Lab5

Cui Qingxuan, Nisal Amashan

2025-03-16

Contents

1 Collaborations

Nisal Amashan: Responsible for the question 1.

Cui Qingxuan: Responsible for the question 2.

2 Question 1

2.1 Part (a): Fitting a Cubic Regression Model

- The cubic regression model was fitted to the data, relating the **yield of cress (mg)** to the **concentration of fertilizer (%)**.
- The estimated coefficients are:

  - **Intercept** (β_0): 199.2835726
  - **Linear term** (β_1): 62.6342313
  - **Quadratic term** (β_2): -298.7662186
  - **Cubic term** (β_3): 158.6639403

The cubic regression model was fitted to the data, relating the yield of cress (mg) to the concentration of fertilizer (%).

From the results, all three polynomial terms are statistically significant at the 5% level, indicating that a cubic model provides a good fit to the data.

2.2 Part (b): Model Improvement, Coefficient Estimation, and Visualization

- The cubic regression model was fitted to the data to analyze the relationship between **cress yield (mg)** and **fertilizer concentration (%)**.
- The estimated coefficients along with their 95% confidence intervals are:
  - **Intercept** (β_0): 199.284 (95% CI: [188.446, 210.121])
  - **Linear term** (β_1): 62.634 (95% CI: [-81.065, 206.333])
  - **Quadratic term** (β_2): -298.766 (95% CI: [-629.220, 31.687])
  - **Cubic term** (β_3): 158.664 (95% CI: [-23.708, 341.036])
- The **linear term is not statistically significant** ($p = 0.388$), while the quadratic and cubic terms show marginal significance ($p = 0.0757$ and $p = 0.0872$, respectively).
- The **adjusted R-squared value is 0.6322**, indicating that the cubic model explains approximately **63.2% of the variance** in the data.
- The **F-statistic is 46.84** with a **p-value < 2.2e-16**, suggesting that the model is significantly better than a model with no predictors.
- The regression plot (below) shows a **nonlinear relationship**, with yield peaking at a moderate fertilizer concentration and decreasing at higher concentrations.

2.3 Part (c): Bootstrap Confidence Interval for Model Parameter

- A **manual bootstrap procedure** was used to estimate the **95% confidence interval** for the slope coefficient (β_1).
- **Bootstrap Setup:**
  - **10,000 bootstrap replicates** were generated by resampling the dataset with replacement.
  - The cubic regression model was refitted to each resampled dataset.
  - The distribution of **bootstrapped estimates** for β_1 was analyzed.
- **Bootstrap Results:**
  - The **95% percentile-based confidence interval** for β_1 is **[-66.85, 193.51]**.
  - The **distribution of bootstrapped slope estimates** is approximately normal.
- The bootstrap distribution plot (below) visually represents the **confidence interval** (green dashed lines) and the **original estimate** (blue dashed line).

2.4 Part (d): Bootstrap Confidence Intervals Using `boot` Package

- A **bootstrap resampling procedure** was performed using the `boot` package to estimate confidence intervals for β_1.

