HW4_Group3

Group 3

2025-04-03

Question 1

In this report, we analyze the limit:

$$I = \lim_{n\to\infty} \int_0^1 \cdots \int_0^1 \frac{x_1^{101} + \cdots + x_n^{101}}{x_1 + \cdots + x_n} dx_1 \ldots dx_n$$

using **Monte Carlo integration** and **copula-based statistical methods**.

Copula Theory

A **copula** is a function that captures the dependence structure between random variables. Given n uniformly distributed random variables $X_1, X_2, \ldots, X_n$, their joint distribution can be expressed using a copula C:

$$F(x_1, x_2, \ldots, x_n) = C(F_1(x_1), F_2(x_2), \ldots, F_n(x_n))$$

where $F_i(x_i)$ are the marginal cumulative distribution functions (CDFs) of each variable. The copula separates **marginal behavior** from **dependence structure**, making it useful for modeling multivariate distributions.

Independence Copula

When variables are independent, the copula simplifies to:

$$C(u_1, u_2, \ldots, u_n) = u_1 u_2 \ldots u_n$$

which corresponds to the product of uniform marginals. This is the case we assume in our analysis.

Monte Carlo Estimation with Copulas

For independent random variables, we approximate the integral using Monte Carlo sampling:

1. **Sample** $X_1, X_2, \ldots, X_n$ **from** $U(0, 1)$.
2. **Evaluate the function** inside the integral.
3. **Take the average over multiple samples** to estimate the integral.

Convergence Analysis

We study **convergence** by: - **Fixing the sample size N and varying n.** - **Fixing n and increasing N.** - **Comparing estimates with the theoretical limit** $1/51$.

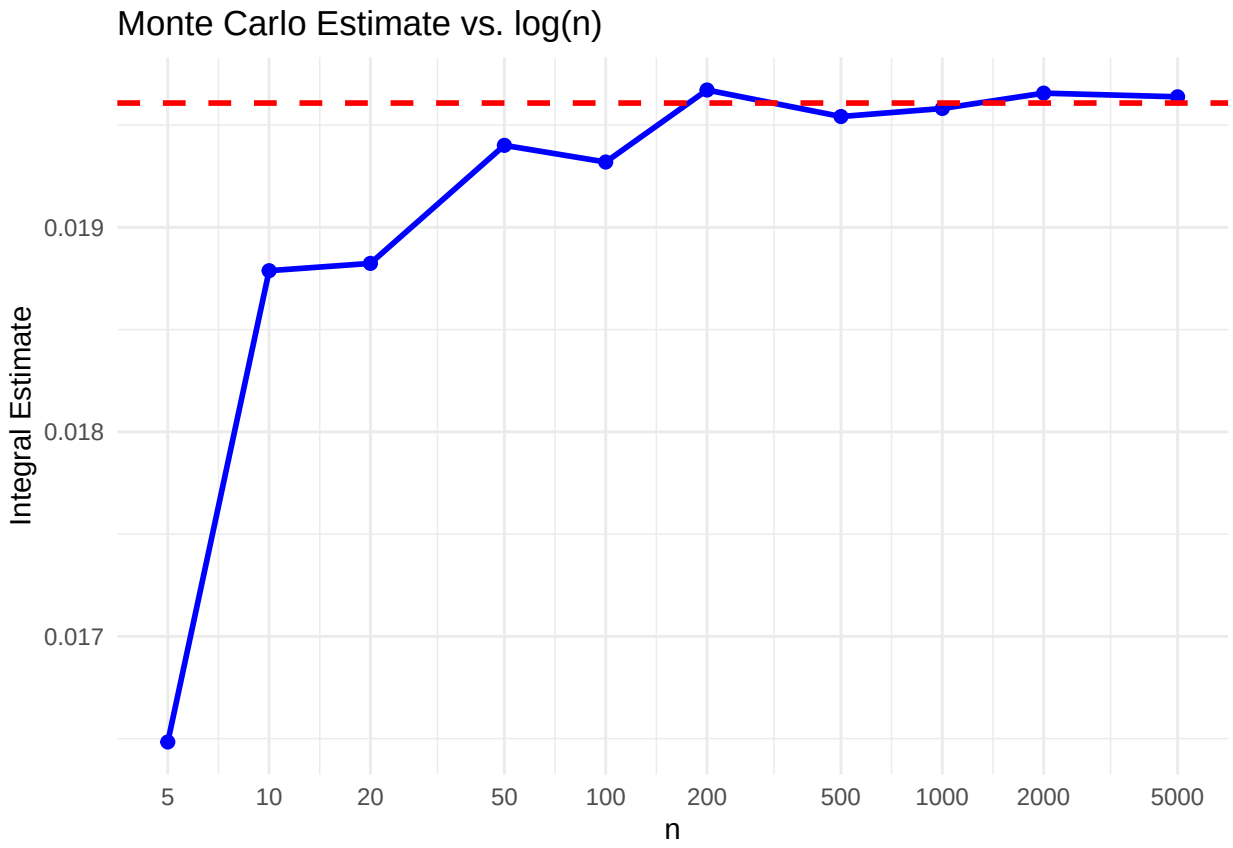
Monte Carlo Estimation for Different n

We now analyze how the integral estimate behaves as we vary n, while keeping the Monte Carlo sample size fixed at $N = 10^4$.

Since n can take large values, we use $\log(n)$ on the x-axis for better visualization.

The steps involved: 1. **Fix sample size** $N = 10^4$. 2. **Vary** n and compute Monte Carlo estimates. 3. **Plot results against** $\log(n)$ and compare with the true value $1/51$.

```r
# Load required libraries
library(ggplot2)

# Function to compute Monte Carlo estimate for given N and n
mc_integral <- function(N, n) {
  sum_values <- numeric(N)

  for (i in 1:N) {
    x <- runif(n)  # Generate independent U(0,1) samples
    numerator <- sum(x^101)
    denominator <- sum(x)
    sum_values[i] <- numerator / denominator
  }

  mean(sum_values)  # Return the Monte Carlo estimate
}

# Set parameters
set.seed(123)      # For reproducibility
N_fixed <- 1e4     # Fixed Monte Carlo sample size
n_values <- c(5, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000)  # Different n values
true_value <- 1/51  # Theoretical limit

# Compute Monte Carlo estimates for different n values
mc_estimates <- sapply(n_values, function(n) mc_integral(N_fixed, n))

# Create a data frame for plotting
df <- data.frame(n = n_values, Estimate = mc_estimates)

# Plot convergence with actual n values
ggplot(df, aes(x = n, y = Estimate)) +
  geom_line(color = "blue", size = 1) +
  geom_point(color = "blue", size = 2) +
  geom_hline(aes(yintercept = true_value), color = "red", linetype = "dashed", size = 1) +
  scale_x_continuous(trans = "log10", breaks = n_values) +  # Log scale for better spacing
  labs(title = "Monte Carlo Estimate vs. log(n)",
       x = "n",
       y = "Integral Estimate") +
  theme_minimal()
```



Monte Carlo Integration for Different N and n

We analyze how the Monte Carlo estimate of the integral behaves as we vary:

- The number of uniform random samples (N).
- The number of terms (n) in the integral computation.

