weisimu: Using Simulations to Assess the Performance of the Truncated Mean of a Sample Generated from a Weibull Distribution

Xiaoya Wang, Bertrand Sodjahin, and Gradon Nicholls

2023-06-20

weisimu stands for **Wei**bull **simu**lation. The package allows to conduct a comparative simulation study of trimmed sample mean vs. untrimmed sample mean for the Weibull distribution. The current vignette includes help documentation for the usage of **weisimu**, along with some simulation illustration examples.

The Weibull Distribution

$X \sim Wei(\theta, \beta)$:

$$f(x;\theta,\beta) = \frac{\beta}{\theta}\left(\frac{x}{\theta}\right)^{\beta-1} e^{-(x/\theta)^{\beta}}$$

- θ: scale parameter
- β: shape parameter

Pre-requisites

Install the package if needed and load it:

```
if (!require("devtools")) install.packages("devtools")
#> Loading required package: devtools
#> Loading required package: usethis
if (!require("weisimu")) devtools::install_github("Dorayaya/weisimu")
#> Loading required package: weisimu
#>
#> Attaching package: 'weisimu'
#> The following object is masked from 'package:base':
#>
#>     mean
library(weisimu)
```

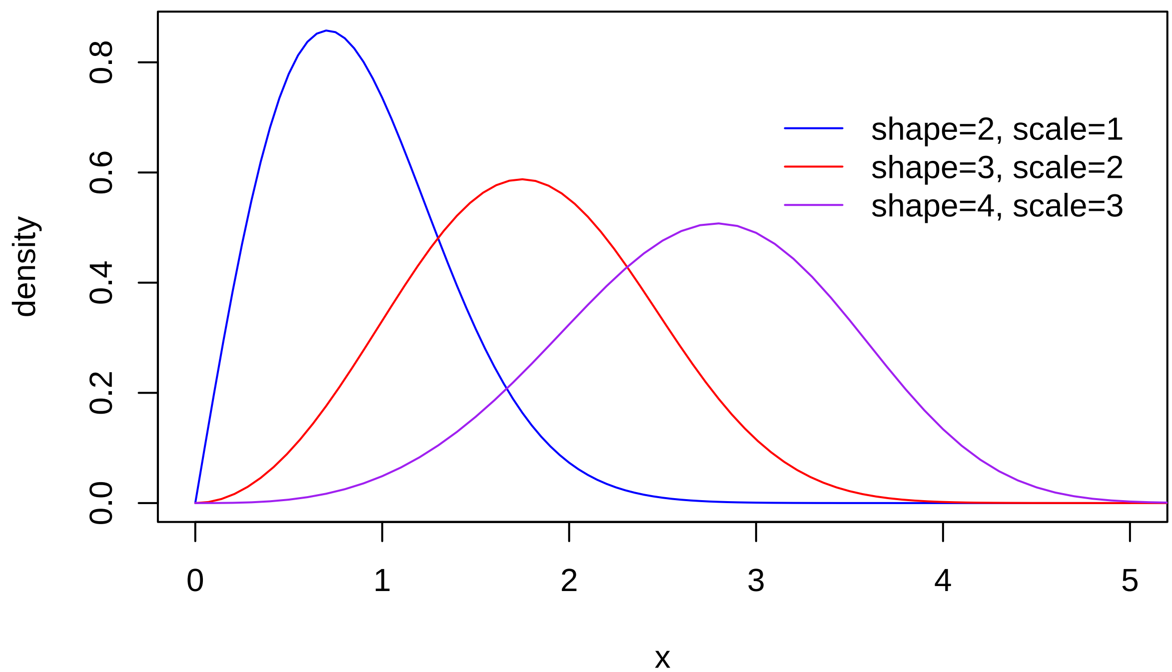


Figure 1: Examples of Weibull distributions by shape and scale

Trimming the mean

We compare sample mean ($\hat{\mu}_0$) to trimmed sample mean ($\hat{\mu}_p$) from above, i.e., from upper quantile. The expressions for the computation of $\hat{\mu}_0$ and $\hat{\mu}_p$ are respectively given according to the following formulas:

$$\hat{\mu}_0 = \frac{\sum_{i=1}^{n} X_i}{n}, \tag{1}$$

$$\hat{\mu}_p = \frac{\sum_{i=1}^{n} I(X_i \leq X_{(1-p)})X_i}{\sum_{i=1}^{n} I(X_i \leq X_{(1-p)})}, \tag{2}$$

where

$X_i \overset{iid}{\sim} F(\cdot)$ and $X_{(1-p)}$ is the $1-p^{th}$ quantile from the sample data.

