	4.9980840	2.000000	0.04%
PM	1000	1	5.1654072	2.000028	3.31%
		2	4.5323446	2.000292	9.35%
		3	5.1056805	2.000087	2.11%
		4	4.8577159	2.000301	2.85%
	100000	1	4.9574049	2.000021	0.85%
		2	4.9728086	2.000011	0.54%
		3	4.9921967	2.000002	0.16%
		4	4.9938595	2.000001	0.12%
	1000000	1	4.9949127	2.000001	0.10%
		2	4.9947259	2.000000	0.11%
		3	4.9959874	2.000000	0.08%
		4	4.9975729	2.000000	0.05%

Table 4 – Continued on next page

Method	Sample Size	Sample	$\hat{\alpha}$	$\hat{x}_{\min}$	$\hat{\alpha}$ error
MPM	1000	1	5.5241262	2.000028	10.48%
		2	4.4701475	2.000292	10.60%
		3	4.9320572	2.000087	1.36%
		4	5.0637775	2.000301	1.28%
	100000	1	4.9419910	2.000021	1.16%
		2	4.9524059	2.000011	0.95%
		3	4.9728058	2.000002	0.54%
		4	4.9907899	2.000001	0.18%
	1000000	1	4.9857602	2.000001	0.28%
		2	4.9911313	2.000000	0.18%
		3	4.9964677	2.000000	0.07%
		4	4.9969832	2.000000	0.06%
GMPM	1000	1	5.2572589	2.000028	5.15%
		2	4.2810147	2.000292	14.38%
		3	5.1427862	2.000087	2.86%
		4	5.1208210	2.000301	2.42%
	100000	1	4.9431381	2.000021	1.14%
		2	4.9554940	2.000011	0.89%
		3	5.0087703	2.000002	0.18%
		4	4.9784400	2.000001	0.43%
	1000000	1	4.9854361	2.000001	0.29%
		2	4.9928066	2.000000	0.14%
		3	4.9986738	2.000000	0.03%
		4	4.9955151	2.000000	0.09%
WLS	1000	1	5.0430631	2.000028	0.86%
		2	4.8554967	2.000292	2.89%
		3	5.1120943	2.000087	2.24%
		4	4.7289651	2.000301	5.42%
	100000	1	4.9807646	2.000021	0.38%
		2	5.0080031	2.000011	0.16%
		3	4.9804244	2.000002	0.39%
		4	5.0031367	2.000001	0.06%
	1000000	1	4.9992902	2.000001	0.01%
		2	5.0041581	2.000000	0.08%
		3	4.9976496	2.000000	0.05%
		4	4.9974531	2.000000	0.05%

Table 4: Sample results for  $\alpha = 5.0$  (for all samples).

Method	Sample Size	Sample	$\hat{\alpha}$	$\hat{x}_{\min}$	$\hat{\alpha}$ error
MLE	1000	1	2.2286936	2.000063	1.30%
		2	2.1458019	2.000664	2.46%
		3	2.2592007	2.000197	2.69%
		4	2.0898835	2.000683	5.01%
	100000	1	2.1916827	2.000048	0.38%
		2	2.2036685	2.000025	0.17%
		3	2.1915330	2.000005	0.38%
		4	2.2015271	2.000002	0.07%
	1000000	1	2.1997049	2.000001	0.01%
		2	2.2018468	2.000000	0.08%
		3	2.1989831	2.000001	0.05%
		4	2.1988966	2.000000	0.05%
LS	1000	1	2.2209569	2.000063	0.95%
		2	2.0706388	2.000664	5.88%
		3	2.2079391	2.000197	0.36%
		4	2.0110313	2.000683	8.59%
	100000	1	2.1854160	2.000048	0.66%
		2	2.1983749	2.000025	0.07%
		3	2.1871973	2.000005	0.58%
		4	2.2024828	2.000002	0.11%
	1000000	1	2.2005712	2.000001	0.03%
		2	2.2002592	2.000000	0.01%
		3	2.1998009	2.000001	0.01%
		4	2.1994256	2.000000	0.03%
MoM	1000	1	2.2816390	2.000063	3.71%
		2	2.1801250	2.000664	0.90%
		3	2.2992300	2.000197	4.51%
		4	2.0812090	2.000683	5.40%
	100000	1	2.1904890	2.000048	0.43%
		2	2.2085240	2.000025	0.39%
		3	2.1897800	2.000005	0.46%
		4	2.2048300	2.000002	0.22%
	1000000	1	2.1994210	2.000001	0.03%
		2	2.2021380	2.000000	0.10%
		3	2.1998500	2.000001	0.01%
		4	2.2005870	2.000000	0.03%
PM	1000	1	2.2727792	2.000063	3.31%
		2	1.9942276	2.000664	9.35%
		3	2.2464994	2.000197	2.11%
		4	2.1373945	2.000683	2.85%
	100000	1	2.1812582	2.000048	0.85%
		2	2.1880358	2.000025	0.54%
		3	2.1965665	2.000005	0.16%
		4	2.1972982	2.000002	0.12%
	1000000	1	2.1977616	2.000001	0.10%
		2	2.1976794	2.000000	0.11%
		3	2.1982344	2.000001	0.08%
		4	2.1989321	2.000000	0.05%