To observe convergence, we take large n values and compare estimates for different N.

Steps:

1. Fix different values of N (e.g., $10^3, 10^4, 10^5$).
2. Vary n over a wide range.
3. Compute Monte Carlo estimates for each (N, n) pair.
4. Compare results with the true value $1/51$ using plots.

```
# Load required libraries
library(ggplot2)
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.4.2
```

```
##
## Attaching package: 'dplyr'
```

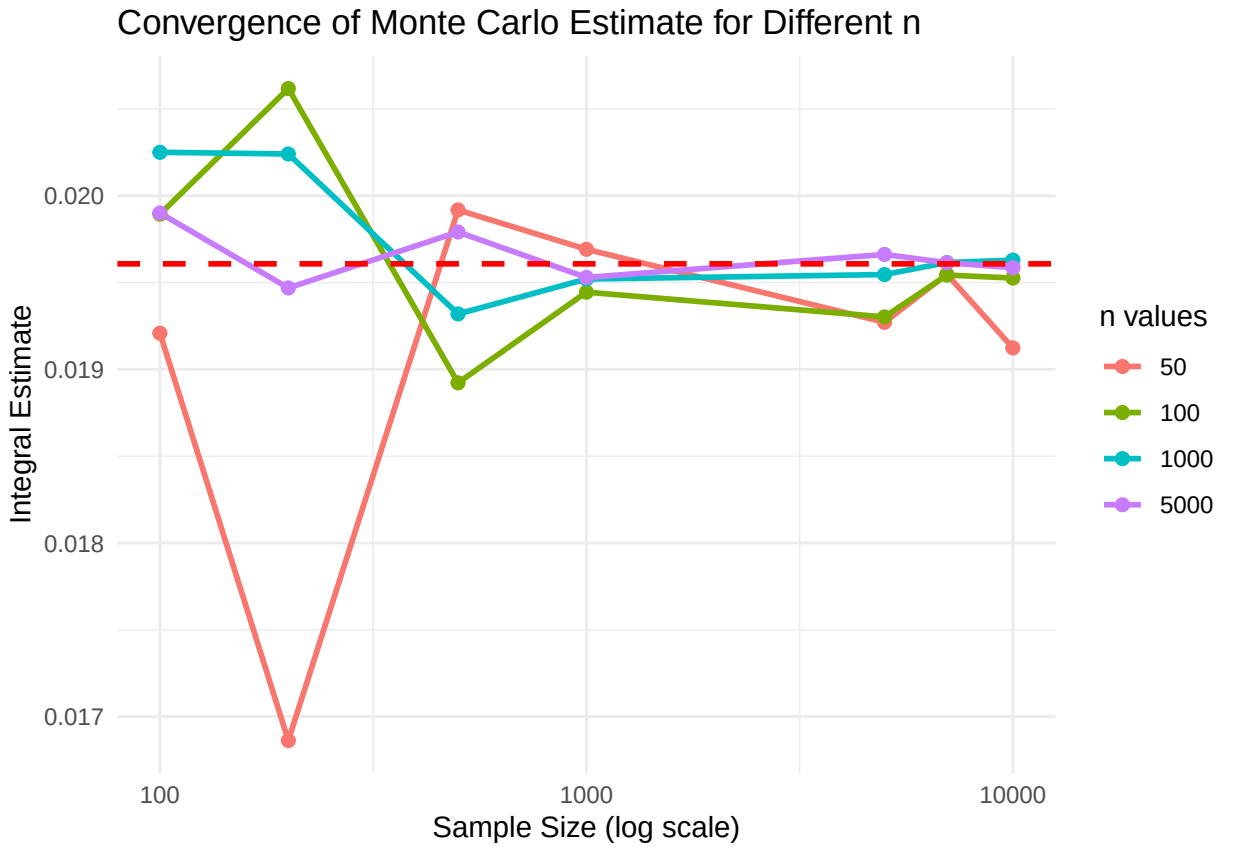
```
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
# Function to compute Monte Carlo estimate for given N and n
mc_integral <- function(N, n) {
  sum_values <- numeric(N)

  for (i in 1:N) {
    x <- runif(n)  # Generate independent U(0,1) samples
    numerator <- sum(x^101)
    denominator <- sum(x)
    sum_values[i] <- numerator / denominator
  }

  mean(sum_values)  # Return the Monte Carlo estimate
}

# Set parameters
set.seed(123)     # For reproducibility
n_values <- c(50, 100, 1000, 5000)  # Different n values
N_values <- c(100, 200, 500, 1000, 5000, 7000, 10000)  # Increasing sample sizes
true_value <- 1/51  # Theoretical limit

# Compute Monte Carlo estimates for each (N, n) pair
results <- expand.grid(N = N_values, n = n_values) %>%
  rowwise() %>%
  mutate(Estimate = mc_integral(N, n))

# Convert to data frame
df <- as.data.frame(results)

# Extract the estimate for the largest n and N
largest_n <- max(n_values)
largest_N <- max(N_values)
largest_estimate <- results %>%
  filter(n == largest_n & N == largest_N) %>%
  pull(Estimate)

# Print the estimated value for the largest n and N
cat("Estimated value for the largest n (n =", largest_n, ") and largest N
    (N =", largest_N, "):", largest_estimate, "\n")
```

```
## Estimated value for the largest n (n = 5000 ) and largest N
##     (N = 10000 ): 0.01958571
```

Estimated value for the largest n (n = 5000) and largest N (N = 10000): 0.01958571

```r
# Plot convergence for different n values
ggplot(df, aes(x = N, y = Estimate, color = factor(n), group = n)) +
  geom_line(size = 1) +
  geom_point(size = 2) +
  geom_hline(aes(yintercept = true_value), color = "red", linetype = "dashed", size = 1) +
  scale_x_log10() +  # Log scale for better visualization
  labs(title = "Convergence of Monte Carlo Estimate for Different n",
       x = "Sample Size (log scale)",
       y = "Integral Estimate",
       color = "n values") +
  theme_minimal()
```



Question 2

Introduction to Sunspot Data and Preprocessing

The sunspot dataset records the number of sunspots observed over time. Sunspots are temporary dark spots on the sun's surface caused by intense magnetic activity. This dataset is widely used in time series analysis and studying solar cycles.

For our analysis, we aim to examine whether a linear relationship exists between a transformed version of the data. Specifically, we define:

- **Position in Cycle**: A measure of time within a sunspot cycle.
- **Percentage of Peak**: The sunspot count as a percentage of the peak value in the cycle.