For sample mean, it is exactly unbiased with variance given in Equ. 3. So, we only simulate trimmed mean.

$$\frac{\theta^2}{n}\left[\Gamma\left(1+\frac{2}{\beta}\right) - \left(\Gamma\left(1+\frac{1}{\beta}\right)\right)^2\right]. \tag{3}$$

The base R `mean` function has a trim option, but it trims both tails, above and below. It does not allow to trim above only. So, we add this functionality to the `mean` function with the parameter `trim.upper` as below:

```
data = 1:100
mean(data,trim=0.05) # trims 2.5% above and 2.5% below
#> [1] 50.5
mean(data,trim=0.05,trim.upper=TRUE) # trims 5% above
#> [1] 48
```

Simulation Function

Here we simulate for some values of the parameters. The function `simtrim` spits out mean, bias, variance, and MSE of sample mean and trimmed mean. As noted above, sample mean is always unbiased. Variance in this case is 1, so MSE is 1. On the other hand, trimmed mean at 5% introduces bias of about -0.66 but reduces variance to ~0.4. Thus overall MSE is about 0.85, or 15% smaller than the unbiased gold standard.

```
simtrim(n=20, shape=0.5, scale=1, p=0.05, S=10000)
#> $mu_0
#> [1] 2
#>
#> $bias_mu_0
#> [1] 0
#>
#> $var_mu_0
#> [1] 1
#>
#> $MSE_mu_0
#> [1] 1
#>
#> $mu_p
#> [1] 1.345241
#>
#> $bias_mu_p
#> [1] -0.6547586
#>
#> $var_mu_p
#> [1] 0.3990922
#>
#> $MSE_mu_p
#> [1] 0.8278011
```

Simulating for ranges of values

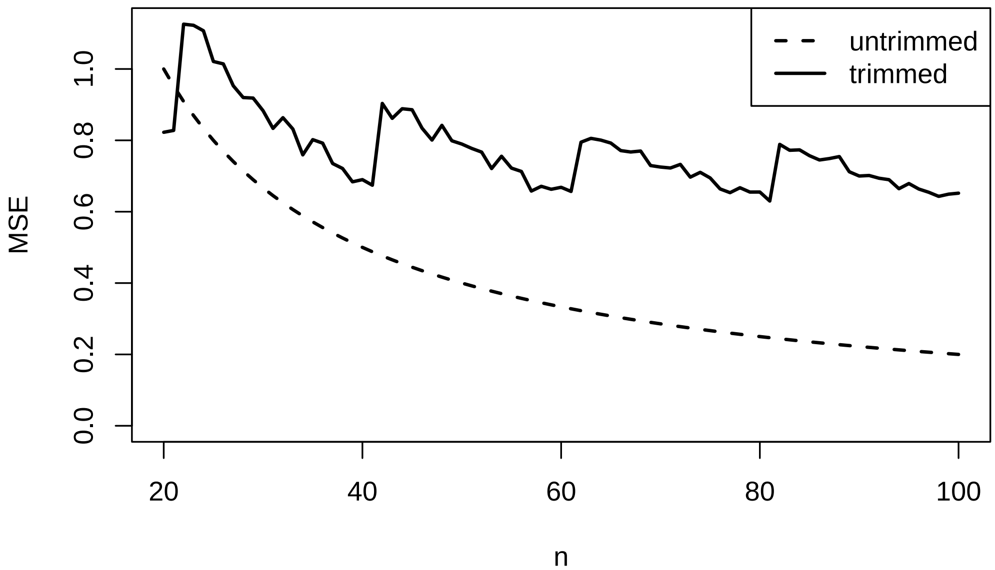
We can run simulations for many values of a parameter using the `simtrim_by` function. It takes the same arguments, but one is given as a vector. It then runs S simulations for each value of the vector and returns the results in a matrix. By default, the function will also plot the MSE of the trimmed and untrimmed means, but you can turn this off with the option `plot=FALSE`.

```
output = simtrim_by(n=seq(20,100,1),shape=0.5,scale=1,p=0.05,
                    S=1000,plot=FALSE)
output[,1:5]
#>           20        21         22         23         24
#> mu_0      2         2          2          2          2
#> bias_mu_0 0         0          0          0          0
#> var_mu_0  1         0.952381   0.9090909  0.8695652  0.8333333
#> MSE_mu_0  1         0.952381   0.9090909  0.8695652  0.8333333
#> mu_p      1.317025  1.341901   1.073305   1.069306   1.083062
#> bias_mu_p -0.682975 -0.6580988 -0.9266954 -0.9306939 -0.9169384
```

```
#> var_mu_p   0.3677932 0.3941197  0.2549288  0.2504925  0.2141495
#> MSE_mu_p   0.834248  0.8272137  1.113693   1.116684   1.054926
```

