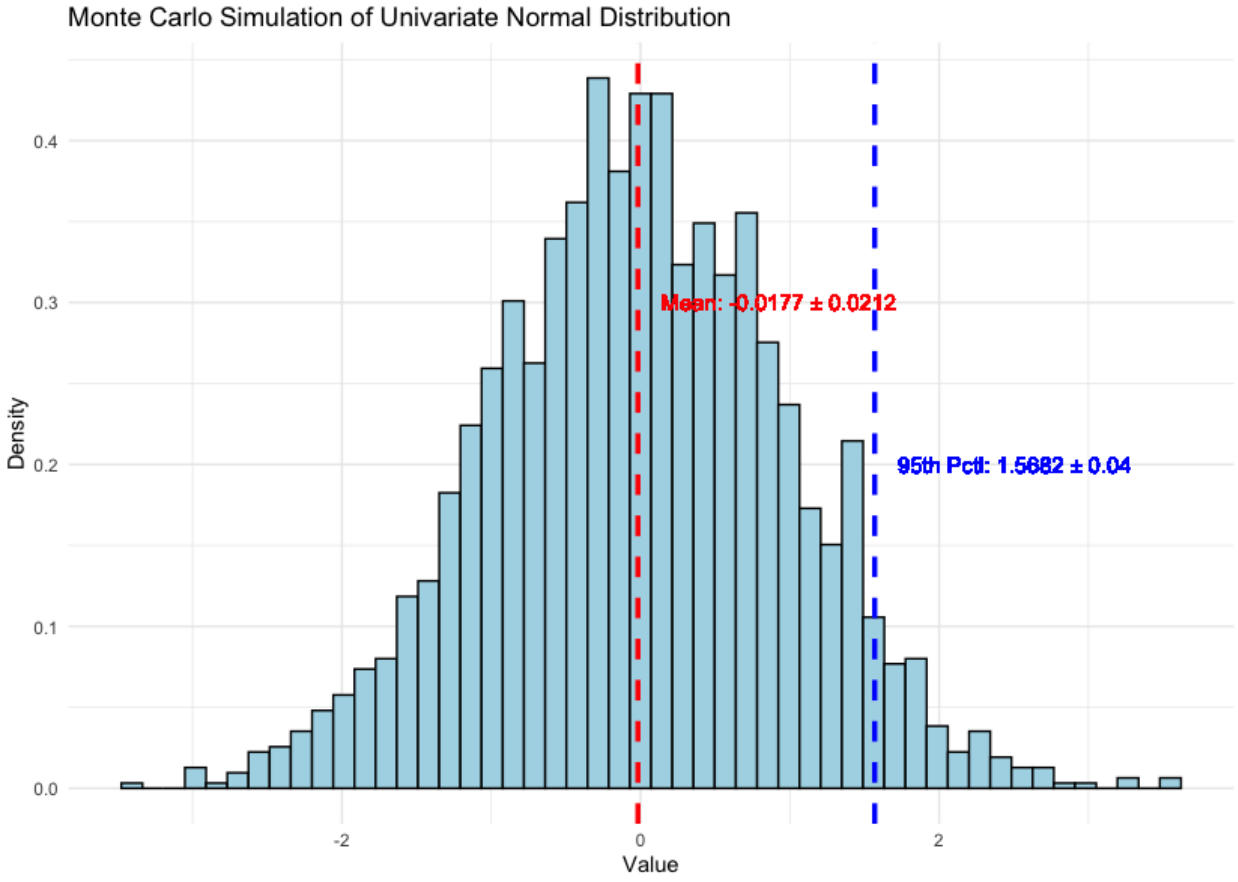
**Mean Estimation**

ci. Unlike the frequentist approach in computing the mean and 95% percentile where the distributions of the parameters are known, in the Bayes approach the said distributions are not initially known and we have to estimate them with certain uncertainties.

In my analysis, I generated 2200 random samples with the Monte Carlo technique using a given mean and percentile. I averaged the values of these samples to derive the mean and calculated the standard error applying the traditional formular SE = SD/sqrt(n). The diagram below showcases the estimated mean and 95$^{th}$ of the distribution obtained.




cii. To obtain the random samples, I employed the rnorm function which basically generates random numbers from a normal distribution. In this case my distribution had a mean of zero and standard deviation of 1. I decided on this pathway as it best simulates the univariate normal distribution whose mean I want to estimate.

cii. The analysis is reproducible given I used a specified seed for random sample generation. Thus, another person can run my experiment and generate the same random numbers if they