Data Preprocessing

We perform the following preprocessing steps: 1. **Cycle Identification**: Identify different sunspot cycles in the dataset. 2. **Peak Normalization**: Convert sunspot counts into a percentage of the peak value in each cycle. 3. **Expected Linearity**: We hope that **Percentage of Peak** exhibits a linear trend with **Position in Cycle**, making it suitable for linear regression analysis.

```r
# Load necessary library
library(zoo)  # For rollmean()

# Load the sunspots dataset
data("sunspots")

# Create the dataframe with values from indices 11 to 2770 (to capture full cycle)
data <- data.frame(
  number_of_sunspots = sunspots[11:2770],  # Extracting the required range
  position_in_cycle = rep(0, 2760),
  percentage_of_peak = rep(0, 2760)  # Initializing percentage_of_peak with zeroes
)

# Smoothen the sunspots data using a moving average filter
window_size <- 11  # Choose an odd number for centered alignment
data$smoothed_sunspots <- rollmean(data$number_of_sunspots, k = window_size, fill = NA, align = "center

# Replace NA values at the start and end with original values
data$smoothed_sunspots[is.na(data$smoothed_sunspots)] <- data$number_of_sunspots[is.na(data$smoothed_su

# Identify indices where smoothed sunspots are below 20
below_20 <- which(data$smoothed_sunspots < 20)

# Find continuous segments where sunspots are below 20
segment_starts <- c(below_20[1], below_20[which(diff(below_20) > 1) + 1])
segment_ends <- c(below_20[which(diff(below_20) > 1)], below_20[length(below_20)])

# Find minima in each segment
filtered_minima <- c()
prev_min <- -Inf

for (i in seq_along(segment_starts)) {
  segment_range <- segment_starts[i]:segment_ends[i]
  segment_values <- data$smoothed_sunspots[segment_range]

  if (length(segment_values) > 0) {
    local_min_index <- segment_range[which.min(segment_values)]  # Find index of min value

    # Ensure minima are at least 100 indices apart
    if (local_min_index - prev_min >= 100) {
      filtered_minima <- c(filtered_minima, local_min_index)
      prev_min <- local_min_index
    }
  }
}

# Identify indices where sunspots are exactly 0
```

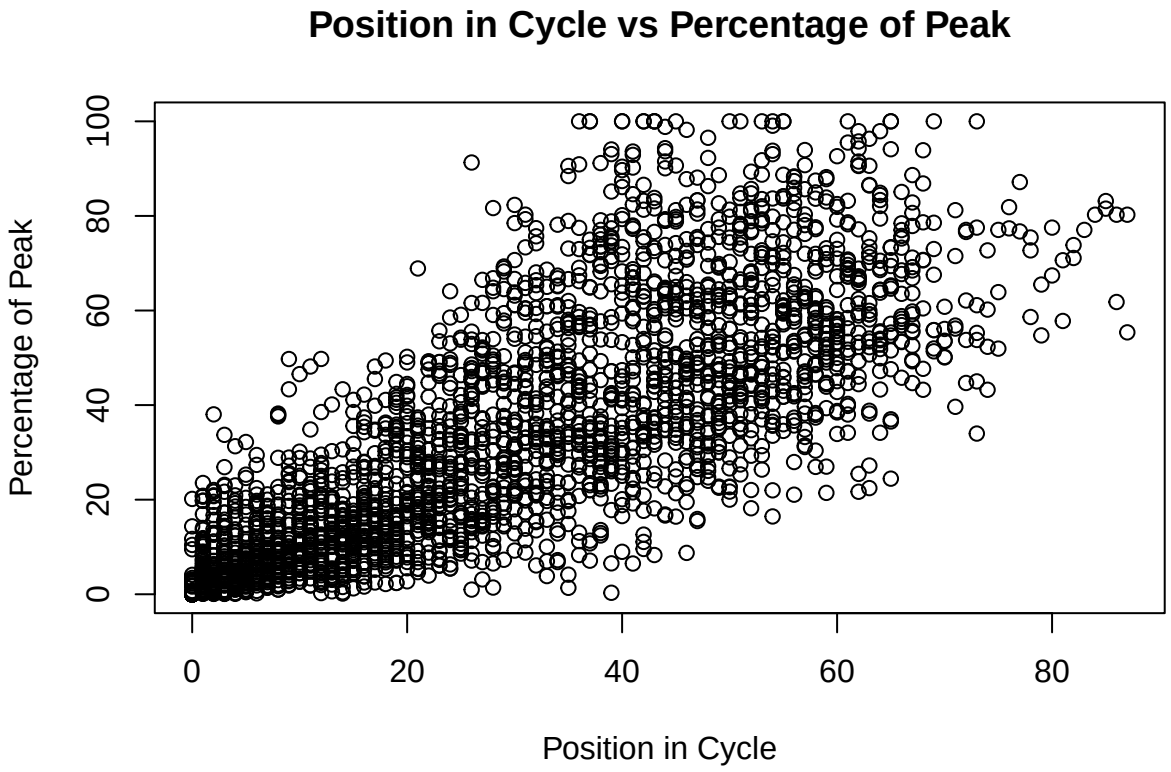
```r
zero_indices <- which(data$number_of_sunspots == 0)

# Compute percentage of peak sunspots in a ±66 index window
for (i in 1:nrow(data)) {
  start_idx <- max(1, i - 66)
  end_idx <- min(nrow(data), i + 66)
  max_sunspots <- max(data$number_of_sunspots[start_idx:end_idx], na.rm = TRUE)  # Avoid NA issues
  data$percentage_of_peak[i] <- ifelse(max_sunspots > 0, (data$number_of_sunspots[i] / max_sunspots) * 1
}

# Assign distance to the nearest minimum or zero for each index based on percentage_of_peak
data$position_in_cycle <- sapply(1:nrow(data), function(i) {
  if (data$percentage_of_peak[i] == 0) {
    return(0)  # If percentage_of_peak is 0, set position_in_cycle to 0
  } else if (data$percentage_of_peak[i] < 50) {
    # Find the closest zero and the closest minimum
    dist_to_minima <- ifelse(length(filtered_minima) > 0, min(abs(filtered_minima - i)), Inf)
    dist_to_zero <- ifelse(length(zero_indices) > 0, min(abs(zero_indices - i)), Inf)
    return(min(dist_to_minima, dist_to_zero))  # Take the smaller of the two
  } else {
    return(ifelse(length(filtered_minima) > 0, min(abs(filtered_minima - i)), Inf))  # Check for empty
  }
})

# Print the first few rows
head(data)
```

```
##   number_of_sunspots position_in_cycle percentage_of_peak smoothed_sunspots
## 1              158.6                65          100.00000         158.60000
## 2               85.2                64           53.72005          85.20000
## 3               73.3                48           46.21690          73.30000
## 4               75.9                47           47.85624          75.90000
## 5               89.2                61           56.24212          89.20000
## 6               88.3                60           55.67465          94.55455
```

```r
# Print detected local minima indices
print(filtered_minima)
```

```
##  [1]   66  200  307  415  550  725  882 1009 1123 1274 1410 1550 1684 1819 1964
## [16] 2088 2207 2320 2454 2579 2716
```

```r
# Plot results
plot(data$position_in_cycle, data$percentage_of_peak, main = "Position in Cycle vs Percentage of Peak",
```




```
cor(data$percentage_of_peak, data$position_in_cycle)
```

```
## [1] 0.7867963
```

Observations

The smoothed sunspot data helped in detecting local minima, marking new sunspot cycles.