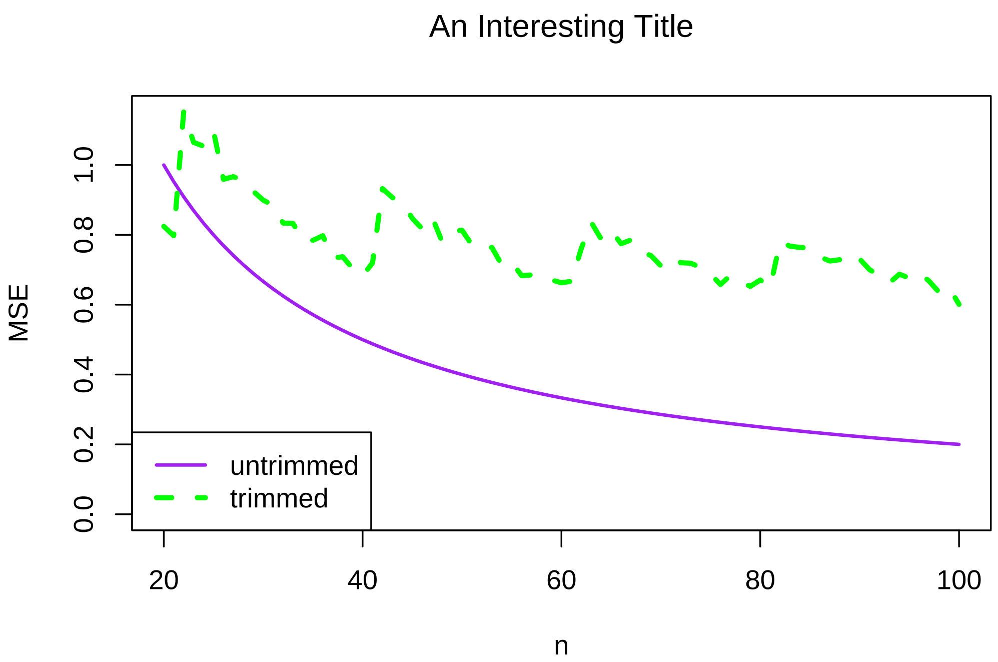
The default plot:

```
output = simtrim_by(n=seq(20,100,1),shape=0.5,scale=1,p=0.05,
                    S=1000)
```



We can also adjust the plot settings, including using syntax from the base `plot` function:

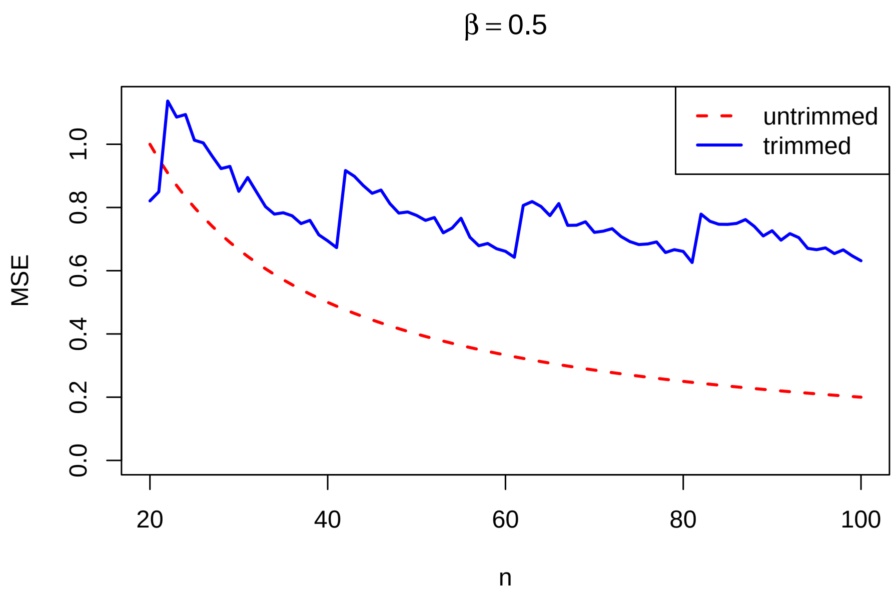
```
output = simtrim_by(n=seq(20,100,1),shape=0.5,scale=1,p=0.05,
                    S=1000, leg.pos="bottomleft",
                    lty=c(1,2),lwd=c(2,3),col=c("purple","green"),
                    main=expression("An Interesting Title"))
```



