Homework 2

Patrick Addona

January 24, 2025

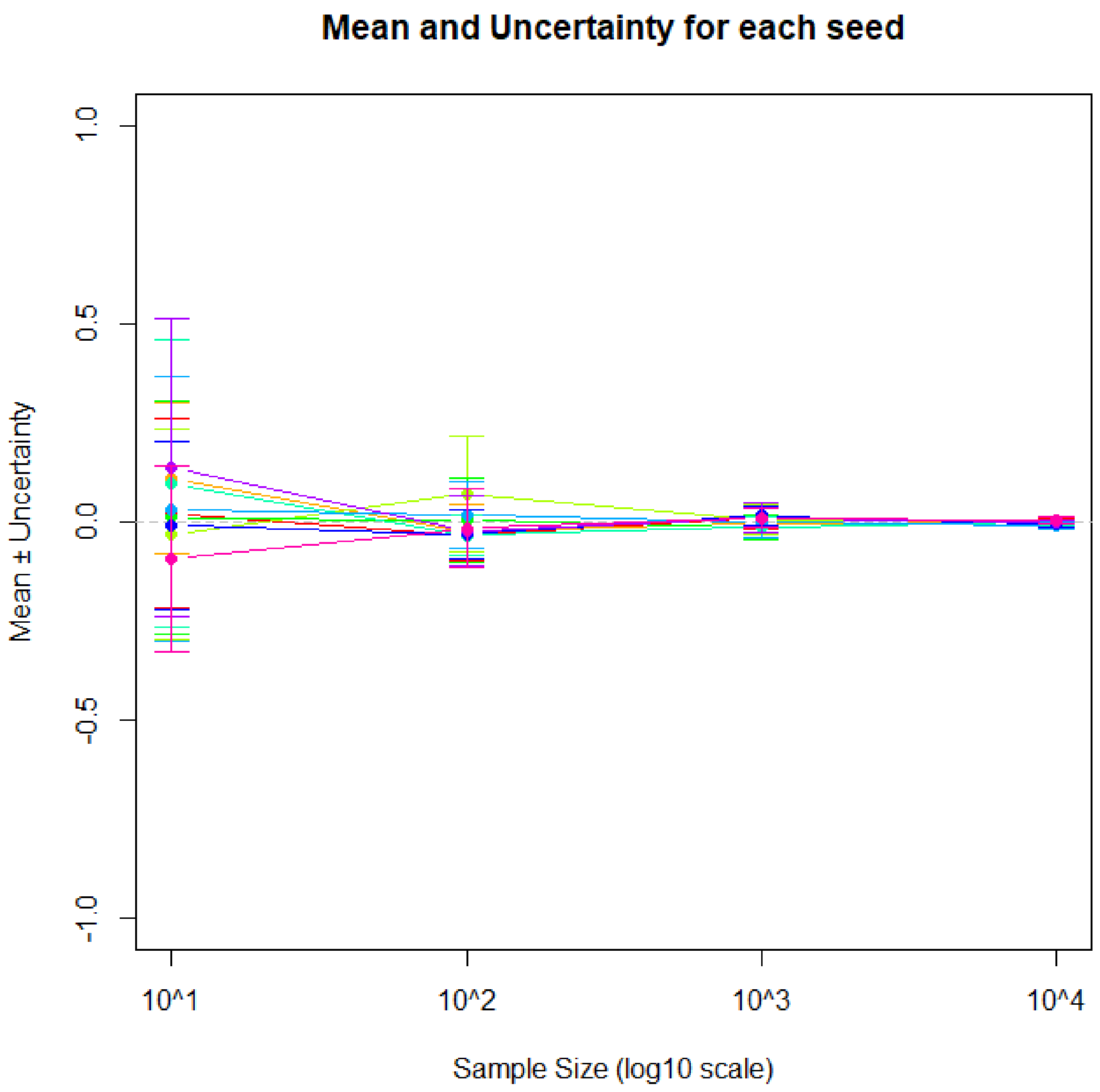
Question 1

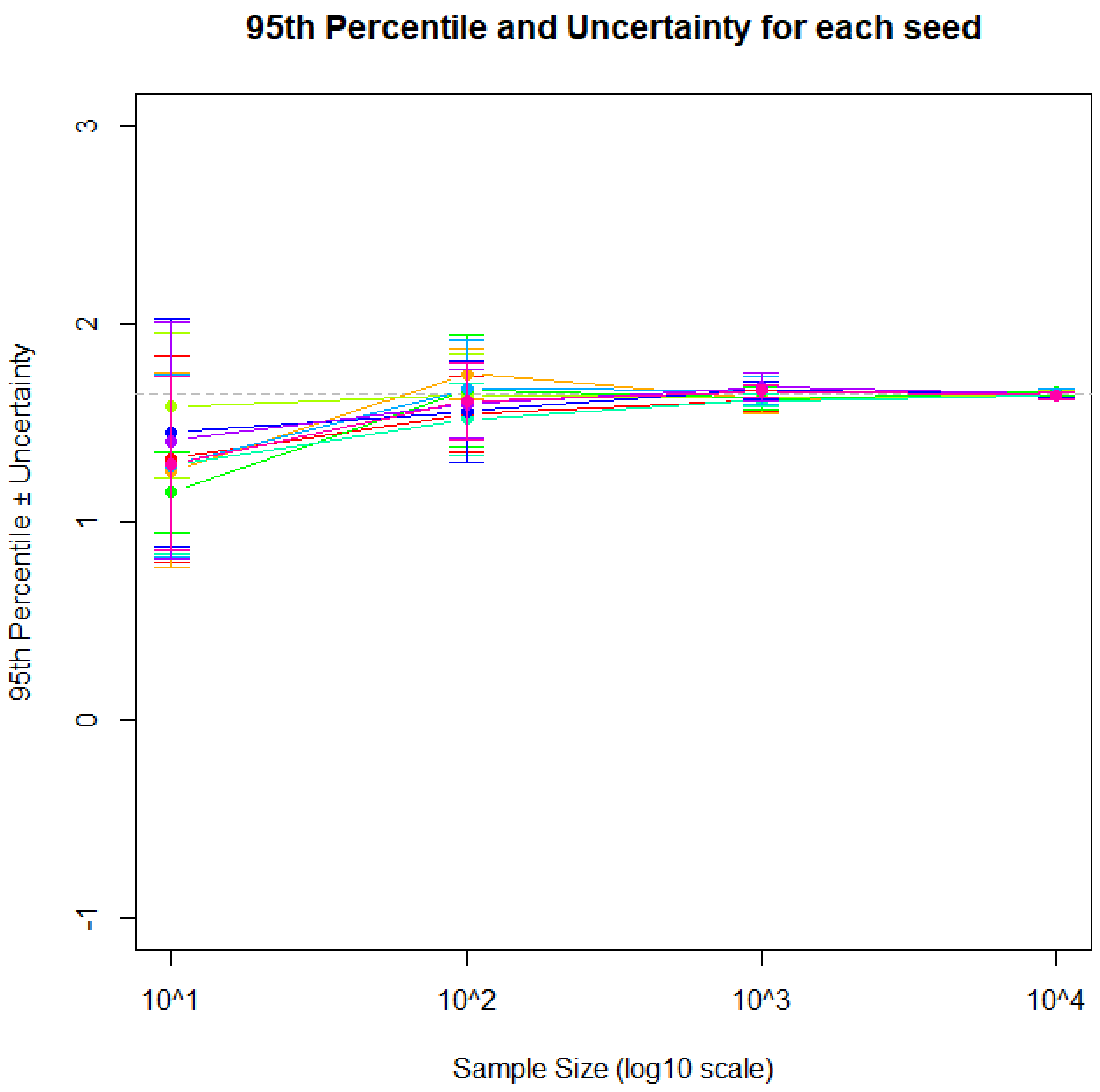
Use a Monte Carlo simulation to determine the mean and the 95th percentile from a known univariate normal distribution with a mean of zero and a standard deviation of one with your estimated uncertainties.