Table 5 – Continued on next page



Method	Sample Size	Sample	$\hat{\alpha}$	$\hat{x}_{\min}$	$\hat{\alpha}$ error
MPM	1000	1	2.4306180	2.000063	10.48%
		2	1.9668589	2.000664	10.60%
		3	2.1701057	2.000197	1.36%
		4	2.2280617	2.000683	1.28%
	100000	1	2.1744760	2.000048	1.16%
		2	2.1790586	2.000025	0.95%
		3	2.1880345	2.000005	0.54%
		4	2.1959476	2.000002	0.18%
	1000000	1	2.1937345	2.000001	0.28%
		2	2.1960978	2.000000	0.18%
		3	2.1984458	2.000001	0.07%
		4	2.1986726	2.000000	0.06%
GMPM	1000	1	2.3131938	2.000063	5.15%
		2	1.8836362	2.000664	14.38%
		3	2.2628257	2.000197	2.86%
		4	2.2531594	2.000683	2.42%
	100000	1	2.1749808	2.000048	1.14%
		2	2.1804174	2.000025	0.89%
		3	2.2038589	2.000005	0.18%
		4	2.1905136	2.000002	0.43%
	1000000	1	2.1935919	2.000001	0.29%
		2	2.1968349	2.000000	0.14%
		3	2.1994165	2.000001	0.03%
		4	2.1980267	2.000000	0.09%
WLS	1000	1	2.2189478	2.000063	0.86%
		2	2.1364185	2.000664	2.89%
		3	2.2493215	2.000197	2.24%
		4	2.0807446	2.000683	5.42%
	100000	1	2.1915364	2.000048	0.38%
		2	2.2035214	2.000025	0.16%
		3	2.1913868	2.000005	0.39%
		4	2.2013802	2.000002	0.06%
	1000000	1	2.1996877	2.000001	0.01%
		2	2.2018296	2.000000	0.08%
		3	2.1989658	2.000001	0.05%
		4	2.1988794	2.000000	0.05%

Table 5: Sample results for  $\alpha = 2.2$  (for all samples).