Position in Cycle and Percentage of Peak were calculated based on local minima and sunspot peak values.

The scatter plot suggests a potential relationship between Position in Cycle and Percentage of Peak.

The correlation coefficient will give us an initial idea of their association.

Now that we have prepared the dataset, we proceed to fitting regression models to assess whether a linear relationship exists.

Least Squares Estimation (LSE)

Least Squares Estimation (LSE) is a method used to fit a linear model by minimizing the sum of squared residuals. Given a response variable Y and a predictor X, the linear model takes the form:

$$Y = \beta_0 + \beta_1 X + \epsilon$$

where β_0 (intercept) and β_1 (slope) are estimated by minimizing:

$$\sum (Y_i - \hat{Y}_i)^2$$

LSE assumes that residuals (errors) are normally distributed with constant variance. We will fit a linear regression model using LSE and examine diagnostic plots to check its validity.

Fitting LSE Model

```r
# Fit the LSE model
lse_model <- lm(percentage_of_peak ~ position_in_cycle, data = data)

# Summary of the model
summary(lse_model)
```

```
##
## Call:
## lm(formula = percentage_of_peak ~ position_in_cycle, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -43.795  -9.573  -3.038   8.284  61.508
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        4.17286    0.53333   7.824 7.22e-15 ***
## position_in_cycle  0.98520    0.01472  66.946  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.87 on 2758 degrees of freedom
## Multiple R-squared:  0.619,  Adjusted R-squared:  0.6189
## F-statistic:  4482 on 1 and 2758 DF,  p-value: < 2.2e-16
```

Observations from LSE Summary

- The estimated regression equation is:

$$\text{Percentage of Peak} = 4.17 + 0.985 \times \text{Position in Cycle}$$

- The **intercept** estimate is **4.17**, meaning when Position in Cycle is zero, the predicted Percentage of Peak is **4.17%**.

- The **slope** estimate is **0.985**, indicating that for every unit increase in Position in Cycle, the Percentage of Peak increases by approximately **0.985%**.
- The **p-value** for the slope coefficient is $<$ **2e-16**, meaning Position in Cycle is a highly significant predictor of Percentage of Peak.
- The **R-squared value** is **0.619**, meaning **61.9% of the variation** in Percentage of Peak is explained by Position in Cycle.
- The **residual standard error** is **15.87**, indicating the typical deviation of observed values from the regression line.
- However, we must check **residual diagnostics** to validate the model assumptions and assess whether a linear model is appropriate before making any conclusions.

Residual Diagnostics for LSE

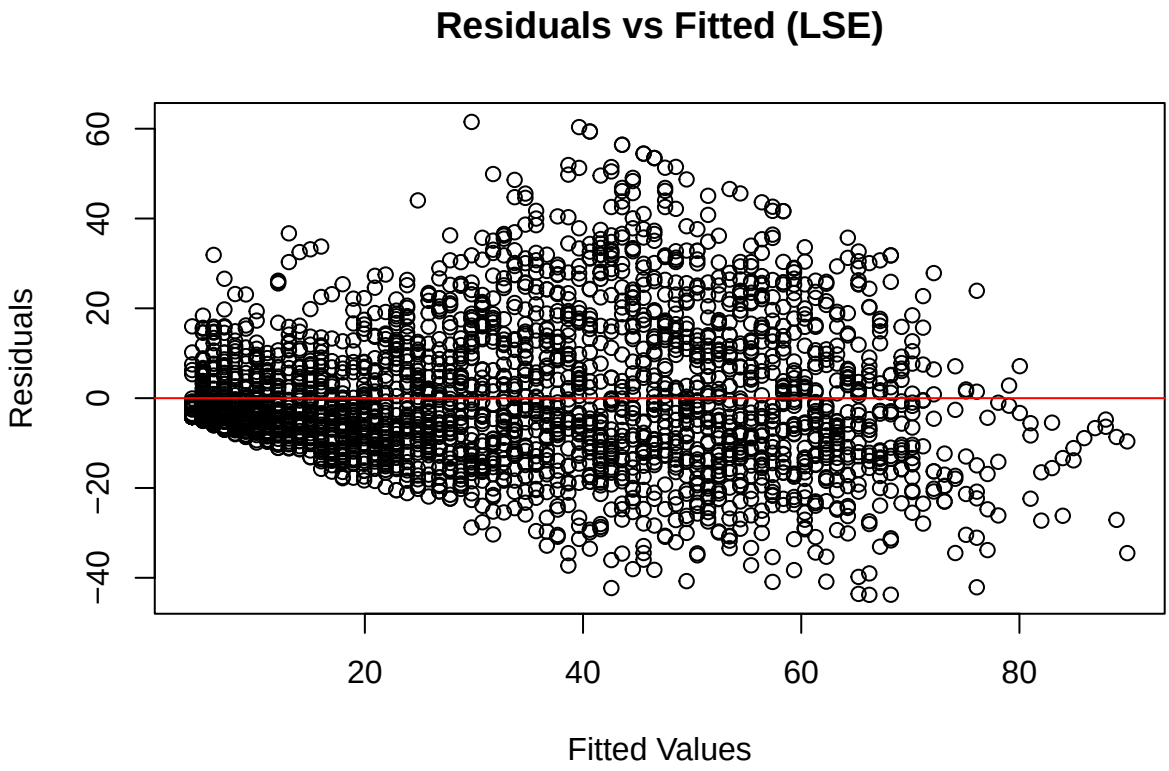
After fitting the Least Squares Estimator (LSE) model, it is essential to check if the assumptions of linear regression hold. These include:

1. **Linearity**: The relationship between `position_in_cycle` and `percentage_of_peak` should be linear.
2. **Homoscedasticity**: The residuals should have constant variance across fitted values.
3. **Normality**: The residuals should be approximately normally distributed.
4. **Independence**: The residuals should not exhibit autocorrelation.

