Lab 10: Large Sample Normal Approximation to the Sampling Distribution of the MLE

Simulation and Comparison

In the written part, you obtained a normal approximation to the sampling distribution of the maximum likelihood estimator for a Pareto distribution. This normal approximation is approximately valid if the sample size is large enough.

Let's obtain a second approximation to the sampling distribution of the MLE, as well as an approximation to the sampling distribution of the Method of Moments estimator, by simulation. We will do this imagining that we know the true parameters are $x_0 = 1$ and $alpha = 1.1$. In real life we would not have this information; here, we are just using simulations to explore how the normal approximation works. The idea is to generate many different samples of a fixed size, and calculate the method of moments and maximum likelihood estimates based on each of those samples. Again, in real life we would have access to only a single sample; the idea here is to explore the behavior of these estimators across many samples.

Here is pseudo code for our procedure:

1. Set the sample size `n` to 20 (how large is each sample)
2. Set the number of simulated samples `n_sims` to 10000 (we will generate 10000 different samples of size n an find the MLE and MOM estimates from each)
3. Create a data frame with `n_sims` rows and two variables `alpha_hat_mom` and `alpha_hat_mle`
4. For i in 1, …, n_sims:
    a. Generate a sample of size n from the Pareto distribution with $x_0 = 1$ and $alpha = 1.1$
    b. Find the method of moments estimate and save it in the results data frame from step 3
    c. Find the maximum likelihod estimate and save it in the results data frame from step 3

In order to do step 4 a, you will need a function to generate samples from a Pareto distribution. Such a function is not built into R, but I have set one up and illustrated its use below:

```r
#' Sample from a Pareto distribution
#'
#' @param n sample size
#' @param x_0 location parameter for the Pareto distribution
#' @param alpha scale parameter for the Pareto distribution
#'
#' @return vector of length n, samples from the Pareto distribution.
rpareto <- function(n, x_0, alpha) {
    u <- runif(n, 0, 1)
    x <- x_0 / (1 - u)^{1/alpha}
    return(x)
}

rpareto(n = 20, x_0 = 1, alpha = 1.1)
```

```
##  [1] 13.733749  9.334469  1.374190  1.891257  1.213178  1.899796  2.224087
##  [8]  3.302528  6.015540 19.559215  2.152161  1.040781  5.812224  7.237435
## [15]  1.203887  1.612231 95.617965 17.342040  1.365327  1.474110
```

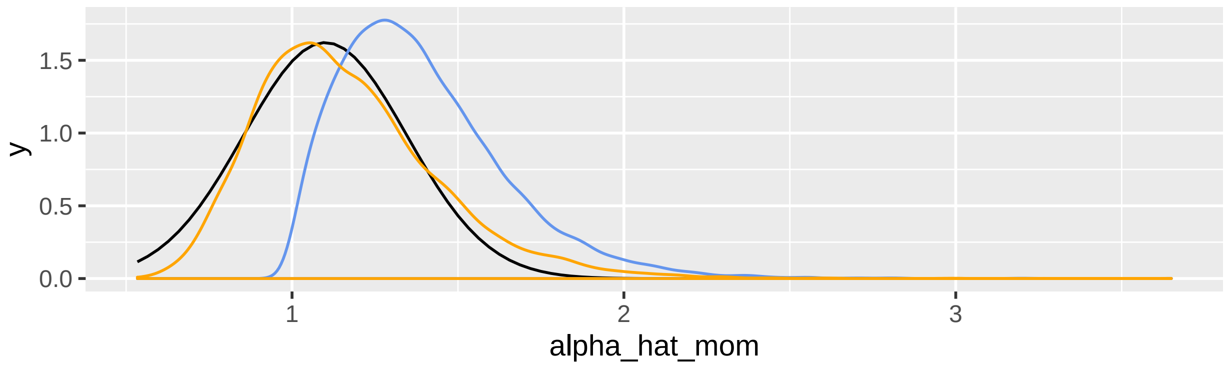
1. Implement the procedure outlined above to obtain sampling-based estimates of the sampling distributions of the MOM and MLE estimators.

```r
n <- 20
n_sims <- 10000

results <- data.frame(
  alpha_hat_mom = rep(NA, n_sims),
  alpha_hat_mle = rep(NA, n_sims)
)
```

```r
for(i in seq_len(n_sims)) {
  # replace NA on the next line with a call to rpareto, specifying n = n, x_0 = 1, and alpha = 1.1
  x <- rpareto(n = n, x_0 = 1, alpha = 1.1)

  # calculate the method of moments estimate based on the sample data x
  results$alpha_hat_mom[i] <- mean(x) / (mean(x) - 1)

  # calculate the MLE based on the sample data x
  results$alpha_hat_mle[i] <- 1/mean(log(x))
}
```

2. Make a plot with 3 layers on it in different colors: (1) the normal approximation to the sampling distribution of the MLE; (2) the approximation to the sampling distribution of the MLE based on simulation; and (3) the approximation to the sampling distribution of the MOM estimator based on simulation.

```r
library(ggplot2)

# add more layers to the plot below
ggplot(data = results) +
  stat_function(fun = dnorm, args = list(mean = 1.1, sd = sqrt(1.1^2 / n))) +
  geom_density(mapping = aes(x = alpha_hat_mom), color = "cornflowerblue") +
  geom_density(mapping = aes(x = alpha_hat_mle), color = "orange")
```



3. Calculate an estimate of the mean and variance of the MLE and the MOM estimators based on your simulation results.

```r
mean(results$alpha_hat_mle)
```

```
## [1] 1.164702
```

```r
var(results$alpha_hat_mle)
```

```
## [1] 0.07604687
```

```r
mean(results$alpha_hat_mom)
```

```
## [1] 1.392071
```

```r
var(results$alpha_hat_mom)
```

```
## [1] 0.06348843
```

4. How well does the normal approximation represent the sampling distribution of the MLE?

The normal approximation is pretty good, but not perfect – the actual sampling distribution is skewed right.

