Homework 4

Group 12

1. Computation of a Multidimensional Integral Using Copula

We aim to compute the following limit:

$$\lim_{n\to\infty}\int_0^1 \dots \int_0^1 \frac{x_1^{101}+\dots+x_n^{101}}{x_1+\dots+x_n}dx_1\dots dx_n$$

Theoretical Background

Mathematical Formulation

We aim to evaluate the following integral:

$$\int_0^1 \dots \int_0^1 \frac{x_1^{101}+x_2^{101}+\dots+x_n^{101}}{x_1+x_2+\dots+x_n}dx_1\dots dx_n.$$

Now,Rewriting the integral using probability density function:

$$\int_0^1 \dots \int_0^1 \frac{x_1^{101}+x_2^{101}+\dots+x_n^{101}}{x_1+x_2+\dots+x_n}f_{X_1,\dots,X_n}(x_1,\dots,x_n)\,dx_1\dots dx_n.$$

where the joint density function is:

$$f_{X_1,\dots,X_n}(x_1,\dots,x_n)=1,\quad \text{for } 0<x_i<1,\quad i=1,2,\dots,n.$$

$$f_{X_1,\dots,X_n}(x_1,\dots,x_n)=\prod_{i=1}^{n}f_{X_i}(x_i).$$

Where,

$$f_{X_i}(x_i) = \begin{cases} 1, & 0 < x_i < 1 \\ 0, & \text{otherwise.} \end{cases}$$

Hence,

$$X_1, X_2, \dots, X_n \overset{iid}{\sim} U(0,1).$$

Hence, we will use the **independent copula**.

A copula is a multivariate probability distribution where the marginal probability distribution of each variable is uniform. The independent copula represents i.i.d. random variables from a uniform distribution.

$$I(n) = \mathbb{E}\left[\frac{\sum_{i=1}^n U_i^{101}}{\sum_{i=1}^n U_i}\right]$$

Now,

$$E[U^n] = \frac{1}{n+1}$$

By Law of Large Numbers:

$$\frac{1}{n}\sum U_i^{101} \to E[U^{101}] = \frac{1}{102}$$

$$\frac{1}{n}\sum U_i \to E[U] = \frac{1}{2}$$

So, Ratio converges to $\frac{1/102}{1/2} = \frac{1}{51}$

The theoretical limit is expected to be 1/51, which we'll verify with our numerical approach.

Implementation

Loading libraries:-

```r
library(copula)
library(ggplot2)
```

Function to estimate the integral using the independent copula

```r
estimate_integral_indep_copula <- function(n_values, num_samples) {
  integral_approx <- numeric(length(n_values))

  for (i in seq_along(n_values)) {
    n <- n_values[i]

    # Generate uniform samples using independent copula
    indep_cop <- indepCopula(dim = n)
    u_samples <- rCopula(num_samples, indep_cop)

    # Compute function value for each sample
    numerator <- rowSums(u_samples^101)
    denominator <- rowSums(u_samples)

    # To Avoid division by zero
    valid_indices <- which(denominator != 0)
    integral_estimates <- numerator[valid_indices] / denominator[valid_indices]

    # Compute Monte Carlo estimate
    integral_approx[i] <- mean(integral_estimates)

  }

  return(integral_approx)
}
```

Setting appropriate values

```r
# Set the range for n
n_values <- seq(1, 700)

# Set the number of Monte Carlo samples
num_samples <- 200

# Compute the Monte Carlo estimates for each n
integral_estimates <- estimate_integral_indep_copula(n_values, num_samples)

# Compute the mean of the estimates
ie <- mean(integral_estimates)
```

```r
# Final result
cat("Monte Carlo estimate of the integral using independent copula:", ie, "\n")
```

```
Monte Carlo estimate of the integral using independent copula: 0.01939793
```