To verify these assumptions, we perform the following checks:

- **Residual vs Fitted Plot**: Checks for homoscedasticity.

```
# Residual vs Fitted Plot
plot(fitted(lse_model), resid(lse_model),
     main = "Residuals vs Fitted (LSE)",
     xlab = "Fitted Values", ylab = "Residuals")
abline(h = 0, col = "red")
```

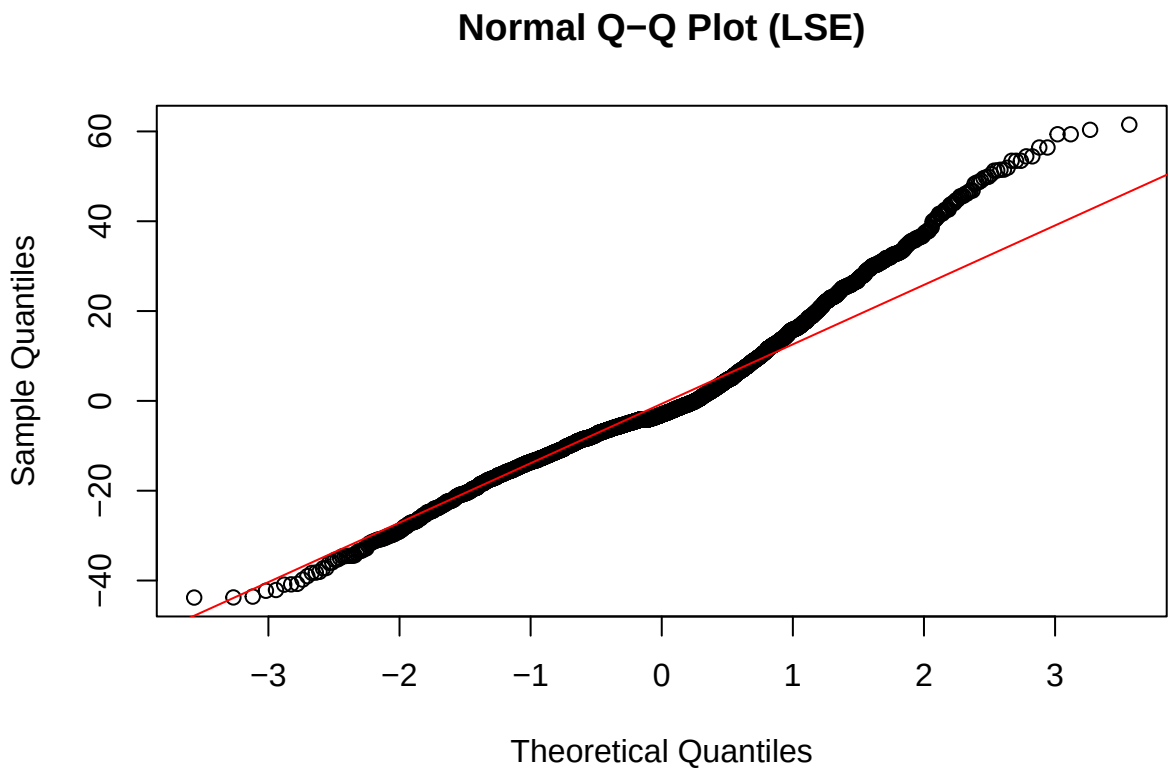


Observation

- For fitted values up to 20, residuals appear denser below the red line.
- As fitted values increase beyond 20, residuals spread more symmetrically on both sides, with increasing variance.
- This suggests potential heteroscedasticity, meaning the variability of residuals is not constant across fitted values.

- Such a pattern indicates that the model might not fully capture all aspects of the data, and transformations or alternative models might be needed for better fit.
- **Normal Q-Q Plot**: Checks if residuals follow a normal distribution.

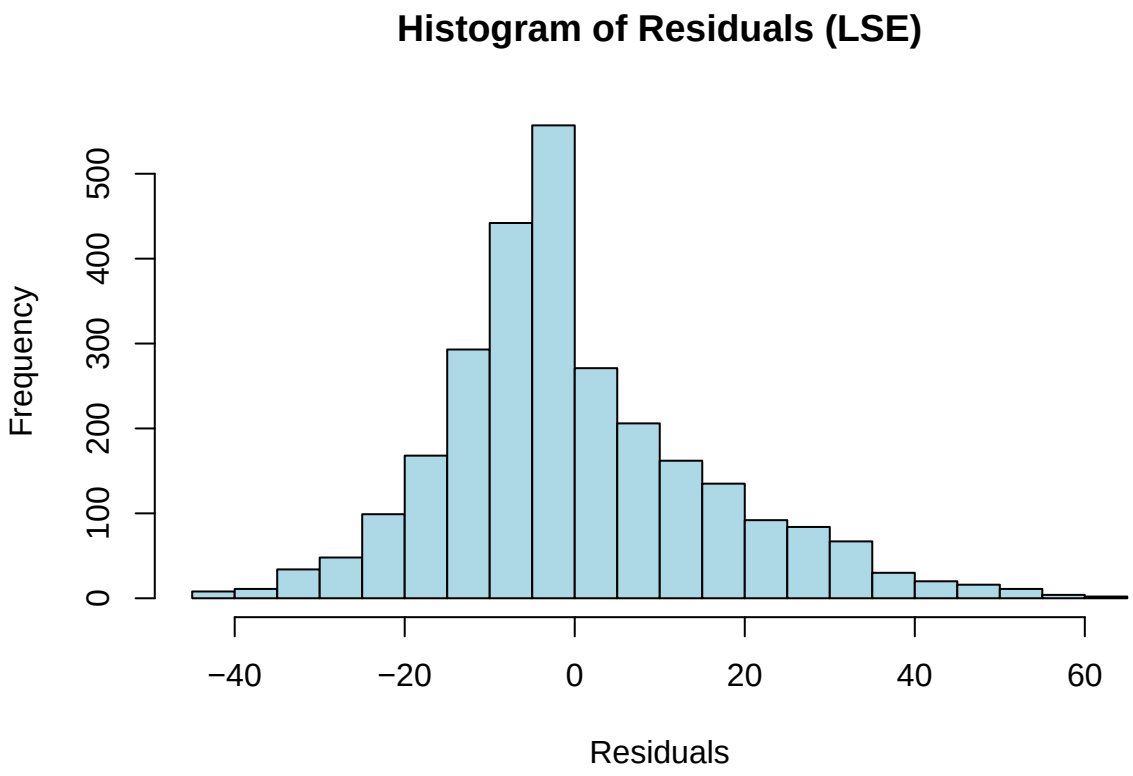
```
# Normal Q-Q Plot
qqnorm(resid(lse_model), main = "Normal Q-Q Plot (LSE)")
qqline(resid(lse_model), col = "red")
```



Observation

- The Q-Q plot shows deviations from the red reference line, particularly in the tails.
- This suggests that the residuals may not be perfectly normal, with some skewness or heavy tails.
- It is important to confirm this with a formal normality test.
- **Histogram of Residuals**: A visual check for normality.

```
# Histogram of Residuals
hist(resid(lse_model), breaks = 30, main = "Histogram of Residuals (LSE)",
     xlab = "Residuals", col = "lightblue", border = "black")
```



Observation

- The histogram of residuals appears slightly right-skewed rather than perfectly symmetric.
- This indicates deviations from a normal distribution, aligning with the Q-Q plot findings.
- The presence of skewness suggests that normality assumptions for the residuals may not hold completely.
- **Shapiro-Wilk Test**: A formal statistical test for normality.

```
# Shapiro-Wilk Test for Normality
shapiro_test_lse <- shapiro.test(resid(lse_model))
shapiro_test_lse
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(lse_model)
## W = 0.96556, p-value < 2.2e-16
```

Observation

- The Shapiro-Wilk test for normality resulted in $W = 0.96556$ with a p-value $< 2.2 \times 10^{-16}$.