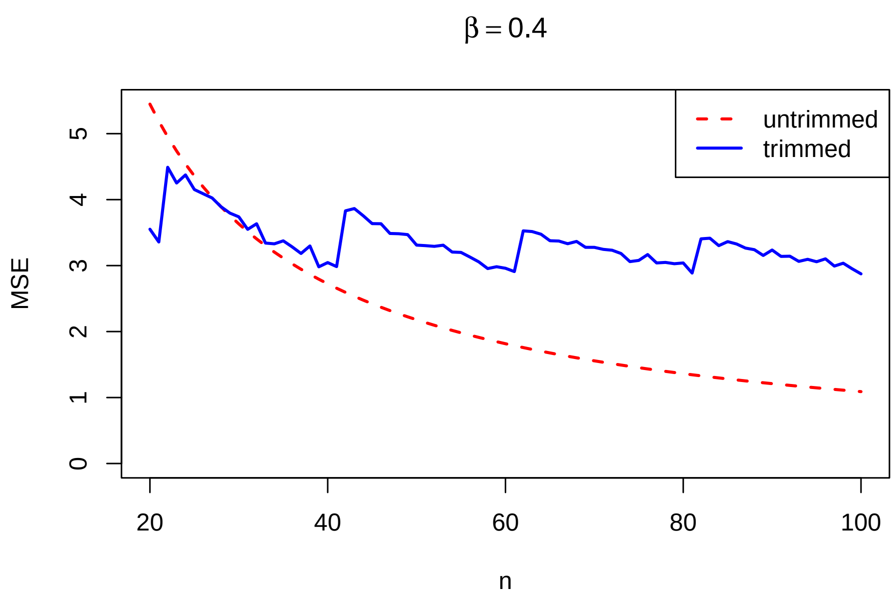
Adjusting shape param

As $\beta \to 0$ the mean and variance of weibull go to ∞. We thus see that as $\beta \to 0$, the trimmed mean becomes more and more useful:

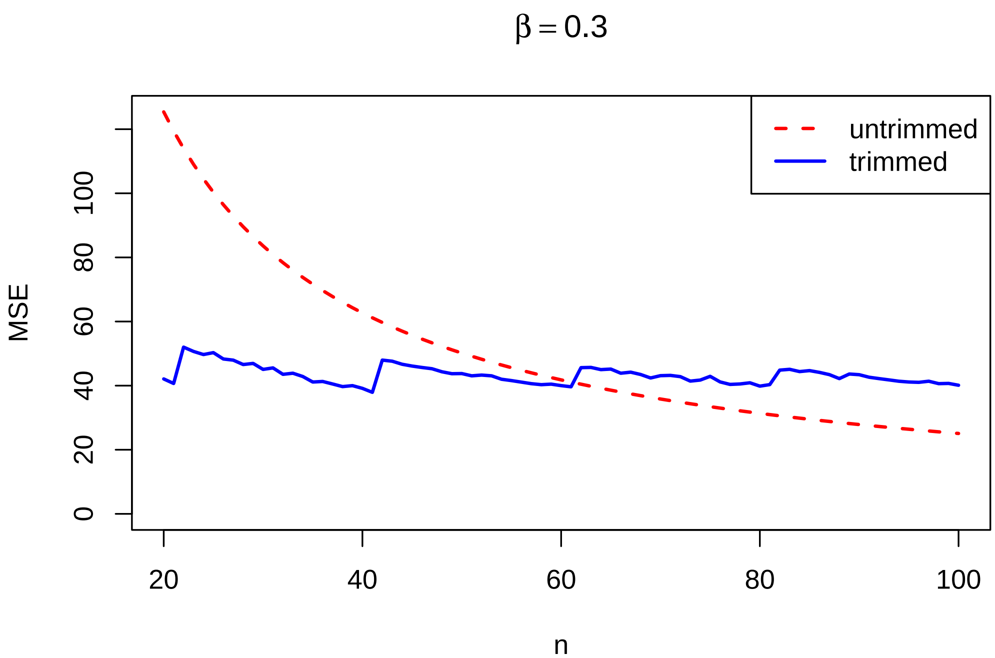
```
output = simtrim_by(n=seq(20,100,1),shape=0.5,scale=1,p=0.05,
                    S=1000,
                    leg.pos="topright",col=c("red","blue"),
                    main=expression(beta == 0.5))
```