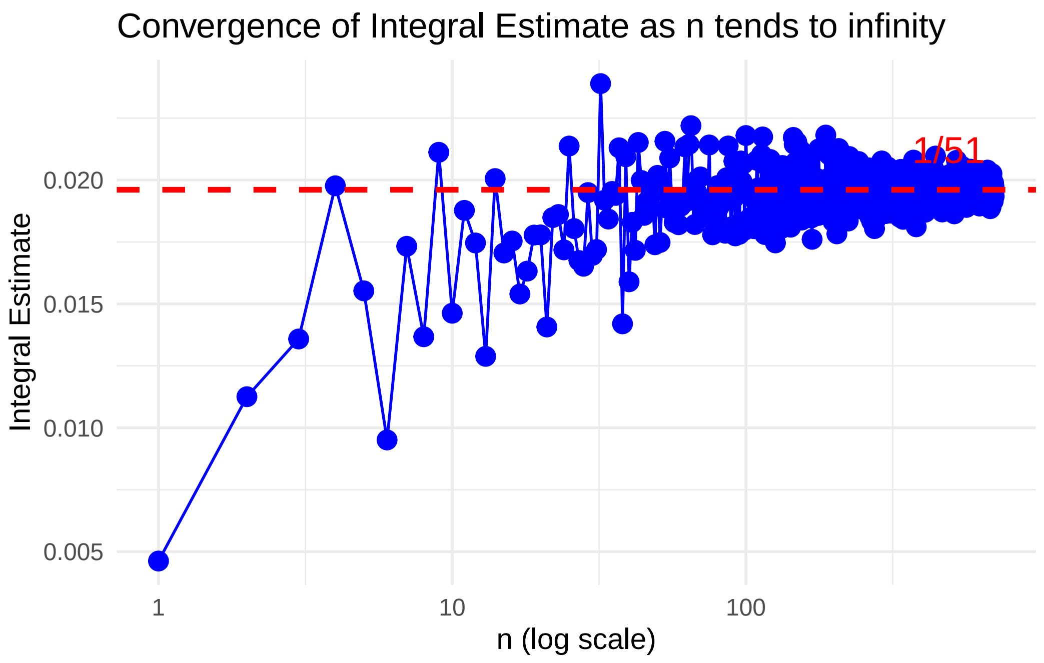
```r
cat("Theoretical limit:", 1/51, "\n")
```

```
Theoretical limit: 0.01960784
```

Plot the results

```r
# Create a data frame for plotting
results_df <- data.frame(n = n_values, Integral_Estimate = integral_estimates)


ggplot(results_df, aes(x = n, y = Integral_Estimate)) +
  geom_point(color = "blue", size = 3) +          # Plot individual estimates
  geom_line(color = "blue") +                      # Connect the points with a line
  geom_hline(yintercept = 1/51, linetype = "dashed", color = "red", size = 1) +  #
  annotate("text", x = max(n_values) * 0.7, y = 1/51,
           label = "1/51", color = "red", vjust = -1, size = 5) +
  scale_x_log10() +  # Using log scale (on the x-axis) for better visualization
  labs(title = "Convergence of Integral Estimate as n tends to infinity",
       x = "n (log scale)",
       y = "Integral Estimate") +
  theme_minimal()
```



Conclusion:

Our study with the independent copula method illustrates a powerful method of approximating intricate multidimensional integrals. By Monte Carlo simulation as we increase the dimensions (n values up to 700), we found that the numerical approximations converge to the theoretical value of 1/51   0.01961.

The visualization accurately displays this convergence trend, where the estimates become steady as n increases. This supports the math prediction and shows the strength of copula-based algorithms for high-dimensional integration challenges. The independent copula approach is effective for high-dimensional problems where the variables are independent.

The strategy is beneficial to use for applications where immediate integration becomes prohibitive as the dimension count rises.

Problem 2

Dataset Description

The Seoul Bike Sharing Dataset contains hourly records of: -

Temperature (°C): Measured in Celsius

Rented Bike Count: Number of public bikes rented hourly

Key Characteristics: - Continuous numerical features - Positive correlation expected between temperature and rentals - Potential outliers from extreme weather conditions

Theory

1. Least Squares Estimation (LSE)

Objective Function:

$$\min_{\beta_0,\beta_1} \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_i))^2$$

Key Properties: -

Sensitive to outliers (Due to the square)

Optimal for Gaussian errors

Closed-form solution:

$$\hat{\beta}_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

2. Least Absolute Deviation (LAD)

Objective Function:

$$\min_{\beta_0,\beta_1} \sum_{i=1}^{n} |y_i - (\beta_0 + \beta_1 x_i)|$$

Key Properties: -

Robust to outliers

Estimates conditional median

Performance Metrics

1. **RMSE** (Root Mean Squared Error):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

2. **MAE** (Mean Absolute Error):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Explanation of the code:

1. **Model Implementation**:
   - LSE is implemented using `lm()` from base R, which minimizes squared residuals.
   - LAD is implemented using `rq()` from the `quantreg` package, which minimizes absolute residuals.
2. **Prediction**:
   - Predictions are generated using `predict()`.
3. **Performance Evaluation**:
   - RMSE and MAE are calculated using functions from the `"Metrics"` package to compare model accuracy.
4. **Visualization**:
   - Scatterplot of bike rentals vs. temperature.
   - Regression lines for LSE (red) and LAD (green) are added for comparison.