- Since the p-value is extremely small, we reject the null hypothesis that the residuals follow a normal distribution.

- This confirms the deviations observed in the Q-Q plot and histogram, suggesting that the residuals are not normally distributed.

Least Absolute Deviations (LAD) Regression

Least Absolute Deviations (LAD) regression, also known as Least Absolute Errors (LAE) regression or median regression, minimizes the sum of absolute residuals instead of squared residuals. This makes LAD more robust to outliers compared to the Least Squares Estimation (LSE) method.

Objective Function

Given a dataset with response variable Y and predictor X, the LAD regression model estimates the coefficients by minimizing the sum of absolute residuals:

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{n} |Y_i - X_i^\top \beta|$$

where: - Y_i is the observed response, - X_i is the vector of predictor variables, - β is the coefficient vector.

Key Differences from LSE:

- **Objective Function:** LSE minimizes squared errors ($\sum (Y_i - \hat{Y}_i)^2$), while LAD minimizes absolute errors ($\sum |Y_i - \hat{Y}_i|$).
- **Sensitivity to Outliers:** LAD gives equal weight to all residuals, reducing the influence of extreme values.
- **Solution Method:** Unlike LSE, which has a closed-form solution, LAD regression requires linear programming techniques.
- **Interpretation:** LAD estimates the median trend of the response variable, whereas LSE estimates the mean trend.

We now proceed with fitting the LAD model and analyzing its performance.

```r
# Load necessary library for LAD regression
library(quantreg)
```

```
## Loading required package: SparseM
```

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

```r
# Fit the LAD model (quantile regression at tau = 0.5, median)
lad_model <- rq(percentage_of_peak ~ position_in_cycle, data = data, tau = 0.5)

# Display model summary
summary(lad_model)
```

```
##
## Call: rq(formula = percentage_of_peak ~ position_in_cycle, tau = 0.5,
##     data = data)
##
## tau: [1] 0.5
##
## Coefficients:
##                   Value    Std. Error t value  Pr(>|t|)
## (Intercept)        0.96385  0.34810    2.76887  0.00566
## position_in_cycle  0.99072  0.01649   60.09572  0.00000
```

Observation

- The estimated intercept is **0.96**, and the coefficient for **position_in_cycle** is **0.99**.
- The coefficient for **position_in_cycle** is statistically significant (**p < 0.001**), indicating a strong association.
- The standard errors for both coefficients are provided, with **0.35** for the intercept and **0.016** for position_in_cycle.
- The **high t-value (60.10)** for position_in_cycle suggests a strong relationship with percentage_of_peak.
- Compared to the LSE model, the estimated slope (0.99) is quite similar to **0.985** in LSE, indicating a consistent linear pattern.
- The standard error for LAD estimates differs slightly from LSE, which may suggest a difference in variability handling.

Residual Analysis for LAD Regression

Residual analysis helps assess the goodness of fit of a regression model. In Least Absolute Deviations (LAD) regression, residuals are defined as:

$$r_i = y_i - \hat{y}_i$$

where y_i are the observed values and $\hat{y}_i$ are the fitted values.

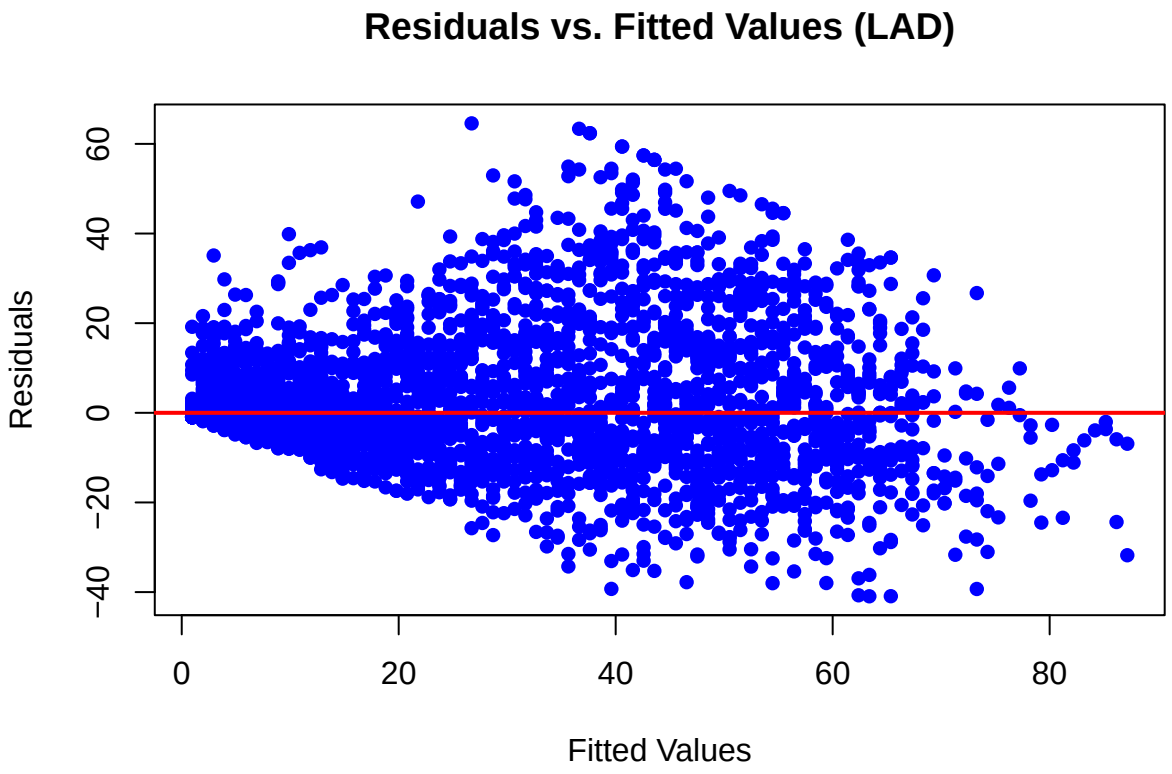
For LAD regression, the residuals are minimized using absolute deviations rather than squared deviations, making the model more robust to outliers.

The **residuals vs. fitted values plot** helps assess: - Whether the residuals are randomly scattered around zero, indicating a good fit. - Any systematic patterns, which might suggest a model misspecification. - The presence of heteroscedasticity (non-constant variance), which can affect inference.

