Homework #8 Solutions

Julian Jenkins, Michela Lio

April 1, 2025

Question #1 - Use Monte Carlo Integration to evaluate the following integrals. Parts (a) and (d) can be evaluated directly and do not need any changes to apply the Monte Carlo technique. However, transformations are needed for parts (b), (c) and (e).

```
n = 5000

# For each problem:
# Generate n uniform RVs
# Calculate X based on the specific integral
# return the mean value of X



# For parts (d) and (e) you need to generate a pair of uniform random variables since you are trying to

# Note: For part (e), you can either use an if/else statement to check if y<x or try something like:
# MCe = ...fill in...
# Ind = Y < X
# MCe = MCe*Ind


N = 5000

# Part (a)
x_a = runif(N)
integral_a = mean((1 - x_a^2)^(3/2)) * (1 - 0)
cat("Estimate for integral (a):", integral_a, "\n")
```

```
## Estimate for integral (a): 0.5976214
```

```
# Part (b)
x_b = runif(N, -2, 2)
integral_b = mean(exp(x_b + x_b^2)) * (2 - (-2))
cat("Estimate for integral (b):", integral_b, "\n")
```

```
## Estimate for integral (b): 91.33495
```

```
# Part (c)
x_c <- rnorm(N, 0, 1)
integral_c = mean(exp(-x_c^2)) * 2
cat("Estimate for integral (c):", integral_c , "\n")
```

```
## Estimate for integral (c): 1.160167
```

```r
# Part (d)
x_d = runif(N, 0, 1)
y_d = runif(N, 0, 1)
integral_d <- mean(exp((x_d + y_d)^2))
cat("Estimate for integral (d):", integral_d, "\n")
```

```
## Estimate for integral (d): 4.797453
```

```r
# Part (e)
x_e = rexp(N, 1)
y_e = runif(N, 0, x_e)
integral_e <- mean(exp(-(x_e + y_e)))
cat("Estimate for integral (e):", integral_e, "\n")
```

```
## Estimate for integral (e): 0.4059524
```

Question #2. This question asks you to approximate the value of $Cov(U, e^U)$ when $U \sim Unif(0, 1)$. Direct calculation shows that this is equal to $E(Ue^U) - \frac{1}{2}(e-1)$, so your goal should be to evaluate $E(Ue^U)$ and then subtract $\frac{1}{2}(e-1)$

```r
# Generate values from the uniform distribution
# Evaluate U*e^U
U = runif(10000)
X = mean(U * exp(U))
e_eU = exp(1) - 1
e_U = 1/2
Y = X - e_eU * e_U
Z = 0.140859

cat("Simulated Covariance: ", Y, "\n")
```

```
## Simulated Covariance:  0.1441949
```

```r
cat("True Value: ", Z, "\n")
```

```
## True Value:  0.140859
```

```r
cat("Difference: ", Y - Z, "\n")
```

```
## Difference:  0.003335876
```

Comment on how closely your answer compares to the true value My answer is pretty close to the true value as the difference is usually around .01. One way to make my answer more accurate is to increase the number of samples I use.

Question #3. This question asks you to calculate Monte Carlo estimates of an integral based on two different sample sizes, N=100 and N=1000 and then to compare the variance of your estimators as the sample size increases.

```r
estimates = function(n) {
  x = runif(n)
  y = exp(exp(x))
  return(mean(y))
}
# Part (a) - Use N = 100 to estimate the value of the integral


# Repeat this process 1000 times (loop) so that you have 1000 estimates of the integral
# Print out the mean and variance of your 1000 Monte Carlo estimates
```

```r
Z.100 = c()
for (i in 1:1000) {
  Z.100 = c(Z.100, estimates(100))
}

cat("Mean of 1000 Monte Carlo estimates at N = 100:", mean(Z.100))
```

```
## Mean of 1000 Monte Carlo estimates at N = 100: 6.3116
```

```r
cat("Variance of 1000 Monte Carlo estimates at N = 100:", var(Z.100))
```

```
## Variance of 1000 Monte Carlo estimates at N = 100: 0.1099674
```