To complete this task, my approach was to sample from the known normal distribution using R's "rnorm" function. The number of samples generated was chosen to vary, so that I could show that increasing sample sizes aids in convergence. I chose to draw 10 sets each of 10, 100, 1000, and 10,000 samples, then calculated the mean and 95th percentile of these sets. I also calculated the standard deviation of these sets, in order to quantify the uncertainty of the approximations. Finally, I repeated this entire process across 10 seeds, to see both how exactly the approximations could vary and how they would get more accurate as the sample size increased.

The assumptions I made in this code were that 10 simulations would be enough to generate accurate approximations across the various sample sizes, and that 10,000 would be a good enough upper limit to show convergence.

My results are displayed in the two figures below:


Mean and Uncertainty for each seed
Mean ± Uncertainty
1.0
0.5
0.0
-0.5
-1.0
10^1
10^2
10^3
10^4
Sample Size (log10 scale)



As we can see, as the sample sizes grow, the variance between approximations decreases and converges to the expected values for both mean and 95th percentile.

As stated earlier, I chose to select samples using R's "rnorm" function, inspired by the sample code from class "lab0_sample.R". This seemed to be the most straightforward way to approach the problem, and referring to the sample code helped me understand R syntax. To determine convergence, I simply relied upon the expected values for each parameter, which were already known (I determined the 95th percentile using R's "qnorm" function). I could have also taken the absolute value of the distances of each mean from the expected value, but for figures as simple as these, visual inspected seemed to be an appropriate heuristic.

These analyses are reproducible through the seed-generation system I included in the code.