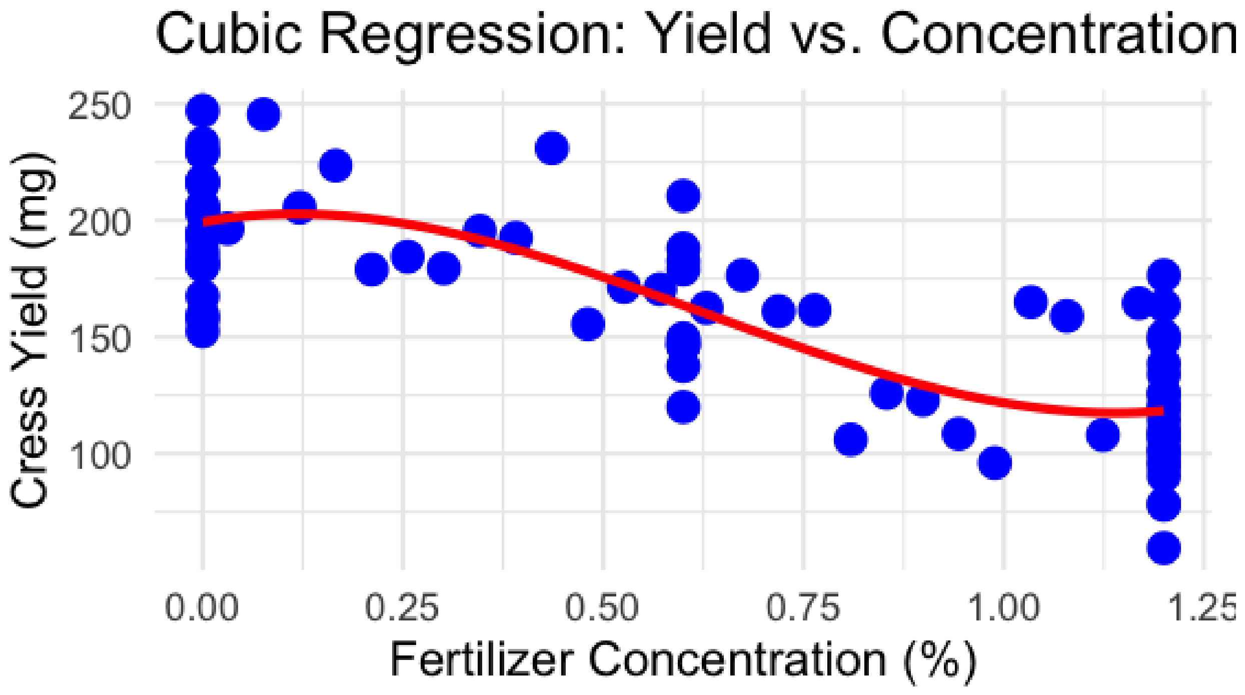


Figure 1: Cubic Regression Plot

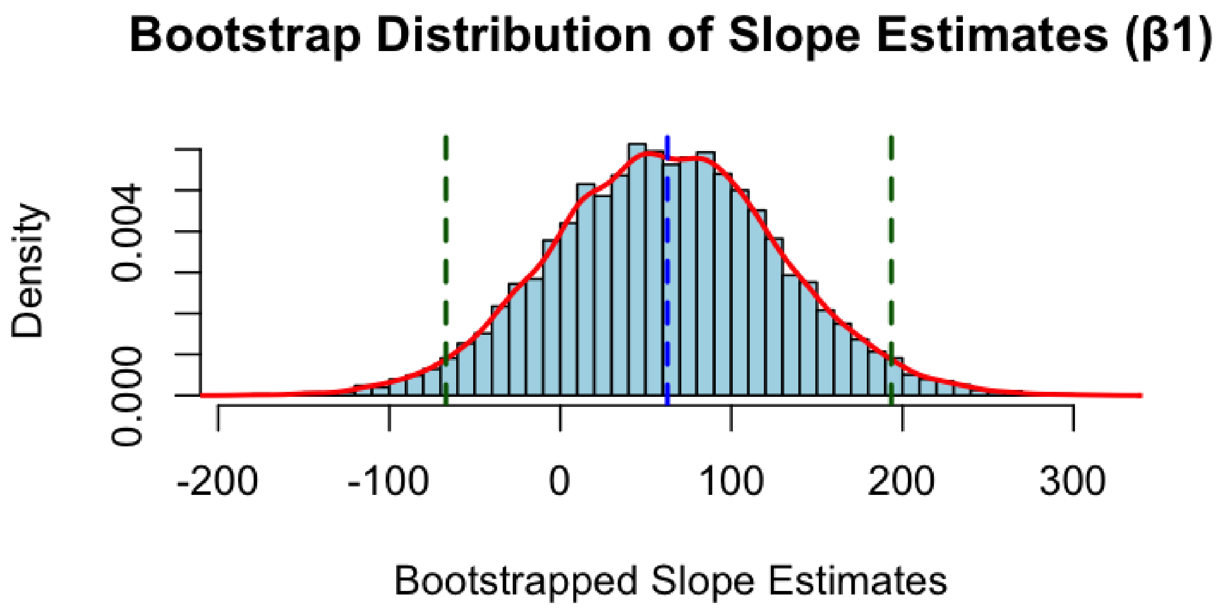


Figure 2: Bootstrap Distribution Plot

- **Bootstrap Setup:**
  - **10,000 bootstrap replicates** were used to estimate the distribution of β_1.
  - Two different methods were applied:
    1. **Percentile Method:** This method uses the 2.5th and 97.5th percentiles from the bootstrapped distribution.
    2. **Bias-Corrected and Accelerated (BCa) Method:** This adjusts for bias and skewness in the bootstrap distribution.
- **Bootstrap Results:**
  - The **95% percentile confidence interval** for β_1 is [**-71.33, 192.71**].
  - The **95% BCa confidence interval** for β_1 is [**-69.48, 193.76**].
- These results confirm that the **bootstrap confidence intervals** are consistent with the manually computed values.

