Homework 4 Group 09

Shrasti Dwivedi, Jayansh Jain, Karan Pratap Lohiya, Sunanda Maity,Shreya Saha

Problem 1

Evaluate the following integral in the asymptotic limit as $n \to \infty$:

$$I_n = \lim_{n\to\infty} \int_0^1 \dots \int_0^1 \frac{x_1^{101} + x_2^{101} + \dots + x_n^{101}}{x_1 + x_2 + \dots + x_n} dx_1 \dots dx_n. \tag{1}$$

Solution

We analyze the behavior of the given integral using the Law of Large Numbers (LLN) and properties of expectations.

Let $X_1, X_2, \dots, X_n$ be independent and identically distributed (i.i.d.) random variables, each following a uniform distribution on $[0, 1]$, i.e., $X_i \sim U(0, 1)$. The integral can be interpreted as:

$$I_n = \lim_{n\to\infty} \mathbb{E}\left[\frac{X_1^{101} + X_2^{101} + \dots + X_n^{101}}{X_1 + X_2 + \dots + X_n}\right]. \tag{2}$$

For large n, by the strong law of large numbers, we approximate the numerator and denominator using their expected values:

$$\frac{1}{n}\sum_{i=1}^{n} X_i \approx \mathbb{E}[X] = \int_0^1 x dx = \frac{1}{2}, \tag{3}$$

$$\frac{1}{n}\sum_{i=1}^{n} X_i^{101} \approx \mathbb{E}[X^{101}] = \int_0^1 x^{101} dx = \frac{1}{102}. \tag{4}$$

Thus, the fraction inside the expectation simplifies as:

$$\frac{\frac{1}{n}\sum_{i=1}^{n} X_i^{101}}{\frac{1}{n}\sum_{i=1}^{n} X_i} \approx \frac{\mathbb{E}[X^{101}]}{\mathbb{E}[X]} = \frac{\frac{1}{102}}{\frac{1}{2}} = \frac{1}{51}. \tag{5}$$

By the Dominated Convergence Theorem (DCT), we exchange the expectation and limit, yielding:

$$I_n = \lim_{n\to\infty} \mathbb{E}\left[\frac{X_1^{101} + X_2^{101} + \dots + X_n^{101}}{X_1 + X_2 + \dots + X_n}\right] = \frac{1}{51}. \tag{6}$$

Final Answer

$$\lim_{n\to\infty}\int_0^1 \dots \int_0^1 \frac{x_1^{101}+x_2^{101}+\dots+x_n^{101}}{x_1+x_2+\dots+x_n}dx_1\dots dx_n = \frac{1}{51}. \tag{7}$$

Monte Carlo-Copula Integration Approach

We aim to estimate the expected value of:

$$E\left[\frac{\sum_{i=1}^{N} X_i^{101}}{\sum_{i=1}^{N} X_i}\right]$$

where $X_i \sim U(0,1)$. The copula-based Monte Carlo approach provides a structured way to simulate dependent random variables and evaluate this expectation.

Copula Transformation

A **copula** captures the dependency structure between random variables without affecting their marginal distributions. We transform the uniform random variables using an appropriate copula function.

Let: - $U_i \sim C(u_1, u_2, ..., u_n)$, a copula-based sample. - Using the inverse transform:

$$X_i = \Phi^{-1}(U_i)$$

where Φ^{-1} is the inverse cumulative distribution function (CDF) of the normal distribution.

This ensures that the samples follow a desired dependence structure.

Algorithm

1. Generate dependent samples $U_1, ..., U_n$ from a copula C.
2. Transform them using:

$$X_i = \Phi^{-1}(U_i)$$

3. Compute the Monte Carlo integral estimate:

$$I_N = \frac{\sum_{i=1}^{N} X_i^{101}}{\sum_{i=1}^{N} X_i}$$

4. Repeat for different sample sizes and analyze convergence.

Code for Monte Carlo - Copula Integration :