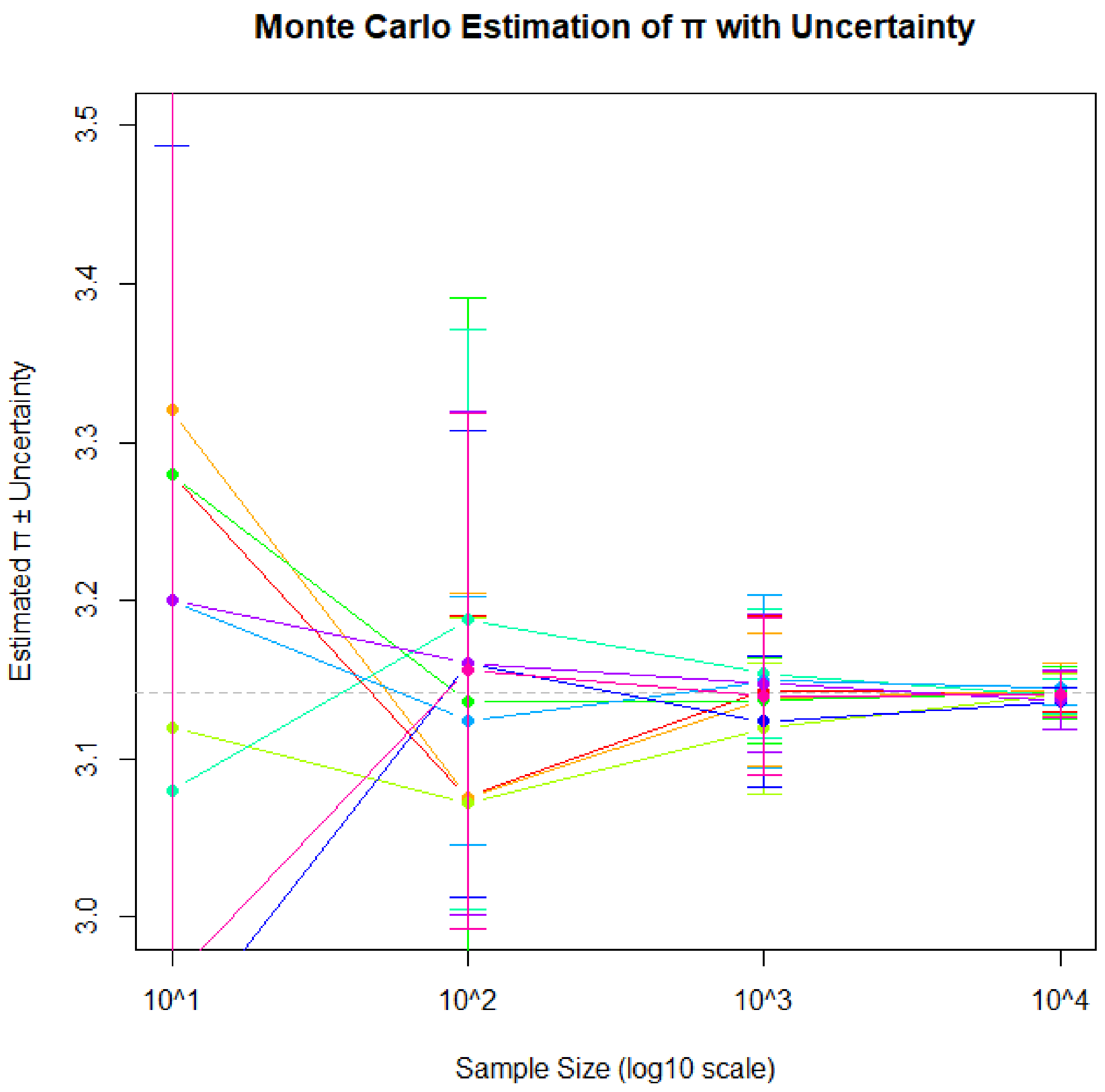
Question 2

Use a Monte Carlo simulation to determine the value of pi with your estimated uncertainties.

To complete this task, I took a similar approach to the previous problem. I generated 10, 100, 1000, and 10,000 two-dimensional uniform samples from between 0 and 1, then checked which of these samples fit inside the unit quarter circle. Then, I took the average of these values and multiplied by 4, and finally took the mean and standard deviation of these figures.

The assumptions I made here were similar, that 10 simulations and the provided sample sizes would be enough to generate decent approximations.

The results are shown below:



As we can see, the approximations begin far more varied than the last problem. Even at 10,000 samples, the variance is still visible in the graph, although the trend is clear.

My choice to approximate π with the inscribed circle method was based on the fact that this is a well-known way to accomplish this task and is an introductory application of Monte Carlo methods. There are other methods, such as infinite series or numerical integration that could have been used, but these are either slow to converge or may have required more complex coding.

Once again, the results are reproducible due to the provided seeds.