Method	Sample Size	Sample	$\hat{\alpha}$	$\hat{x}_{\min}$	$\hat{\alpha}$ error
MLE	1000	1	1.5195638	2.000093	1.30%
		2	1.4630468	2.000974	2.46%
		3	1.5403641	2.000289	2.69%
		4	1.4249206	2.001002	5.01%
	100000	1	1.4943291	2.000071	0.38%
		2	1.5025012	2.000037	0.17%
		3	1.4942271	2.000007	0.38%
		4	1.5010412	2.000002	0.07%
	1000000	1	1.4997988	2.000002	0.01%
		2	1.5012592	2.000001	0.08%
		3	1.4993066	2.000001	0.05%
		4	1.4992477	2.000000	0.05%
LS	1000	1	1.5142888	2.000093	0.95%
		2	1.4117992	2.000974	5.88%
		3	1.5054130	2.000289	0.36%
		4	1.3711577	2.001002	8.59%
	100000	1	1.4900564	2.000071	0.66%
		2	1.4988920	2.000037	0.07%
		3	1.4912709	2.000007	0.58%
		4	1.5016928	2.000002	0.11%
	1000000	1	1.5003895	2.000002	0.03%
		2	1.5001767	2.000001	0.01%
		3	1.4998643	2.000001	0.01%
		4	1.4996084	2.000000	0.03%
MoM	1000	1	1.5974170	2.000093	6.49%
		2	1.5323020	2.000974	2.15%
		3	1.6119070	2.000289	7.46%
		4	1.4335130	2.001002	4.43%
	100000	1	1.4960330	2.000071	0.26%
		2	1.5201780	2.000037	1.35%
		3	1.4872030	2.000007	0.85%
		4	1.5105550	2.000002	0.70%
	1000000	1	1.4956750	2.000002	0.29%
		2	1.5064640	2.000001	0.43%
		3	1.5007780	2.000001	0.05%
		4	1.5062520	2.000000	0.42%
PM	1000	1	1.5496222	2.000093	3.31%
		2	1.3596983	2.000974	9.35%
		3	1.5317042	2.000289	2.11%
		4	1.4573142	2.001002	2.85%
	100000	1	1.4872215	2.000071	0.85%
		2	1.4918426	2.000037	0.54%
		3	1.4976590	2.000007	0.16%
		4	1.4981578	2.000002	0.12%
	1000000	1	1.4984738	2.000002	0.10%
		2	1.4984178	2.000001	0.11%
		3	1.4987962	2.000001	0.08%
		4	1.4992719	2.000000	0.05%

Table 6 – Continued on next page

Method	Sample Size	Sample	$\hat{\alpha}$	$\hat{x}_{\min}$	$\hat{\alpha}$ error
MPM	1000	1	1.6572410	2.000093	10.48%
		2	1.3410367	2.000974	10.60%
		3	1.4796178	2.000289	1.36%
		4	1.5191328	2.001002	1.28%
	100000	1	1.4825973	2.000071	1.16%
		2	1.4857218	2.000037	0.95%
		3	1.4918417	2.000007	0.54%
		4	1.4972370	2.000002	0.18%
	1000000	1	1.4957281	2.000002	0.28%
		2	1.4973394	2.000001	0.18%
		3	1.4989403	2.000001	0.07%
		4	1.4990950	2.000000	0.06%
GMPM	1000	1	1.5771776	2.000093	5.15%
		2	1.2842916	2.000974	14.38%
		3	1.5428356	2.000289	2.86%
		4	1.5362441	2.001002	2.42%
	100000	1	1.4829414	2.000071	1.14%
		2	1.4866482	2.000037	0.89%
		3	1.5026311	2.000007	0.18%
		4	1.4935320	2.000002	0.43%
	1000000	1	1.4956308	2.000002	0.29%
		2	1.4978420	2.000001	0.14%
		3	1.4996021	2.000001	0.03%
		4	1.4986545	2.000000	0.09%
WLS	1000	1	1.5129189	2.000093	0.86%
		2	1.4566490	2.000974	2.89%
		3	1.5336283	2.000289	2.24%
		4	1.4186895	2.001002	5.42%
	100000	1	1.4942294	2.000071	0.38%
		2	1.5024009	2.000037	0.16%
		3	1.4941273	2.000007	0.39%
		4	1.5009410	2.000002	0.06%
	1000000	1	1.4997871	2.000002	0.01%
		2	1.5012474	2.000001	0.08%
		3	1.4992949	2.000001	0.05%
		4	1.4992359	2.000000	0.05%

Table 6: Sample results for  $\alpha = 1.5$  (for all samples).