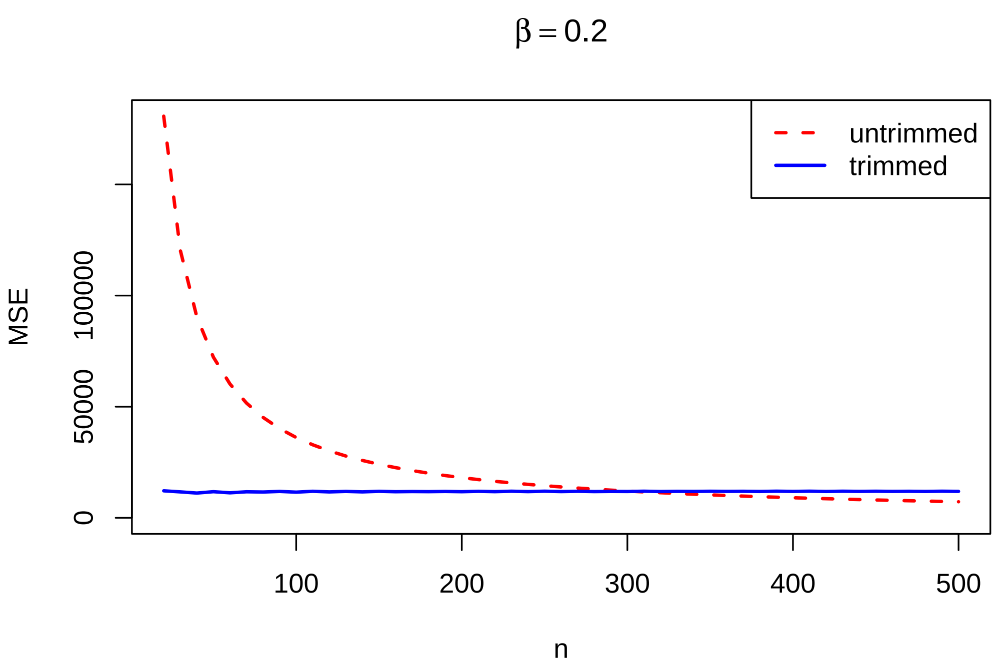

```
output = simtrim_by(n=seq(20,100,1),shape=0.4,scale=1,p=0.05,
                    S=1000,
                    leg.pos="topright",col=c("red","blue"),
                    main=expression(beta == 0.4))
```




```
output = simtrim_by(n=seq(20,100,1),shape=0.3,scale=1,p=0.05,
                    S=1000,
                    leg.pos="topright",col=c("red","blue"),
                    main=expression(beta == 0.3))
```




```
output = simtrim_by(n=seq(20,500,10),shape=0.2,scale=1,p=0.05,
                    S=1000,
                    leg.pos="topright",col=c("red","blue"),
                    main=expression(beta == 0.2))
```



Adjusting Trimming Proportion

We see that for a sample size of 300 and $\beta = 0.2$, MSE favors only a little trimming–that is, only the largest observation. Once the trimming proportion is around 0.05 or greater, the sample mean becomes better.

```
output = simtrim_by(n=300,shape=0.2,scale=1,p=seq(0.005,0.15,0.005),
                    S=2000,
                    leg.pos="bottomright",col=c("red","blue"))
```