We now plot residuals against fitted values for the LAD model.

```r
# Compute residuals for LAD model
lad_residuals <- resid(lad_model)

# Compute fitted values
lad_fitted <- fitted(lad_model)
```

```r
# Plot residuals vs. fitted values
plot(lad_fitted, lad_residuals,
     main = "Residuals vs. Fitted Values (LAD)",
     xlab = "Fitted Values",
     ylab = "Residuals",
     pch = 16, col = "blue")

# Add horizontal reference line at zero
abline(h = 0, col = "red", lwd = 2)
```

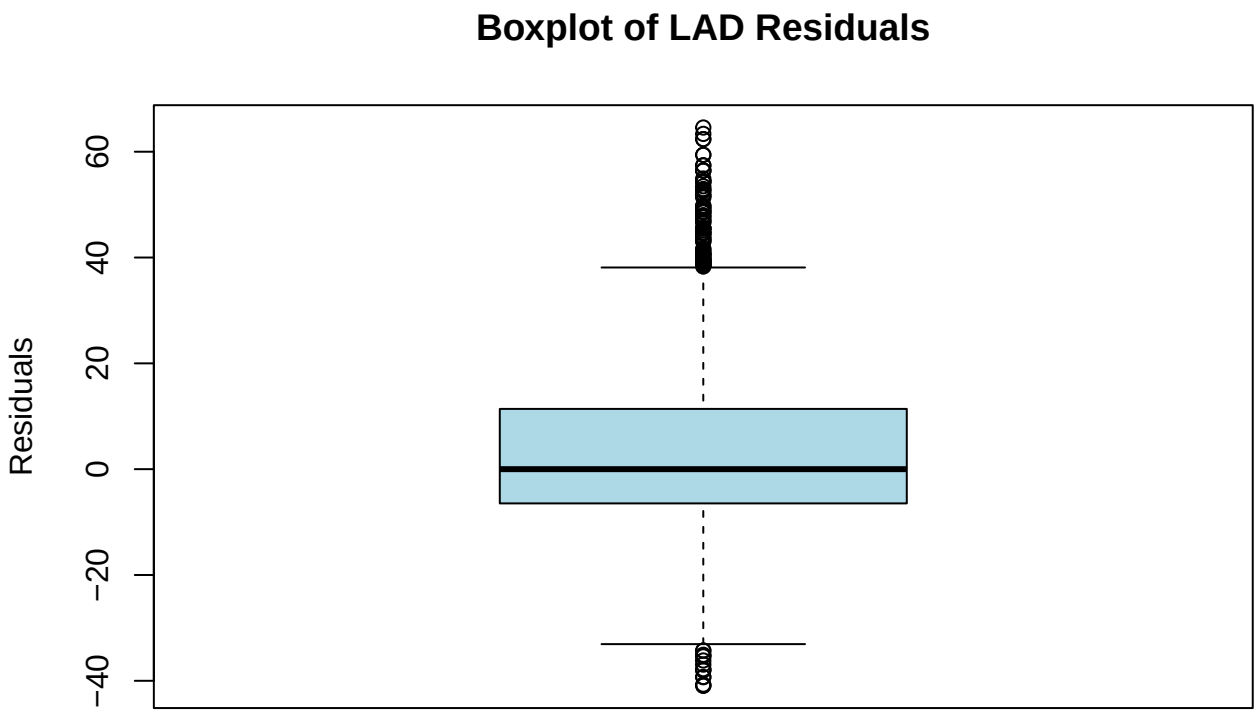


Observation

- Residuals are more tightly clustered around zero for lower fitted values, suggesting better model fit in that range.
- As fitted values increase, the spread of residuals widens, indicating increasing variability in prediction errors.
- At higher fitted values, the residuals become more dispersed and show a tendency toward negative values, which may suggest underestimation in that range.
- The overall pattern suggests that the model's accuracy varies across the range of fitted values, with more stable predictions at lower fitted values and higher uncertainty at larger fitted values.

Residual Symmetry: Boxplot and Histogram

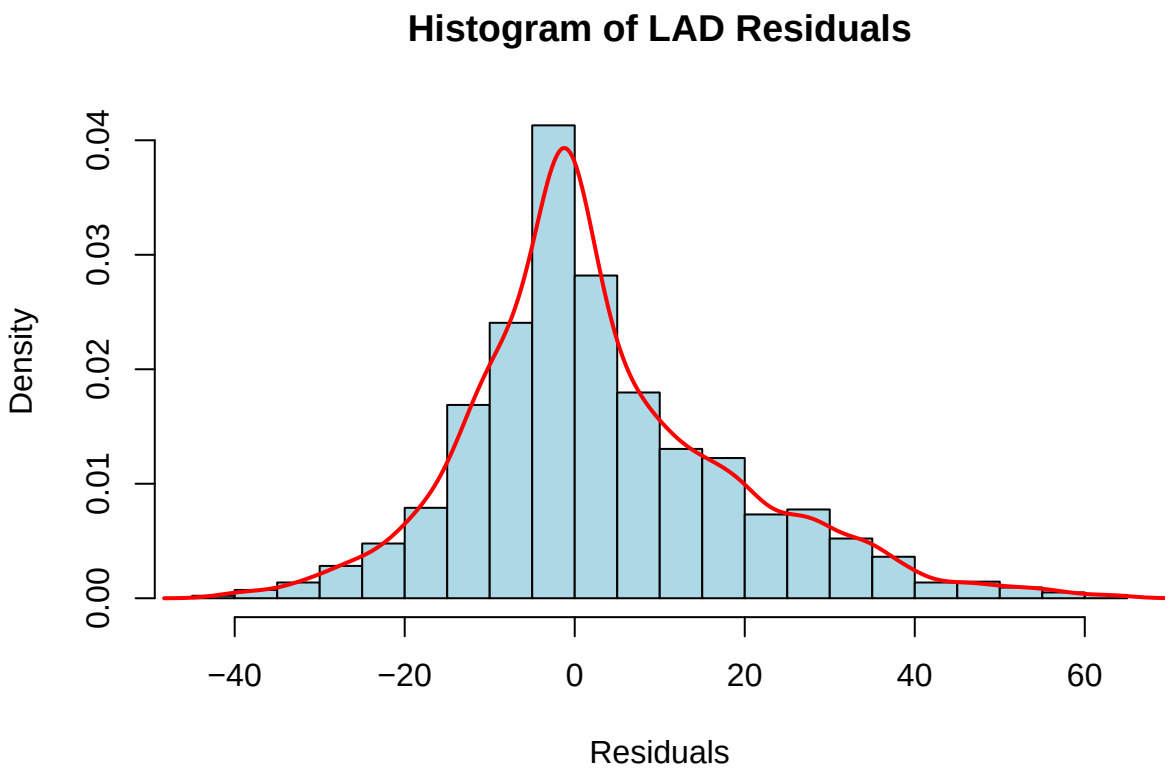
```r
# Boxplot of residuals
boxplot(resid(lad_model), main = "Boxplot of LAD Residuals", ylab = "Residuals", col = "lightblue")
```