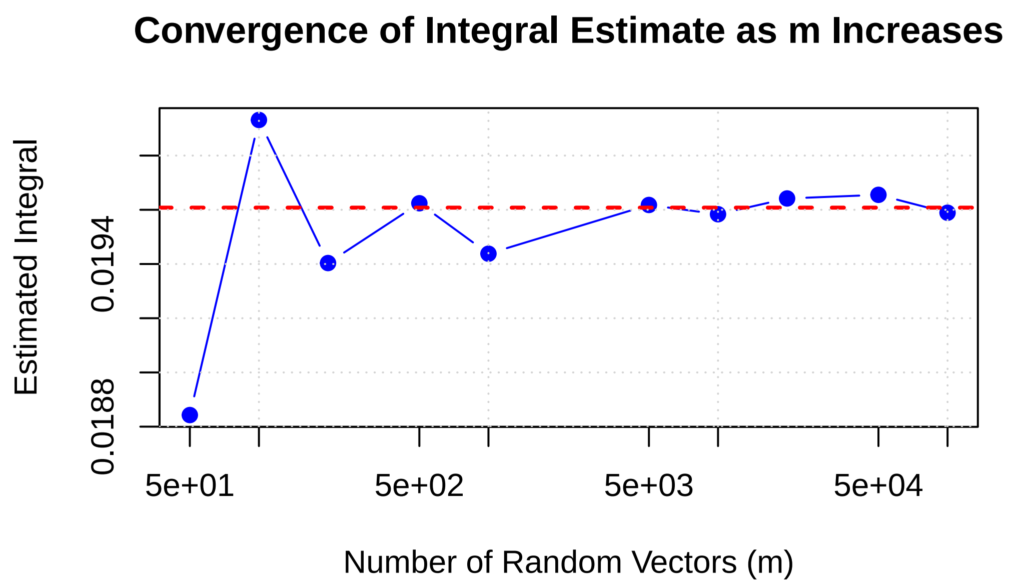
```
set.seed(123)
# Number of elements in each random vector (n is fixed to 1000)
```

```r
n <- 1000

# Vector m representing different values for the number of random vectors to generate (from small to la
m <- c(50, 100, 200, 500, 1000, 5000, 10000, 20000, 50000, 100000)

# Create an empty vector to store the integral estimates for each value of m
Ans <- numeric(length(m))

# Loop through each value of m
for(i in 1:length(m)) {

  # Get the current value of N (number of random vectors to generate for the current m)
  N <- m[i]

  # Create an empty vector to store the calculated integral estimates for each random vector
  b <- numeric(1000)

  # Loop through 1000 times (Monte Carlo simulations) to calculate the integral estimate
  for(j in 1:1000) {

    # Generate a random vector of size N with values uniformly distributed between 0 and 1
    a <- runif(N)

    # Apply the integrand: (sum of a_i^101) / (sum of a_i) for the generated vector a
    b[j] <- sum(a^101) / sum(a)
  }

  # Take the average of all 1000 simulations for the current value of N
  Ans[i] <- sum(b) / 1000
}

# Print the calculated estimates for each value of m
print(Ans)
```

```
 [1] 0.01884277 0.01993148 0.01940352 0.01962415 0.01943799 0.01961782
 [7] 0.01958354 0.01964182 0.01965517 0.01958914
```

```r
# Plot the estimates of the integral as a function of m
plot(m, Ans, type = "b", col = "blue", pch = 19,
     xlab = "Number of Random Vectors (m)",
     ylab = "Estimated Integral",
     main = "Convergence of Integral Estimate as m Increases",
     log = "x")  # log scale on x-axis for better visualization
```

```
# Add a horizontal line at y = 1/51
abline(h = 1/51, col = "red", lwd = 2, lty = 2)

# Adding grid lines for better clarity
grid()
```



Tabulating the absolute difference from theoretical value

```
# Define the reference value 1/51
reference_value <- 1 / 51

# Compute the distance from the reference value for each m
distances <- abs(Ans - reference_value)

# Create a table with the values of m, the estimated integral, and the distance from 1/51
distance_table <- data.frame(
  m = m,
  Integral_Estimate = Ans,
  Distance_from_theoretical_value = distances
)

# Print the table
print(distance_table)
```

```
       m Integral_Estimate Distance_from_theoretical_value
1  5e+01        0.01884277                    7.650747e-04
2  1e+02        0.01993148                    3.236360e-04
3  2e+02        0.01940352                    2.043219e-04
4  5e+02        0.01962415                    1.631106e-05
5  1e+03        0.01943799                    1.698528e-04
6  5e+03        0.01961782                    9.975737e-06
7  1e+04        0.01958354                    2.430414e-05
8  2e+04        0.01964182                    3.397858e-05
9  5e+04        0.01965517                    4.732404e-05
10 1e+05        0.01958914                    1.870675e-05
```

Here , theoretical value refer to 1/51.