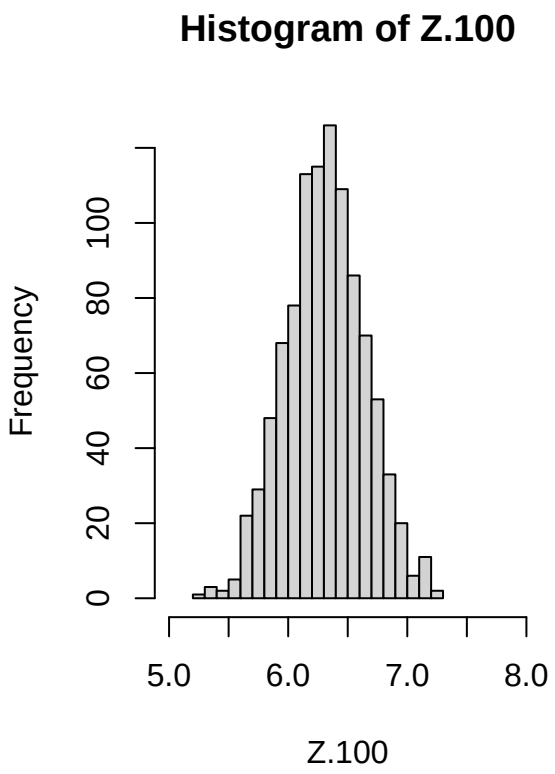
```r
# Part (b) - Use N = 1000 to estimate the value of the integral

# Repeat this process 1000 times (loop) so that you have 1000 estimates of the integral
# Print out the mean and variance of your 1000 Monte Carlo estimates
Z.1000 = c()
for (i in 1:1000) {
  Z.1000 = c(Z.1000, estimates(1000))
}

cat("\nMean of 1000 Monte Carlo estimates at N = 1000:", mean(Z.1000))
```

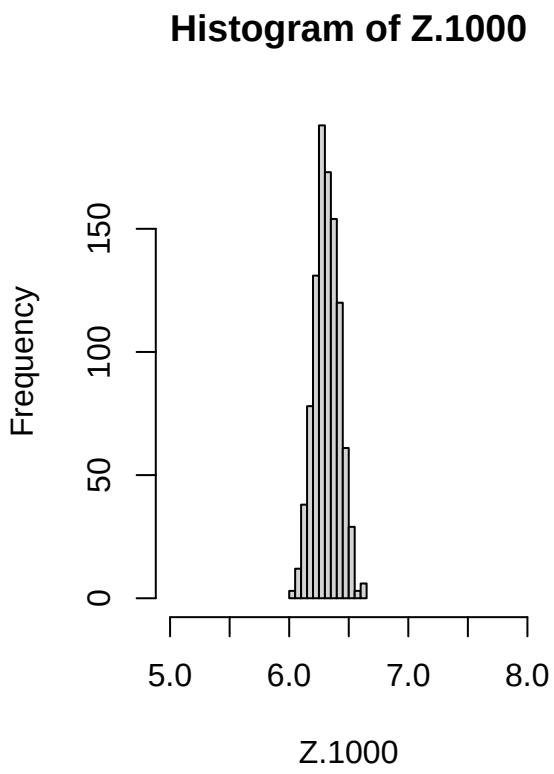
```
##
## Mean of 1000 Monte Carlo estimates at N = 1000: 6.316247
```

```r
cat("\nVariance of 1000 Monte Carlo estimates at N = 1000:", var(Z.1000))
```