Observation: Boxplot

The boxplot of LAD residuals reveals the presence of numerous outliers.
- This suggests that the residual distribution contains extreme values, which is expected in LAD regression as it is less sensitive to outliers than LSE.
- The spread of residuals does not appear perfectly symmetric, indicating potential deviations from an ideal distribution.

```r
# Histogram of residuals
hist(resid(lad_model), breaks = 30, main = "Histogram of LAD Residuals",
     xlab = "Residuals", col = "lightblue", border = "black", probability = TRUE)

# Add density curve
lines(density(resid(lad_model)), col = "red", lwd = 2)
```



Observation: Histogram

The histogram of LAD residuals shows noticeable right skewness.
- There is a higher concentration of residuals on the left side, with a longer tail extending toward positive values.
- This suggests that LAD residuals are not symmetrically distributed, reinforcing the observation from the boxplot.
- The skewness indicates that the LAD model may be capturing some non-normal error structure in the data.

Pseudo R-Squared for LAD Regression

Unlike the traditional coefficient of determination (R^2) used in Least Squares Estimation (LSE), Least Absolute Deviations (LAD) regression does not minimize squared errors but rather the absolute deviations. Hence, the usual R^2 is not applicable.

To evaluate the goodness-of-fit for LAD, we use **pseudo** R^2, which is defined as:

$$R^2_{\text{pseudo}} = 1 - \frac{\sum |y_i - \hat{y}_i|}{\sum |y_i - \tilde{y}|}$$

where: - y_i are the observed values, - $\hat{y}_i$ are the fitted values from the LAD model, - $\tilde{y}$ is the median of the observed values.

This metric assesses how much improvement the LAD model provides over using just the median as a predictor.

Next, we compute pseudo R^2 for our LAD model.

```r
## Compute Pseudo R-Squared for LAD Regression
pseudo_r2 <- 1 - sum(abs(data$percentage_of_peak - fitted(lad_model))) /
                 sum(abs(data$percentage_of_peak - median(data$percentage_of_peak)))

## Print the result
pseudo_r2
```

```
## [1] 0.4572562
```

Observation

The pseudo R^2 for the LAD model is **0.457**, indicating that the model explains approximately **45.7%** of the variability in the response variable based on absolute deviations.

Compared to the **multiple R^2 of 0.619** from the LSE model, the LAD model captures less overall variation. However, since LAD is robust to outliers, this lower value does not necessarily mean a worse fit—rather, it suggests that LSE might be more influenced by extreme values.

Further comparison of residual behavior and predictive performance will provide a clearer picture of which model is more suitable.

Mean Absolute Error (MAE) & Root Mean Squared Error (RMSE) Comparison

Both LSE and LAD are optimized for different error measures:

- **LSE minimizes squared errors**, so it is expected to have a lower RMSE.
- **LAD minimizes absolute errors**, so it is expected to have a lower MAE.

To better understand the nature of the errors, we compute:

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

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

Interpretation

- **LAD had a lower MAE (11.737) than LSE (12.056), as expected.** This suggests that LAD provides more stable predictions for most observations.
- **LSE had a lower RMSE (15.861) than LAD (16.151), also expected.** This suggests that LSE fits the majority of points better but might be affected by large deviations.
- The gap between MAE and RMSE for LSE suggests that **some large residuals are inflating the RMSE**, making LSE more sensitive to extreme values.

Thus, while this comparison does not determine which model is better, it highlights that **LAD provides a more consistent fit across data points, whereas LSE gives more weight to large deviations**.

Compute and Compare MAE & RMSE

```r
# Compute residuals
lse_resid <- resid(lse_model)
lad_resid <- resid(lad_model)

# Compute MAE
mae_lse <- mean(abs(lse_resid))
mae_lad <- mean(abs(lad_resid))

# Compute RMSE
rmse_lse <- sqrt(mean(lse_resid^2))
rmse_lad <- sqrt(mean(lad_resid^2))

# Display results
mae_lse; mae_lad
```

```
## [1] 12.05616
```

```
## [1] 11.73735
```

```r
rmse_lse; rmse_lad
```

```
## [1] 15.8612
```

```
## [1] 16.15106
```

Observation

Both **LSE** and **LAD** optimize different loss functions, so comparing **MAE** and **RMSE** primarily helps us understand the **nature of their residuals**, rather than choosing a "better" model outright.

- **MAE (Mean Absolute Error):**
  - LSE: **12.056**
  - LAD: **11.737**
  - As expected, **LAD shows a lower MAE**, reflecting its design to minimize absolute deviations.
- **RMSE (Root Mean Squared Error):**
  - LSE: **15.861**
  - LAD: **16.151**
  - Similarly, **LSE achieves a lower RMSE**, which aligns with its goal of minimizing squared deviations.

Interpretability & Robustness Discussion

Both **Least Squares Estimation (LSE)** and **Least Absolute Deviations (LAD)** offer useful perspectives for modeling the relationship between **Position in Cycle** and **Percentage of Peak**. Their appropriateness depends on the goals of the analysis and the nature of the data.

Sensitivity to Outliers

- **LSE** minimizes squared residuals, making it **sensitive to large errors and outliers**.
- **LAD** minimizes absolute deviations, offering **greater robustness to outliers**.
- Residual diagnostics suggested the presence of influential outliers in LSE, while LAD handled them more gracefully.

Model Fit & Error Metrics

- **LSE** had a higher traditional R^2 value (**0.619**) compared to LAD's **pseudo** R^2 (**0.457**), indicating that LSE explains more variability in terms of squared error.
- However, LAD showed a **lower MAE (11.737)** compared to **LSE (12.056)**, consistent with its loss function.
- **LSE had a lower RMSE (15.861)** than **LAD (16.151)**, which aligns with its goal of minimizing squared deviations.
- Since each model optimizes a different criterion, these metrics are not meant to directly compete but rather to highlight **differences in residual characteristics**.

Residual Behavior

- **LSE residuals** exhibited signs of **heteroscedasticity**, with a fan-shaped spread in the residual vs. fitted plot.
- **LAD residuals** were more uniformly spread, supporting its robustness against variance instability.
- **Normality assessments** (Shapiro-Wilk test and visualizations) revealed **departures from normality** in both models, especially in LAD with **right-skewed residuals and several outliers**.

Conclusion

- If the analysis goal is **to explain variability and make predictions under normality assumptions**, **LSE** may be appropriate.
- If the focus is on **robustness and mitigating the influence of extreme values**, **LAD** offers a more stable alternative.
- Given the presence of **non-normal residuals, outliers, and heteroscedasticity**, **LAD may provide a more reliable fit** under these conditions.

Future improvements could include **transformations, outlier analysis, or exploring nonlinear models** for better capturing underlying patterns.