Problem 2

Download a bivariate data and fit into a simple linear regression model. For this given data, estimate the unknown parameters of the above mentioned model using the LSE and the LAD techniques. Compare the performance of the LSE and the LAD estimators.

Link for Dataset : https://github.com/chandanverma07/DataSets/blob/master/weight-height.csv

Regression analysis is a used to model the relationship between a dependent variable and one or more independent variables. In this dataset, we analyze the relationship between Height and Weight using two regression techniques:

1. Ordinary Least Squares (LSE): Minimizes the sum of squared residuals.
2. Least Absolute Deviations (LAD): Minimizes the sum of absolute residuals, making it more robust to outliers.

We compare these two approaches by estimating the regression parameters and evaluating their sensitivity to outliers.

Methodology

We use a dataset containing height and weight measurements of individuals. The analysis follows these steps:

1. Fit a simple linear regression model using LSE (via `lm()` function).
2. Fit a regression model using LAD (via `rq()` function from the `quantreg` package).
3. Compare the estimated parameters (intercept & slope) and residual errors (RSS for LSE, LAD-RSS for LAD).
4. Analyze residuals and evaluate the impact of outliers by adding artificial outliers and refitting both models.
5. Visualize the regression lines and residual distributions.

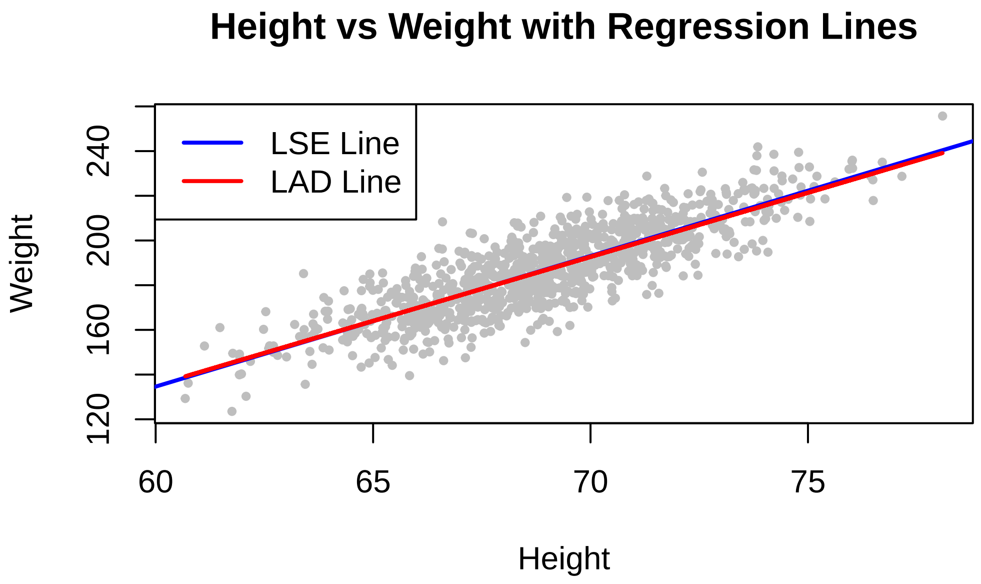
Code:

```r
# Load necessary packages

library(quantreg)
```

```
Warning: package 'quantreg' was built under R version 4.4.3

Loading required package: SparseM

Warning: package 'SparseM' was built under R version 4.4.3
```

```r
# Load your data
data <- read.csv("weight-height.csv")
Bdata <- data[1:1000, 2:3]  # Use first 1000 rows and columns 2 and 3 for Height and Weight
x <- Bdata$Height
y <- Bdata$Weight

# 1. Fit the model using LSE (Ordinary Least Squares)
model_lse <- lm(Weight ~ Height, data = Bdata)

# 2. Fit the model using LAD (Least Absolute Deviations)
model_lad <- rq(Weight ~ Height, data = Bdata)

# 3. Extract coefficients for both models
beta0_lse <- coef(model_lse)[1]
beta1_lse <- coef(model_lse)[2]
beta0_lad <- coef(model_lad)[1]
beta1_lad <- coef(model_lad)[2]

# 4. Plot the scatter plot and regression lines
plot(Bdata$Height, Bdata$Weight,
     main = "Height vs Weight with Regression Lines",
     xlab = "Height",
     ylab = "Weight",
     pch = 19, col = "gray", cex = 0.5)

# Add LSE regression line
abline(model_lse, col = "blue", lwd = 2)



# Add LAD regression line
lines(Bdata$Height, predict(model_lad), col = "red", lwd = 2)

# Add a legend
```

```
legend("topleft", legend = c("LSE Line", "LAD Line"), col = c("blue", "red"), lwd = 2)
```