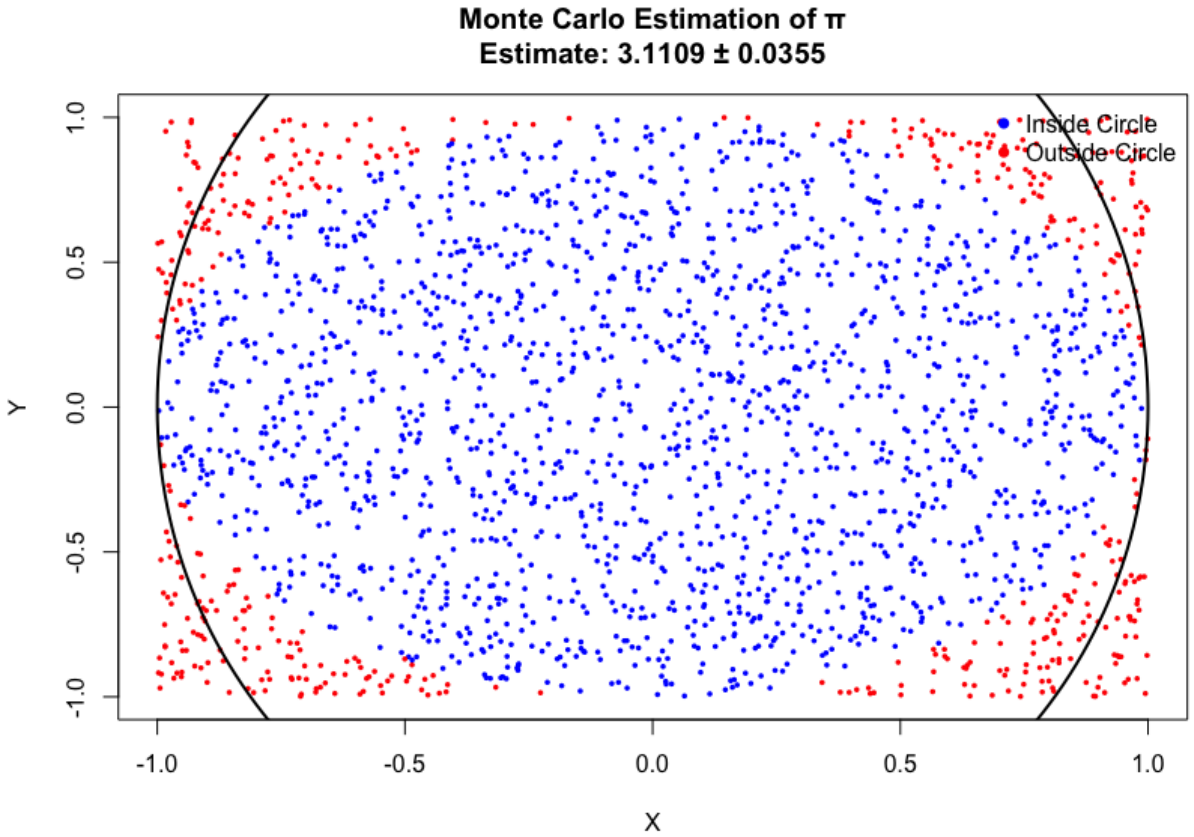
utilize the seed I chose. This helps to standardize the analysis by different people and at different times.

cv. I would like to reference sampling techniques employed from Confidence Interval Estimation chapter of Verzani's "Simple R"

cvi. I used the license of GNU for this project.

**Pi Estimation**

ci. For pi estimation with Monte Carlo, I wanted to estimate that given a random number of 2200 points falling on a unit circle embedded in a square, how many of those points would fall in the circle. With these values, I estimate the value of pi with the proportion of points inside the circle. The random points x and y were sampled between -1 to 1 on both axes of the square. A visual illustration of the process can be seen below.




cii. I decided to sample the points from -1 to 1 on the axes of a unit square in order to restrain my samplings within the confines of the square.

ciii. The analysis is reproducible given I set the seed for my random samplings.

cv. I would like to reference sampling techniques employed from Confidence Interval Estimation chapter of Verzani's "Simple R"

cvi. The code utilizes the GNU license.

## Appendix

```
library(ggplot2)

#Question 1

#Generating random samples to simulate the mean and 95th percentile

set.seed(42) #seed for random sample generation to ensure reproducibility

meanx = 0
sdx = 1
sim = 2200

#generating random samples per Monte Carlo simulation
samples = rnorm(sim, mean = meanx,sd = sdx)

#estimating mean and its uncertainties
mean_pr = mean(samples)
mean_se = sd(samples)/sqrt(sim)

#estimating the 95th percentile and its uncertainties
nine5 = quantile(samples, 0.95)
nine5_se = sd(samples[samples>=nine5])/sqrt(sum(samples>=nine5))

cat("Estimate Mean:",mean_pr,"±",mean_se)
cat("Estimate 95th percentile:",nine5,"±",nine5_se)

#Visualizing results
ggplot(data.frame(samples), aes(x = samples)) +
  geom_histogram(aes(y = ..density..), bins = 50, fill = "lightblue", color = "black") +
```

```
  geom_vline(xintercept = mean_pr, color = "red", linetype = "dashed", size = 1.2) +
  geom_vline(xintercept = nine5, color = "blue", linetype = "dashed", size = 1.2) +
  geom_text(aes(x = mean_pr, y = 0.3, label = paste0("Mean: ", round(mean_pr, 4), " ± ",
round(mean_se, 4))),
            color = "red", hjust = -0.1) +
  geom_text(aes(x = nine5, y = 0.2, label = paste0("95th Pctl: ", round(nine5, 4), " ± ",
round(nine5_se, 4))),
            color = "blue", hjust = -0.1) +
  labs(title = "Monte Carlo Simulation of Univariate Normal Distribution",
       x = "Value",
       y = "Density") +
  theme_minimal()

#Question 2 - Computing pi with the determined uncertainties

x = runif(sim,-1,1) #sampling x from -1 to 1 on both axes
y = runif(sim, -1,1)  #sampling y from -1 to 1 on both axes

innercircle = x^2 + y^2 <= 1

pi_val = (sum(innercircle)/sim)*4 #pi estimate

pi_prop = sum(innercircle)/sim #proportion of points in inner circle

pi_se = 4 * sqrt((pi_prop*(1-pi_prop))/sim) #uncertainty of pi



cat("Estimated pi:",pi_val,"±",pi_se)

#Visualizing
plot(x, y, col = ifelse(innercircle, "blue", "red"), pch = 16, cex = 0.5,
```

```
    xlab = "X", ylab = "Y", main = sprintf("Monte Carlo Estimation of π\nEstimate: %.4f
± %.4f", pi_val, pi_se))
symbols(0, 0, circles = 1, inches = FALSE, add = TRUE, lwd = 2, col = "black")

legend("topright", legend = c("Inside Circle", "Outside Circle"),
       col = c("blue", "red"), pch = 16, bty = "n")
```