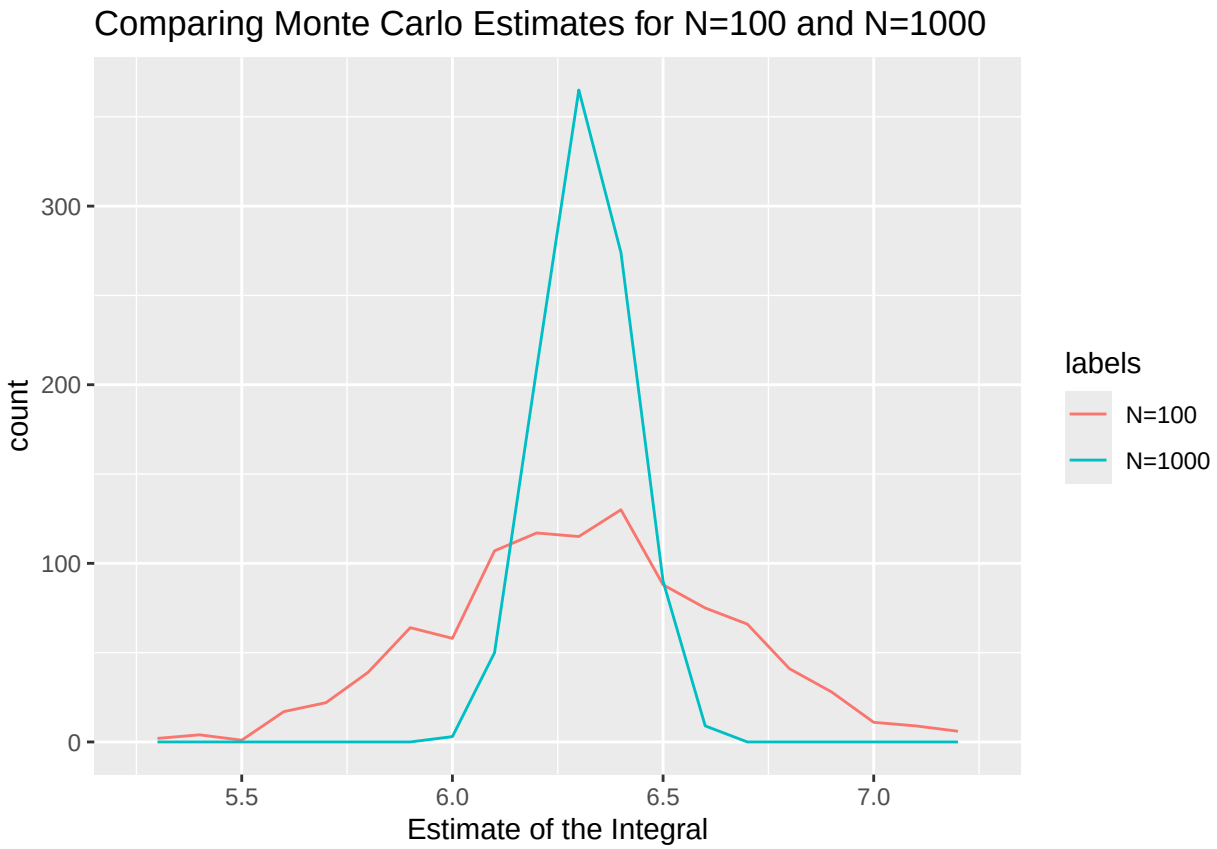
```
##
## Variance of 1000 Monte Carlo estimates at N = 1000: 0.01057032
```

```r
# Let's make histograms of our sampled values.
# Suppose that our estimated values are stored in vectors called Z.100 and Z.1000.  Then we can use:
par(mfrow=c(1,2))
hist(Z.100, breaks=20, xlim=c(5,8))
hist(Z.1000, breaks=20, xlim=c(5,8))
```






```
library(ggplot2)
values = c(Z.100, Z.1000)  #This assumes your values are stored in vectors called Z.100 and Z.1000
labels = c(rep("N=100", 1000), rep("N=1000", 1000))

data = data.frame(values, labels)

ggplot(data = data, aes(x = values, color = labels)) +
  geom_line(stat = "bin", binwidth = 0.1) +
  labs(title = "Comparing Monte Carlo Estimates for N=100 and N=1000", x= "Estimate of the Integral")
```



Comment on the means and variances of the two histograms. The means and of the two histograms are extremely similar to each other with both having a mean of around 6.3 but the variance of the N = 1000 histogram is a lot smaller than N = 100 due to the increase of N.

Question #4. Use Monte Carlo integration with antithetic variables to estimate

$$\int_0^1 \frac{exp^{-x}}{1+x^2} dx$$

```
# part (a) - Produce a single estimate using the regular Monte Carlo technique
# use n = 1000
normalEst = function() {
N = 1000
X = runif(N)
Y = exp(-X) / (1 + X^2)
return(mean(Y))
}
cat("Monte Carlo estimate of the integral: ", normalEst(), "\n")
```

```
## Monte Carlo estimate of the integral:  0.523573
```

```
# Part (b) - Produce a single estimate using antithetic variables U and (1-U)
# This time, you only need to sample 500 values from the uniform distribution
Anti = function() {
N = 500
X = runif(N)
Y = exp(-X) / (1 + X^2) + exp(-(1 - X)) / (1 + (1 - X)^2)
return(mean(Y) / 2)
```

```
}
cat("Monte Carlo estimate of the integral using Antithetic: ", Anti(), "\n")
```

```
## Monte Carlo estimate of the integral using Antithetic:  0.5264749
```

```
# part (c) - repeat parts (a) and (b) 1000 times so that you have 1000 estimates using each approach.
# Then calculate the percent reduction in variance as (V(old) - V(new)) / V(old)
Z.Normal = c()
Z.Anti = c()
for (i in 1:1000) {
  Z.Normal = c(Z.Normal, normalEst())
  Z.Anti = c(Z.Anti, Anti())
}
Reduct = (var(Z.Normal) - var(Z.Anti)) / var(Z.Normal)
cat("Percent of Variance without Reduction:", Reduct * 100, "\n")
```

```
## Percent of Variance without Reduction: 96.16331
```

Comment on the percent reduction in the variance.
The percent reduction in the variance is usually less than 5 percent.