Method	Sample Size	Sample	$\hat{\alpha}$	$\hat{x}_{\min}$	$\hat{\alpha}$ error
MLE	1000	1	0.5065213	2.000278	1.30%
		2	0.4876823	2.002923	2.46%
		3	0.5134547	2.000866	2.69%
		4	0.4749735	2.003009	5.01%
	100000	1	0.4981097	2.000212	0.38%
		2	0.5008337	2.000112	0.17%
		3	0.4980757	2.000021	0.38%
		4	0.5003471	2.000007	0.07%
	1000000	1	0.4999329	2.000007	0.01%
		2	0.5004197	2.000002	0.08%
		3	0.4997689	2.000004	0.05%
		4	0.4997492	2.000000	0.05%
LS	1000	1	0.5047629	2.000278	0.95%
		2	0.4705997	2.002923	5.88%
		3	0.5018043	2.000866	0.36%
		4	0.4570526	2.003009	8.59%
	100000	1	0.4966855	2.000212	0.66%
		2	0.4996307	2.000112	0.07%
		3	0.4970903	2.000021	0.58%
		4	0.5005643	2.000007	0.11%
	1000000	1	0.5001298	2.000007	0.03%
		2	0.5000589	2.000002	0.01%
		3	0.4999548	2.000004	0.01%
		4	0.4998695	2.000000	0.03%
MoM	1000	1	1.0007560	2.000278	100.15%
		2	1.0003550	2.002923	100.07%
		3	1.0014970	2.000866	100.30%
		4	1.0000360	2.003009	100.01%
	100000	1	1.0000010	2.000212	100.00%
		2	1.0000100	2.000112	100.00%
		3	1.0000000	2.000021	100.00%
		4	1.0000030	2.000007	100.00%
	1000000	1	1.0000000	2.000007	100.00%
		2	1.0000000	2.000002	100.00%
		3	1.0000000	2.000004	100.00%
		4	1.0000010	2.000000	100.00%
PM	1000	1	0.5165408	2.000278	3.31%
		2	0.4532280	2.002923	9.35%
		3	0.5105681	2.000866	2.11%
		4	0.4857708	2.003009	2.85%
	100000	1	0.4957405	2.000212	0.85%
		2	0.4972809	2.000112	0.54%
		3	0.4992197	2.000021	0.16%
		4	0.4993859	2.000007	0.12%
	1000000	1	0.4994913	2.000007	0.10%
		2	0.4994726	2.000002	0.11%
		3	0.4995987	2.000004	0.08%
		4	0.4997573	2.000000	0.05%

Table 7 – Continued on next page

Method	Sample Size	Sample	$\hat{\alpha}$	$\hat{x}_{\min}$	$\hat{\alpha}$ error
MPM	1000	1	0.5524166	2.000278	10.48%
		2	0.4470051	2.002923	10.60%
		3	0.4932065	2.000866	1.36%
		4	0.5063772	2.003009	1.28%
	100000	1	0.4941991	2.000212	1.16%
		2	0.4952406	2.000112	0.95%
		3	0.4972806	2.000021	0.54%
		4	0.4990790	2.000007	0.18%
	1000000	1	0.4985760	2.000007	0.28%
		2	0.4991131	2.000002	0.18%
		3	0.4996468	2.000004	0.07%
		4	0.4996983	2.000000	0.06%
GMPM	1000	1	0.5257258	2.000278	5.15%
		2	0.4280850	2.002923	14.38%
		3	0.5142783	2.000866	2.86%
		4	0.5120792	2.003009	2.42%
	100000	1	0.4943138	2.000212	1.14%
		2	0.4955494	2.000112	0.89%
		3	0.5008770	2.000021	0.18%
		4	0.4978440	2.000007	0.43%
	1000000	1	0.4985436	2.000007	0.29%
		2	0.4992807	2.000002	0.14%
		3	0.4998674	2.000004	0.03%
		4	0.4995515	2.000000	0.09%
WLS	1000	1	0.5043063	2.000278	0.86%
		2	0.4855497	2.002923	2.89%
		3	0.5112094	2.000866	2.24%
		4	0.4728965	2.003009	5.42%
	100000	1	0.4980765	2.000212	0.38%
		2	0.5008003	2.000112	0.16%
		3	0.4980424	2.000021	0.39%
		4	0.5003137	2.000007	0.06%
	1000000	1	0.4999290	2.000007	0.01%
		2	0.5004158	2.000002	0.08%
		3	0.4997650	2.000004	0.05%

Table 7: Sample results for  $\alpha = 0.5$  (for all samples).