5. Does the MLE appear to be unbiased? What about the MoM estimator?

```r
mean(results$alpha_hat_mom) - 1.1
```

```
## [1] 0.292071
```

```r
mean(results$alpha_hat_mle) - 1.1
```

```
## [1] 0.06470207
```

Based on these results, neither of the estimators seems to be exactly unbiased, but the bias of the MLE is smaller.

6. How does the variance of the sampling distribution of the MLE compare to the variance of the sampling distribution of the MoM estimator?

The method of moments estimator has slightly smaller variance.

7. Combine estimates of the bias and variance of the MLE from the simulation-based approximation to the sampling distribution to approximate the Mean Squared Error (MSE) of the MLE. Do this again to approximate the MSE of the MoM estimator. Which has a lower mean squared error?

```r
(mean(results$alpha_hat_mom) - 1.1)^2 + var(results$alpha_hat_mom)
```

```
## [1] 0.1487939
```

```r
(mean(results$alpha_hat_mle) - 1.1)^2 + var(results$alpha_hat_mle)
```

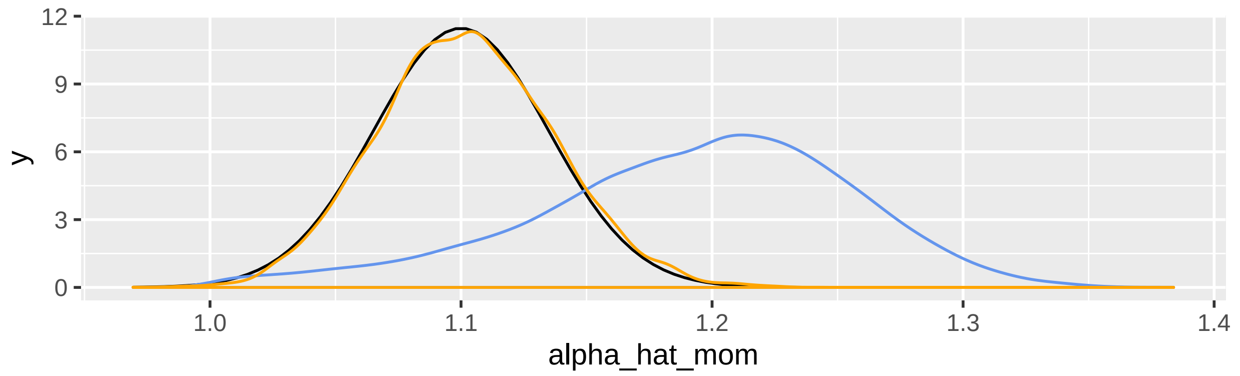
```
## [1] 0.08023323
```

The maximum likelihood estimator has lower mean squared error.

8. Update the sample size to 1000 and re-run your code from part 1. How do your answers to questions 4, 5, and 6 above change?

```r
# copy and paste your code from part 1, and change the sample size to 1000
n <- 1000
n_sims <- 10000

results <- data.frame(
  alpha_hat_mom = rep(NA, n_sims),
  alpha_hat_mle = rep(NA, n_sims)
)

for(i in seq_len(n_sims)) {
  # replace NA on the next line with a call to rpareto, specifying n = n, x_0 = 1, and alpha = 1.1
  x <- rpareto(n = n, x_0 = 1, alpha = 1.1)

  # calculate the method of moments estimate based on the sample data x
  results$alpha_hat_mom[i] <- mean(x) / (mean(x) - 1)

  # calculate the MLE based on the sample data x
  results$alpha_hat_mle[i] <- 1/mean(log(x))
}
```

```r
# copy and paste your plot code from part 2 here
ggplot(data = results) +
  stat_function(fun = dnorm, args = list(mean = 1.1, sd = sqrt(1.1^2 / n))) +
  geom_density(mapping = aes(x = alpha_hat_mom), color = "cornflowerblue") +
  geom_density(mapping = aes(x = alpha_hat_mle), color = "orange")
```



Normal approximation for MLE: very good

Bias:

```
# find the approximate bias of each estimator based on the results from your simulated samples
mean(results$alpha_hat_mom) - 1.1
```

```
## [1] 0.09124651
```

```
# find the approximate bias of each estimator based on the results from your simulated samples
mean(results$alpha_hat_mle) - 1.1
```

```
## [1] 0.002052477
```

MLE has lower bias (unchanged). The MLE's bias is now approximately 0.

MSE:

```
# find the approximate MSE of each estimator based on the results from your simulated samples
(mean(results$alpha_hat_mom) - 1.1)^2 + var(results$alpha_hat_mom)
```

```
## [1] 0.01235224
```

```
# find the approximate MSE of each estimator based on the results from your simulated samples
(mean(results$alpha_hat_mle) - 1.1)^2 + var(results$alpha_hat_mle)
```

```
## [1] 0.001224372
```

The MLE has lower MSE (unchanged).