Tabulating slope and intercept both for LSE and LAD

```
#graph

# 6. Numerically compare the goodness of fit
residuals_lse <- Bdata$Weight - predict(model_lse)
residuals_lad <- Bdata$Weight - predict(model_lad)
# Sum of squared residuals for LSE (RSS)
rss_lse <- sum(residuals_lse^2)

# Sum of absolute residuals for LAD (LAD-RSS)
lad_rss <- sum(abs(residuals_lad))

# Sum of squared residuals for LSE (RSS)
rss_lse <- sum(residuals_lse^2)

# Sum of absolute residuals for LAD (LAD-RSS)
lad_rss <- sum(abs(residuals_lad))

# Create a table with the results
results_table <- data.frame(
  Metric = c("Beta0 (Intercept)", "Beta1 (Slope)", "Sum of Squared Residuals (RSS)", "Sum of Absolute R
  LSE = c(beta0_lse, beta1_lse, rss_lse, NA),
```

```r
  LAD = c(beta0_lad, beta1_lad, NA, lad_rss)
)

# Print the table
cat("\n--- Results Table ---\n")
```

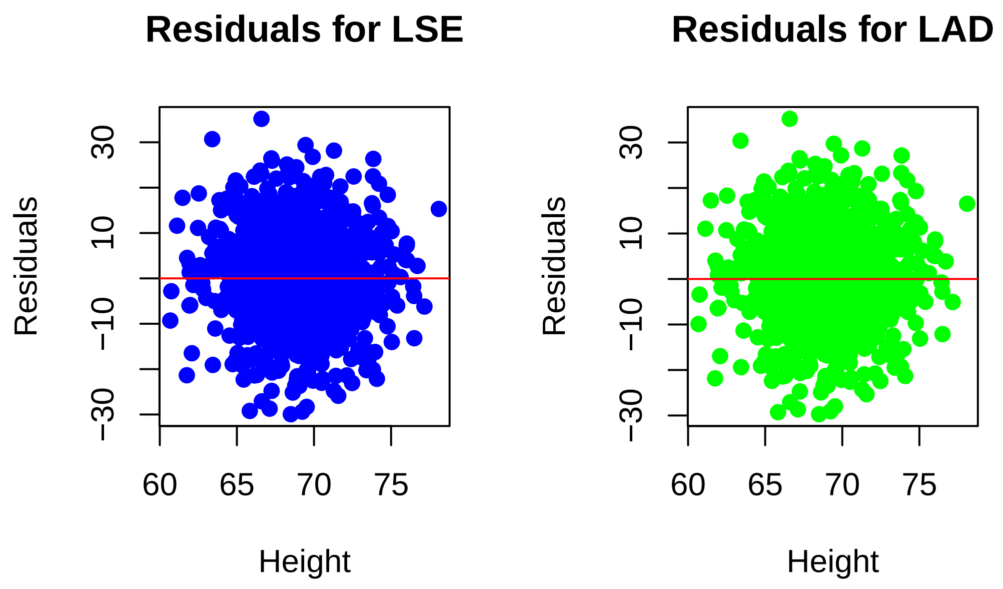
```
--- Results Table ---
```

```r
print(results_table)
```

```
                                Metric           LSE         LAD
1                   Beta0 (Intercept)   -216.028817 -209.002445
2                       Beta1 (Slope)      5.844271    5.738394
3      Sum of Squared Residuals (RSS) 109610.765896          NA
4 Sum of Absolute Residuals (LAD-RSS)            NA 8362.398697
```