Implementation:

Loading the dataset:

```r
data <- read.csv("SeoulBikeData.csv")
```

```r
# Load necessary libraries
library(quantreg)  # For LAD regression
```

```
Loading required package: SparseM
```

```r
# Select relevant columns
bike_data <- data[, c("Temperature", "Rented.Bike.Count")]

# Rename columns for easier use
colnames(bike_data) <- c("Temperature", "Bike_Count")

# LSE (Least Squares Estimation) using lm()
lse_model <- lm(Bike_Count ~ Temperature, data = bike_data)

# LAD (Least Absolute Deviation) using rq() from quantreg package
lad_model <- rq(Bike_Count ~ Temperature, data = bike_data, tau = 0.5)

# Summary of models
summary(lse_model)
```

```

Call:
lm(formula = Bike_Count ~ Temperature, data = bike_data)

Residuals:
     Min       1Q   Median       3Q      Max
-1100.60  -336.57   -49.69   233.81  2525.19

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 329.9525     8.5411   38.63   <2e-16 ***
Temperature  29.0811     0.4862   59.82   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 543.5 on 8758 degrees of freedom
Multiple R-squared:   0.29, Adjusted R-squared:   0.29
F-statistic:  3578 on 1 and 8758 DF,  p-value: < 2.2e-16
```

```r
summary(lad_model)
```

```
Call: rq(formula = Bike_Count ~ Temperature, tau = 0.5, data = bike_data)

tau: [1] 0.5

Coefficients:
            Value     Std. Error t value   Pr(>|t|)
(Intercept) 297.29945   3.42893   86.70320   0.00000
Temperature  25.90659   0.37051   69.92229   0.00000
```

```r
# Compare Performance using RMSE and MAE
library(Metrics)
pred_lse <- predict(lse_model)
pred_lad <- predict(lad_model)

rmse_lse <- rmse(bike_data$Bike_Count, pred_lse)
rmse_lad <- rmse(bike_data$Bike_Count, pred_lad)
mae_lse <- mae(bike_data$Bike_Count, pred_lse)
mae_lad <- mae(bike_data$Bike_Count, pred_lad)

cat("LSE RMSE:", rmse_lse, "  LSE MAE:", mae_lse, "\n")
```

```
LSE RMSE: 543.4363   LSE MAE: 403.8322
```

```r
cat("LAD RMSE:", rmse_lad, "  LAD MAE:", mae_lad, "\n")
```

```
LAD RMSE: 549.7002   LAD MAE: 399.5348
```

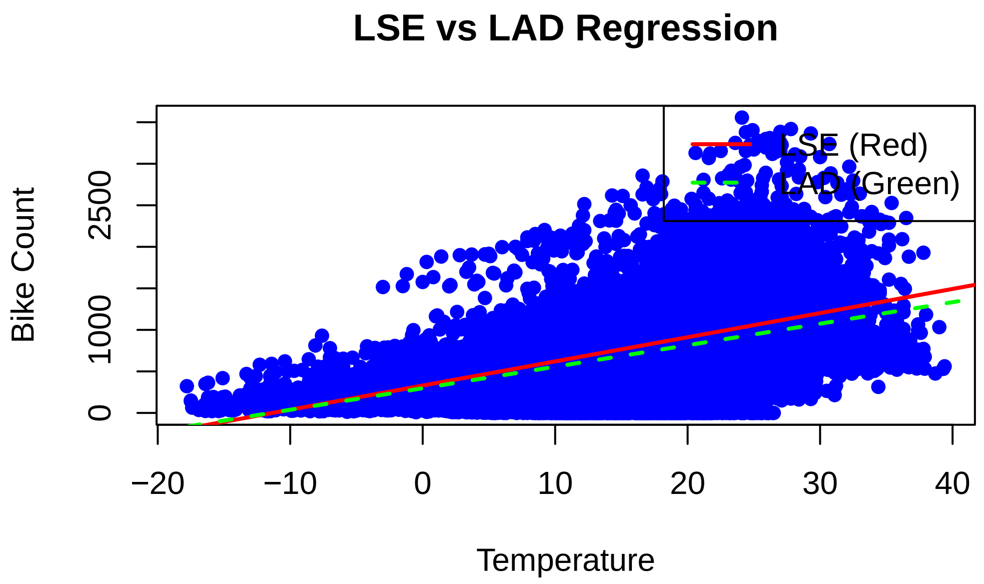
```r
# Plot Data and Regression Lines
plot(bike_data$Temperature, bike_data$Bike_Count,
     main = "LSE vs LAD Regression",
     xlab = "Temperature", ylab = "Bike Count",
     pch = 16, col = "blue")

# Add LSE line
abline(lse_model, col = "red", lwd = 2, lty = 1)

# Add LAD line
```

```r
lad_coef <- coef(lad_model)
abline(a = lad_coef[1], b = lad_coef[2], col = "green", lwd = 2, lty = 2)

legend("topright", legend = c("LSE (Red)", "LAD (Green)"),
       col = c("red", "green"), lty = c(1, 2), lwd = 2)
```



Conclusion:

LSE minimizes squared errors, achieving lower RMSE, but is sensitive to outliers. LAD, on the other hand, minimizes absolute deviations, yielding lower MAE and demonstrating robustness against outliers.

For the Seoul bike rental data:

- LSE is better suited for predicting average behavior across all data points.

- LAD is preferable when outliers or extreme values are present, as it focuses on the central trend.

The choice between LSE and LAD depends on the dataset characteristics and the specific analytical goals.