ECON 6200

Problem Set 6

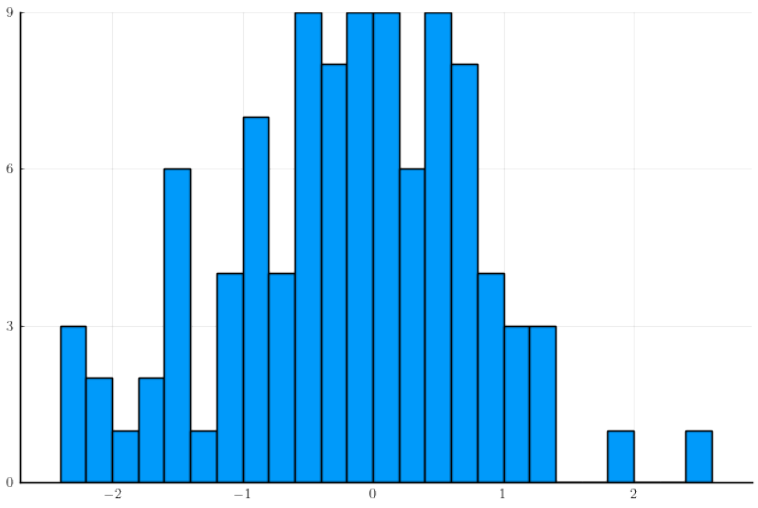
Gabe Sekeres

May 1, 2025

n.b. Code is below, in the code section. It is entirely in Julia

1. **Bootstrap CI for the Median**

   (a) I drew 100 realizations from a standard normal distribution, fixing the seed for replicability. A histogram of my data is:



   The sample median is -0.0899, and the confidence intervals using the bootsrapped standard errors and percentiles are, respectively,

$$(-0.3006, 0.1207) \quad ; \quad (-0.3302, 0.0871)$$

   Both cover zero, though they clearly reflect the skew of the original sample.

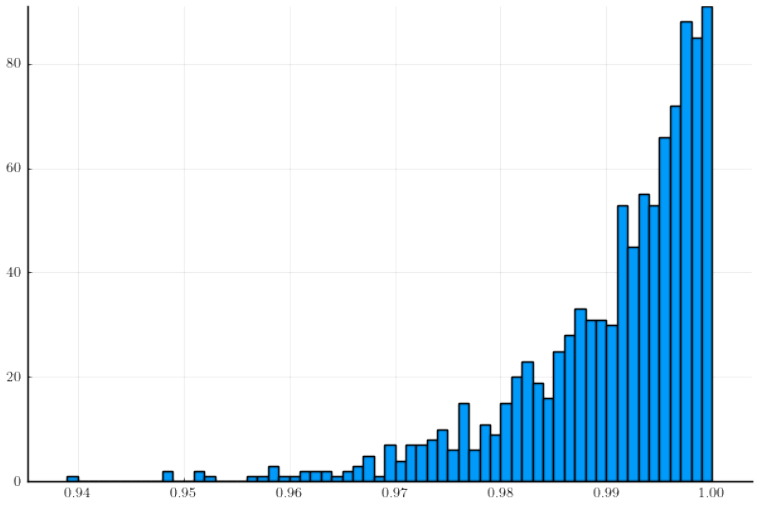
   (b) I simulated this 1,000 times using different seeds, which took approximately 3.33 seconds to run. I got that the coverage of the standard error method was 0.889, and the coverage of the percentile method was 0.895. Interestingly, the average length of the percentile method was smaller at 0.405 while the average length of the standard error method was 0.416.

2. **Bootstrap Failures** I first give the quick red flags for why bootstrapping may not work for the below cases:

   - The Cauchy distribution has no finite moments, which raises an immediate red flag, since the sample mean does not have finite variance and does not converge to the population mean. The same is true for all moments.
   - The red flag here is that there is a kink in the function at exactly the population mean. For bootstrapping, we want to find an interior minimum but there is no point where the derivative of T_n is (only) zero in any open neighborhood of $\mathbb{E}(X)$.

- When the true population mean is zero, we will have a large point mass at zero, and so the bootstrapped distribution will be degenerate, in a way that cannot be normalized by $\sqrt{n}$
- Since $\max_i X_i < \alpha$ with probability 1, the order statistic is biased downwards in every finite sample.