Question #5 - Importance Sampling. Importance sampling works best when the variance of $\frac{g(x)}{f(x)}$ is as small as possible. Run the code below to evaluate the variance of this ratio for two different "importance" functions. If the standard deviation of the standard Monte Carlo estimator is 0.244, what can you say about the quality of these two importance functions? Which (if either) would you want to use to estimate our integral? *Note*: All you have to do for this question is run the code and answer the question. No additional coding is needed.

```
# Run the following code
m = 10000
g = function(x)
{
  exp(-x - log(1+x^2))*(x>0)*(x<1)
}

# f2
x = rcauchy(m)
i = c(which(x>1), which(x<0))
x[i] = 2  # to catch overflow errors in g(x)
fg = g(x)/dcauchy(x)
cat("The variance of the f2 importance function is", sd(fg), "\n")
```

```
## The variance of the f2 importance function is 0.9390023
```

```
#f3
U = runif(m)
x = -log(1-U*(1-exp(-1)))
fg = g(x)/(exp(-x)/(1-exp(-1)))
cat("The variance of the f3 importance function is", sd(fg))
```

```
## The variance of the f3 importance function is 0.09662878
```

Comment on the two potential importance functions. Which (if either) should you consider using? Why?
We should consider using the f3 importance function because the variance of g(x)/f(x) is smaller than the Monte Carlo estimator variance. The f2 function is .95 which is a lot bigger than the Monte Carlo estimator variance so we should not consider using it.

Question #6 What sample size is needed for a given error of estimation? Suppose $X \sim N(0,1)$ and we are

looking to estimate $E(cos(X))$ to three digits of accuracy.

```
# Figure out the correct value for N
N = 1000

X = rnorm(N)
Y = cos(X)
sigma2 = var(Y)

error = .0005
newN = ceiling(sigma2 / error^2)

Xnew = rnorm(newN)
Ynew = cos(Xnew)
estimate = mean(Ynew)
std_error = sd(Ynew) / sqrt(newN)
# Generate N standard normal RVs
# Print out the mean: E(cos(X))
cat("Computed Sample Size N:", newN, "\n")
```

```
## Computed Sample Size N: 795086
```

```
cat("Monte Carlo Estimate of E[cos(X)]:", round(estimate, 3), "\n")
```

```
## Monte Carlo Estimate of E[cos(X)]: 0.606
```

```
cat("Standard Error:", std_error, "\n")
```

```
## Standard Error: 0.0005016655
```

Question #7 - Monte Carlo Estimates of the Bias. In this problem we are looking to estimate the correlation of two random variables. X has a standard normal distribution and Y is a function of X along with a second standard normal RV η. A bit of algebra shows that the correlation of X and Y should be ρ. (a) Calculate the bias of our estimator when $\rho = -0.5$.

```
rho = -0.5
n=10
N = 1000

# Write a loop
  # Generate X and eta as standard normal RVs (n=10)
  # Calculate Y as given in the problem
  # Calculate the sample correlation of X and Y.  Save this value in a vector
cor_estimates = c()
for (i in 1:N) {
  X = rnorm(n)
  eta = rnorm(n)
  Y <- rho * X + eta * sqrt(1 - rho^2)
  cor_estimates = c(cor_estimates, cor(X, Y))

}
# Outside of the loop, calculate the mean of your sample correlation vector.
# Compare this value to the true value of rho, -0.5
mean_cor = mean(cor_estimates)
bias = mean_cor - rho
cat("Estimated mean correlation:", mean_cor, "\n")
```

```
## Estimated mean correlation: -0.4792067
```

```r
cat("Bias of the correlation estimator:", bias, "\n")
```

```
## Bias of the correlation estimator: 0.02079334
```

Part (b) - Calculate the variance
Part (c) - Calculate $MSE = bias^2(\hat{\rho}) + V(\hat{\rho})$

```r
# Calculate the variance of your vector from part (a) and then the MSE
cat("Variance of the estimator:", var(cor_estimates), "\n")
```

```
## Variance of the estimator: 0.07430426
```

```r
cat("Mean Squared Error (MSE):", var(cor_estimates) + bias^2)
```

```
## Mean Squared Error (MSE): 0.07473662
```