2.5 Part (e): Comparison of Confidence Intervals and Conclusion

- **Comparison of Results:**
  - The **lm-based confidence interval (Part b)** is [**-81.07, 206.33**].
  - The **manual bootstrap confidence interval (Part c)** is [**-66.85, 193.51**].
  - The **percentile bootstrap confidence interval (Part d)** is [**-71.33, 192.71**].
  - The **BCa bootstrap confidence interval (Part d)** is [**-69.48, 193.76**].
- **Observations:**
  - All intervals are **reasonably close**, but the **lm-based interval is slightly wider**.
  - The bootstrap confidence intervals are **more centered** and have slightly smaller ranges compared to the lm-based method.
  - The percentile and BCa intervals are similar, suggesting that the bias-correction does not significantly affect the estimates.
- **Conclusion:**
  - The consistency between the intervals suggests that **bootstrapping is a reliable alternative** to the standard **lm-based confidence intervals**.
  - The variability in the confidence intervals highlights the **uncertainty** in estimating β_1, reinforcing the importance of using **bootstrap methods** to validate model results.

3 Question 2

3.1 Inverse Transformation Method Generating Random Values

X obeys Gumbel distribution whose cumulative distribution function is:

$$F(x) = \exp(-\exp(-(x-\mu-c))), \quad c = \log(\log(2))$$

.

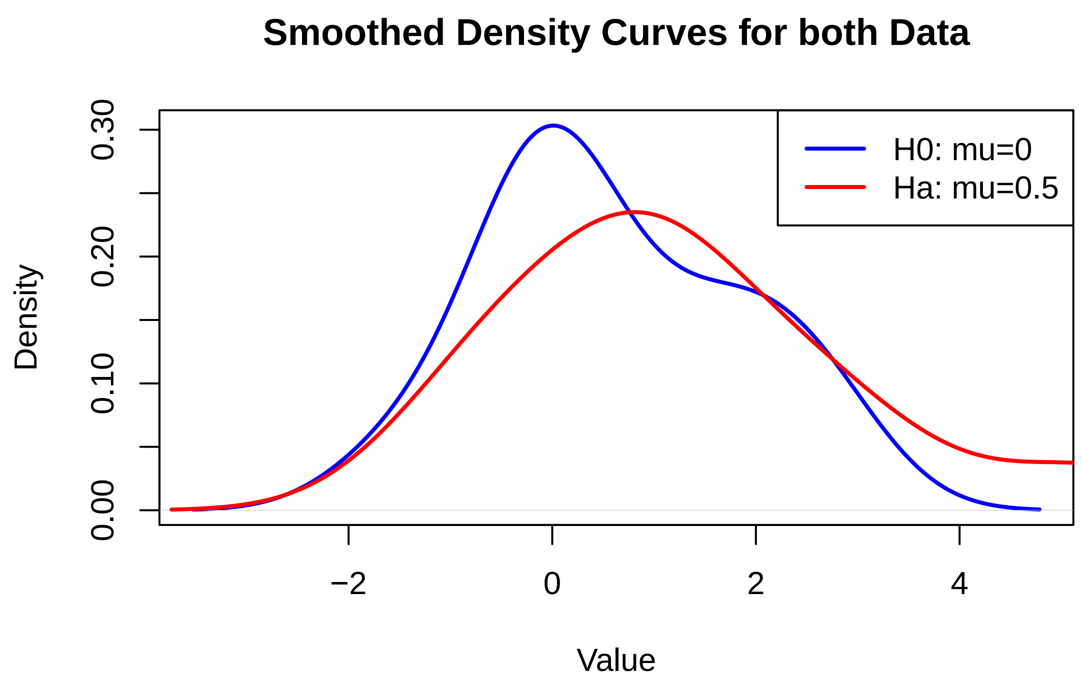
Then for a uniform distribution $U \sim U(0,1)$, $X = F^{-1}(U)$ will obey that distribution.