**B. Hill's Estimates**

Sample Size	k	Sample	$\hat{\alpha}$	$\hat{x}_{\min}$	$\hat{\alpha}$ error
1000	250	1	5.1384879	2.622409	2.77%
		2	5.0187719	2.688776	0.38%
		3	5.1645221	2.619874	3.29%
		4	4.7188721	2.662987	5.62%
	500	1	5.358747	2.314142	7.17%
		2	4.7658326	2.301751	4.68%
		3	4.9729643	2.276799	0.54%
		4	4.84922	2.32247	3.02%
	750	1	5.1040343	2.120318	2.08%
		2	4.7126907	2.108961	5.75%
		3	5.0974016	2.112979	1.95%
		4	4.7370529	2.124039	5.26%
100000	25000	1	4.9723687	2.643396	0.55%
		2	5.0014969	2.639987	0.03%
		3	4.9550096	2.640709	0.90%
		4	5.0164426	2.639534	0.33%
	50000	1	4.9599243	2.297478	0.80%
		2	4.9784265	2.295197	0.43%
		3	4.9626286	2.297129	0.75%
		4	5.0032577	2.297264	0.07%
	75000	1	4.968077	2.117967	0.64%
		2	4.9870807	2.116682	0.26%
		3	4.9819341	2.119073	0.36%
		4	5.0027546	2.118298	0.06%
1000000	250000	1	5.0046153	2.639683	0.09%
		2	5.0012057	2.638868	0.02%
		3	4.9950568	2.639315	0.10%
		4	4.9990826	2.639467	0.02%
	500000	1	4.9958074	2.297068	0.08%
		2	4.9965736	2.296702	0.07%
		3	4.9951842	2.297432	0.10%
		4	4.9973861	2.297595	0.05%
	750000	1	5.0002598	2.11851	0.01%
		2	4.9976201	2.117838	0.05%
		3	4.9951151	2.118314	0.10%
		4	4.9987034	2.118581	0.03%

Table 8: Sample results for  $\alpha = 5.0$  (Hill's Estimate).

Sample Size	k	Sample	$\hat{\alpha}$	$\hat{x}_{\min}$	$\hat{\alpha}$ error
1000	250	1	2.2609347	3.702224	2.77%
		2	2.2082596	3.918605	0.38%
		3	2.2723897	3.694097	3.29%
		4	2.0763037	3.833704	5.62%
	500	1	2.3578487	2.786314	7.17%
		2	2.0969663	2.752523	4.68%
		3	2.1881043	2.685175	0.54%
		4	2.1336568	2.809155	3.02%

Sample Size	k	Sample	$\hat{\alpha}$	$\hat{x}_{\min}$	$\hat{\alpha}$ error
100000	750	1	2.2457751	2.283975	2.08%
		2	2.0735839	2.256266	5.75%
		3	2.2428567	2.266049	1.95%
		4	2.0843033	2.293095	5.26%
	25000	1	2.1878422	3.769907	0.55%
		2	2.2006586	3.758865	0.03%
		3	2.1802042	3.761201	0.90%
		4	2.2072347	3.7574	0.33%
	50000	1	2.1823667	2.740923	0.80%
		2	2.1905077	2.734741	0.43%
		3	2.1835566	2.739976	0.75%
		4	2.2014334	2.740342	0.07%
75000	1	2.1859539	2.278224	0.64%	
	2	2.1943155	2.275084	0.26%	
	3	2.192051	2.280928	0.36%	
	4	2.201212	2.279033	0.06%	
1000000	250000	1	2.2020307	3.757883	0.09%
		2	2.2005305	3.755245	0.02%
		3	2.197825	3.756693	0.10%
		4	2.1995964	3.757182	0.02%
	500000	1	2.1981553	2.739811	0.08%
		2	2.1984924	2.738818	0.07%
		3	2.197881	2.740798	0.10%
		4	2.1988499	2.741238	0.05%
	750000	1	2.2001143	2.279551	0.01%
		2	2.1989528	2.277909	0.05%
		3	2.1978506	2.279071	0.10%
		4	2.1994295	2.279726	0.03%

Table 9: Sample results for  $\alpha = 2.2$  (Hill's Estimate).