Appendix

Problem 1:

```r
##  author: Patrick Addona
##  copyright by the author
##  distributed under the GNU general public license
##  https://www.gnu.org/licenses/gpl.html
##  no warranty (see license details at the link above)

rm(list = ls())
graphics.off()

#parameters
sample_sizes <- c(1:4)
n_simulations <- 10
n_seeds <- 9

#pre-allocate results
means <- numeric(n_simulations)
percentiles_95 <- numeric(n_simulations)
mean_mean <- numeric(length(sample_sizes))
mean_percentile <- numeric(length(sample_sizes))
mean_uncertainty <- numeric(length(sample_sizes))
percentile_uncertainty <- numeric(length(sample_sizes))

seed_colors <- rainbow(n_seeds)

windows()
#initialize graph for mean
plot(NULL,xlim = c(1,length(sample_sizes)),ylim=c(-1,1),
     xlab = "Sample Size (log10 scale)",ylab = "Mean    Uncertainty",
     main = "Mean and Uncertainty for each seed",
     xaxt = "n")
axis(1,at=1:length(sample_sizes),labels = paste0("10^",sample_sizes))
#line showing expected value of mean
abline(h=0,col="gray",lty=2)

#set first seed
set.seed(0)

#for each seed, then for each sample size, run 10 simulations and calculate mean and 95th
    percentile
#store the standard deviation of mean and 95th percentile for each sample size
for (k in 1:n_seeds) {
  for (j in sample_sizes) {
    for (i in 1:n_simulations) {
      samples <- rnorm(10^j, mean = 0, sd = 1)

      means[i] <- mean(samples)
      percentiles_95[i] <- quantile(samples, 0.95)
    }

    mean_mean[j] = mean(means)
    mean_percentile[j] = mean(percentiles_95)
    mean_uncertainty[j] <- sd(means)
    percentile_uncertainty[j] <- sd(percentiles_95)
  }

  points(1:length(sample_sizes),mean_mean,type="b",col=seed_colors[k],pch=19,lwd=1.5)
  arrows(1:length(sample_sizes),mean_mean-mean_uncertainty,
         1:length(sample_sizes),mean_mean+mean_uncertainty,
         angle=90,code=3,length=0.1,col=seed_colors[k])
  set.seed(k)
}

windows()
# initialize plot for 95th percentile
plot(NULL, xlim = c(1, length(sample_sizes)), ylim = c(-1, 3),
     xlab = "Sample Size (log10 scale)", ylab = "95th Percentile    Uncertainty",
     main = "95th Percentile and Uncertainty for each seed",
     xaxt = "n")
```

```r
axis(1, at = 1:length(sample_sizes), labels = paste0("10^", sample_sizes))
#line showing expected value of 95th percentile
abline(h = qnorm(0.95, mean = 0, sd = 1), col = "gray", lty = 2)

# Loop for calculating and plotting Percentiles
for (k in 1:n_seeds) {
  for (j in sample_sizes) {
    for (i in 1:n_simulations) {
      samples <- rnorm(10^j, mean = 0, sd = 1)
      percentiles_95[i] <- quantile(samples, 0.95)
    }

    mean_percentile[j] <- mean(percentiles_95)
    percentile_uncertainty[j] <- sd(percentiles_95)
  }

  points(1:length(sample_sizes), mean_percentile, type = "b", col = seed_colors[k], pch = 19, lwd =
     1.5)
  arrows(1:length(sample_sizes), mean_percentile - percentile_uncertainty,
         1:length(sample_sizes), mean_percentile + percentile_uncertainty,
         angle = 90, code = 3, length = 0.1, col = seed_colors[k])
  set.seed(k)
}
```

Problem 2:

```r
##  author: Patrick Addona
##  copyright by the author
##  distributed under the GNU general public license
##  https://www.gnu.org/licenses/gpl.html
##  no warranty (see license details at the link above)

rm(list = ls())
graphics.off()

#parameters
sample_sizes <- c(1:4)
n_simulations <- 10
n_seeds <- 9

#pre-allocate results
pi_estimates <- numeric(n_simulations)
pi_mean <- numeric(length(sample_sizes))
pi_uncertainty <- numeric(length(sample_sizes))

seed_colors <- rainbow(n_seeds)

#initialize graph
plot(NULL, xlim = c(1, length(sample_sizes)), ylim = c(3, 3.5),
     xlab = "Sample Size (log10 scale)", ylab = "Estimated      Uncertainty",
     main = "Monte Carlo Estimation of    with Uncertainty",
     xaxt = "n")
axis(1, at = 1:length(sample_sizes), labels = paste0("10^", sample_sizes))
#line showing expected value of pi
abline(h = pi, col = "gray", lty = 2)

set.seed(0)
#for each seed, for each sample size, run 10 simulations and calculate mean approximation of pi
for (k in 1:n_seeds) {
  for (j in sample_sizes) {
    for (i in 1:n_simulations) {
      #generate random points with coordinates from U[0,1]
      x <- runif(10^j, 0, 1)
      y <- runif(10^j, 0, 1)

      #count points inside the quarter circle
      inside_circle <- (x^2 + y^2) <= 1
      #ratio of points inside the circle to total number approximates   /4
      pi_estimates[i] <- 4 * sum(inside_circle) / (10^j)
    }

    pi_mean[j] <- mean(pi_estimates)
    pi_uncertainty[j] <- sd(pi_estimates)
  }

  points(1:length(sample_sizes), pi_mean, type = "b", col = seed_colors[k], pch = 19, lwd = 1.5)
  arrows(1:length(sample_sizes), pi_mean - pi_uncertainty,
         1:length(sample_sizes), pi_mean + pi_uncertainty,
         angle = 90, code = 3, length = 0.1, col = seed_colors[k])
  set.seed(k)
}
```