Plotting residuals For LSE and LAD

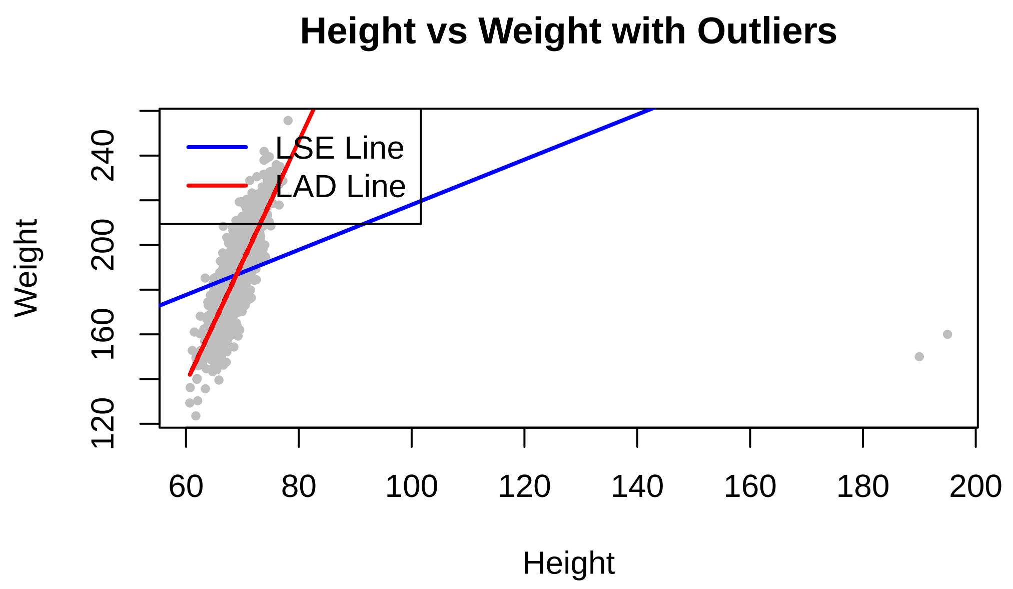
```r
# 5. Plot the residuals for LSE and LAD side by side
par(mfrow = c(1, 2))  # Set up a 1x2 plotting layout

# Plot residuals for LSE
residuals_lse <- Bdata$Weight - predict(model_lse)
plot(Bdata$Height, residuals_lse,
     main = "Residuals for LSE",
     xlab = "Height", ylab = "Residuals",
     pch = 19, col = "blue")
abline(h = 0, col = "red")  # Add a horizontal line at 0 for reference

# Plot residuals for LAD
residuals_lad <- Bdata$Weight - predict(model_lad)
plot(Bdata$Height, residuals_lad,
     main = "Residuals for LAD",
     xlab = "Height", ylab = "Residuals",
     pch = 19, col = "green")
abline(h = 0, col = "red")  # Add a horizontal line at 0 for reference
```



```r
# Reset plotting layout
par(mfrow = c(1, 1))
```

Adding Outliers for Comparison

```r
# Adding outliers
set.seed(42)
Bdata_with_outliers <- rbind(Bdata, data.frame(Height = c(190, 195), Weight = c(150, 160)))

# Fit models with the outlier data
model_lse_outliers <- lm(Weight ~ Height, data = Bdata_with_outliers)
model_lad_outliers <- rq(Weight ~ Height, data = Bdata_with_outliers)

# Plot the data with outliers
plot(Bdata_with_outliers$Height, Bdata_with_outliers$Weight,
     main = "Height vs Weight with Outliers",
     xlab = "Height", ylab = "Weight", pch = 19, col = "gray", cex = 0.5)

# Add LSE regression line with outliers
abline(model_lse_outliers, col = "blue", lwd = 2)

# Add LAD regression line with outliers
lines(Bdata_with_outliers$Height, predict(model_lad_outliers), col = "red", lwd = 2)

# Add a legend
legend("topleft", legend = c("LSE Line", "LAD Line"), col = c("blue", "red"), lwd = 2)
```



Observations from the Comparison

- **LSE (Least Squares) is sensitive to outliers**, leading to larger deviations when extreme values are present.
- **LAD (Least Absolute Deviations) is robust to outliers**, as it minimizes absolute errors rather than squared errors.
- **Residuals from LSE tend to have larger variations**, whereas LAD residuals are more uniformly distributed.