Sample Size	k	Sample	$\hat{\alpha}$	$\hat{x}_{\min}$	$\hat{\alpha}$ error
1000	250	1	1.5415464	4.934745	2.77%
		2	1.5056316	5.363466	0.38%
		3	1.5493566	4.918864	3.29%
		4	1.4156616	5.193897	5.62%
	500	1	1.6076241	3.252596	7.17%
		2	1.4297498	3.194905	4.68%
		3	1.4918893	3.08091	0.54%
		4	1.454766	3.291777	3.02%
	750	1	1.5312103	2.429964	2.08%
		2	1.4138072	2.38685	5.75%
		3	1.5292205	2.402043	1.95%
		4	1.4211159	2.444209	5.26%
25000	1	1.4917106	5.067623	0.55%	
	2	1.5004491	5.045869	0.03%	
	3	1.4865029	5.050468	0.90%	
	4	1.5049328	5.042983	0.33%	

Table 10 – Continued on next page

100000

Sample Size	k	Sample	$\hat{\alpha}$	$\hat{x}_{\min}$	$\hat{\alpha}$ error	
	50000	1	1.4879773	3.175178	0.80%	
		2	1.493528	3.16468	0.43%	
		3	1.4887886	3.173568	0.75%	
		4	1.5009773	3.174191	0.07%	
	75000	1	1.4904231	2.420997	0.64%	
		2	1.4961242	2.416103	0.26%	
		3	1.4945802	2.425212	0.36%	
		4	1.5008264	2.422256	0.06%	
	1000000	250000	1	1.5013846	5.043935	0.09%
			2	1.5003617	5.038743	0.02%
			3	1.4985171	5.041592	0.10%
			4	1.4997248	5.042554	0.02%
500000		1	1.4987422	3.173289	0.08%	
		2	1.4989721	3.171602	0.07%	
		3	1.4985553	3.174965	0.10%	
		4	1.4992158	3.175713	0.05%	
750000		1	1.5000779	2.423064	0.01%	
		2	1.499286	2.420505	0.05%	
		3	1.4985345	2.422317	0.10%	
		4	1.499611	2.423337	0.03%	

Table 10: Sample results for  $\alpha = 1.5$  (Hill's Estimate).

Sample Size	k	Sample	$\hat{\alpha}$	$\hat{x}_{\min}$	$\hat{\alpha}$ error
1000	250	1	0.5138488	30.042363	2.77%
		2	0.5018772	38.572395	0.38%
		3	0.5164522	29.753256	3.29%
		4	0.4718872	35.028376	5.62%
	500	1	0.5358747	8.602613	7.17%
		2	0.4765833	8.152935	4.68%
		3	0.4972964	7.311007	0.54%
		4	0.484922	8.917257	3.02%
	750	1	0.5104034	3.587069	2.08%
		2	0.4712691	3.399504	5.75%
		3	0.5097402	3.464835	1.95%
		4	0.4737053	3.65052	5.26%
100000	25000	1	0.4972369	32.535157	0.55%
		2	0.5001497	32.11795	0.03%
		3	0.495501	32.20586	0.90%
		4	0.5016443	32.062887	0.33%
	50000	1	0.4959924	8.002846	0.80%
		2	0.4978427	7.923726	0.43%
		3	0.4962629	7.990674	0.75%
		4	0.5003258	7.995384	0.07%
	75000	1	0.4968077	3.547502	0.64%
		2	0.4987081	3.526034	0.26%
		3	0.4981934	3.566063	0.36%
		4	0.5002755	3.553042	0.06%

Table 11 – Continued on next page



Sample Size	k	Sample	$\hat{\alpha}$	$\hat{x}_{\min}$	$\hat{\alpha}$ error
1000000	250000	1	0.5004615	32.081033	0.09%
		2	0.5001206	31.982071	0.02%
		3	0.4995057	32.036353	0.10%
		4	0.4999083	32.054697	0.02%
	500000	1	0.4995807	7.988564	0.08%
		2	0.4996574	7.975831	0.07%
		3	0.4995184	8.001228	0.10%
		4	0.4997386	8.006891	0.05%
	750000	1	0.500026	3.556597	0.01%
		2	0.499762	3.545339	0.05%
		3	0.4995115	3.553309	0.10%
		4	0.4998703	3.557801	0.03%

Table 11: Sample results for  $\alpha = 0.5$  (Hill's Estimate).

**Affiliation:**

Ranjiva Munasinghe  
 CEO  
 MIND Analytics & Management (Pvt) Ltd  
 10/1 De Fonseka Place  
 Colombo 5  
 Sri Lanka  
 E-mail: [ranjiva@mintlanka.org](mailto:ranjiva@mintlanka.org)  
 URL: [www.mind-am.org](http://www.mind-am.org)