I verified the final bootstrap failure using Julia, by simulating a resampling with sample size 100 on $U[0,1]$. I made a 95% confidence interval for α, and verified that the coverage probability over 1,000 repetitions was 0, since the bootstrap interval is always shorter than the real one and never reaches the true $\alpha = 1$. The code is below, and took 0.89 seconds to run. To get further intuition, observe that except in the event that 3 or more of the samples are exactly 1, the interval will always fail to include 1. The event that even one sample is equal to 1 has measure 0, and occurs with precisely probability 0. Included below is a histogram of the upper bound of the 1,000 intervals.



Code

```julia
using Random, Distributions, Plots

# Make plots look pretty
pyplot()
PyPlot.rc("text", usetex=true)
PyPlot.rc("font", family="serif")
PyPlot.matplotlib.rcParams["mathtext.fontset"] = "cm"

# Set seed for reproducibility
Random.seed!(12345)

# 1.1 Bootstrapping the median

data = rand(Normal(0, 1), 100)

p1 = histogram(data, bins=25, background=:transparent, legend=false)
savefig(p1, "ps6_code/ps6_hist.png")

println("The median of the data is $(median(data))")

B = 2000
bootstrap_medians = [median(sample(data, 100, replace=true)) for _ in 1:B]

se = std(bootstrap_medians)
z = quantile(Normal(), 0.95)

ci_se = [median(data) - z*se, median(data) + z*se]
ci_perc = quantile(bootstrap_medians, [0.05, 0.95])

println("The 95% confidence interval using the standard error is $(ci_se)")
println("The 95% confidence interval using the percentile method is $(ci_perc)
    ")

# 1.2 Test size with monte carlo simulation
function simulate_bootstrap_median(n, B, iters)
    cover_se = zeros(iters)
    cover_perc = zeros(iters)
    len_se = zeros(iters)
    len_perc = zeros(iters)
    z = quantile(Normal(), 0.95)

    for i in 1:iters
        Random.seed!(12345 * i)
        data = rand(Normal(0, 1), n)
        medians = [median(sample(data, n, replace=true)) for _ in 1:B]

        se = std(medians)
        ci_se = [median(data) - z*se, median(data) + z*se]
        ci_perc = quantile(medians, [0.05, 0.95])
```

```
        cover_se[i] = ci_se[1] <= 0 <= ci_se[2]
        cover_perc[i] = ci_perc[1] <= 0 <= ci_perc[2]

        len_se[i] = ci_se[2] - ci_se[1]
        len_perc[i] = ci_perc[2] - ci_perc[1]
    end

    return cover_se, cover_perc, len_se, len_perc
end

@time cover_se, cover_perc, len_se, len_perc = simulate_bootstrap_median(100,
    2000, 1000)

println("The coverage of the standard error method is $(mean(cover_se))")
println("The coverage of the percentile method is $(mean(cover_perc))")

println("The average length of the standard error method is $(mean(len_se))")
println("The average length of the percentile method is $(mean(len_perc))")

# 2 Bootstrap uniform 0,1 maximum

function simulate_bootstrap_uniform_max(n, B, iters)
    cover = zeros(iters)
    len = zeros(iters)
    maxima = zeros(iters)

    for i in 1:iters
        Random.seed!(12345 * i)
        data = rand(Uniform(0, 1), n)
        maxes = [maximum(sample(data, n, replace=true)) for _ in 1:B]

        se = std(maxes)
        ci = quantile(maxes, [0.05, 0.95])

        cover[i] = ci[1] <= 1 <= ci[2]
        len[i] = ci[2] - ci[1]
        maxima[i] = maximum(data)
    end

    return cover, len, maxima
end

@time cover, len, maxima = simulate_bootstrap_uniform_max(100, 2000, 1000)
println("The coverage of the bootstrap is $(mean(cover))")
println("The average length of the bootstrap is $(mean(len))")
println("The average maximum of the bootstrap is $(mean(maxima))")

# Plot maxima
p2 = histogram(maxima, bins=100, background=:transparent, legend=false)
savefig(p2, "ps6_code/ps6_max.png")
```