Based on that, the solution of equation $U = exp(-exp(-(x-\mu-c)))$ is:

$$X = -\ln(-\ln(U)) + \mu + c$$

where, $U \sim U(0,1)$, $c = \log(\log(2))$. So we can generate random values in term of this.

We set $\mu = [0,\ 0.5]$, and we get random values like the plot below.

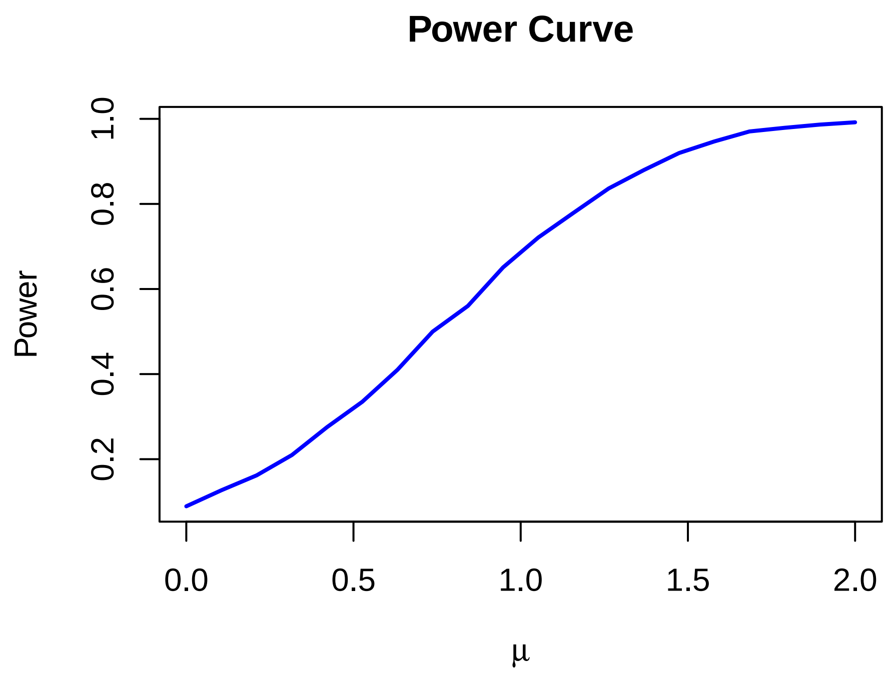


3.2 Sign Test

We selected 20 values for μ and set the number of repetitions to 10,000. To determine an appropriate number of repetitions, we initially tested values of 10, 100, 1,000, and 10,000. However, lower repetition counts resulted in noticeable fluctuations in the power values, which should ideally exhibit a smooth and stable increase as the plot below. Since our null hypothesis states that $\mu = 0$, the power is expected to be lower when μ is closer to 0. To ensure the reliability of our estimation, we considered the simulation's precision by examining the standard error (SE) of power estimates:

$$SE = \sqrt{\frac{P(1-P)}{\text{reps}}}$$

where P represents the estimated power. A larger number of repetitions reduces SE, leading to more precise estimates. With 10,000 repetitions, the standard error became sufficiently small, ensuring that the estimated power values were stable across repeated simulations. Therefore, we chose 10,000 repetitions to ensure the reliability and consistency of our results.



4 Appendix

4.1 Question 1

```
# Load necessary libraries
library(boot)
library(ggplot2)

# ---- (a) Read in the dataset and fit a cubic regression model ---- #


kresse_data <- read.table("data/kresseertrag.dat", header = FALSE)
colnames(kresse_data) <- c("ID", "fertilizer_conc", "cress_yield")  # Rename columns
data <- kresse_data[, -1]  # Remove the "ID" column

# Fit a cubic regression model
cubic_model <- lm(cress_yield ~ fertilizer_conc + I(fertilizer_conc^2) + I(fertilizer_conc^3), data = da

# Display model summary
summary(cubic_model)

# ---- (b) Model Improvement, Confidence Intervals, and Visualization ---- #

# Compute 95% confidence intervals
conf_intervals <- confint(cubic_model)
print(conf_intervals)
```

```r
# Generate smooth regression curve
conc_seq <- seq(min(data$fertilizer_conc), max(data$fertilizer_conc), length.out = 100)
predicted_yield <- predict(cubic_model, newdata = data.frame(fertilizer_conc = conc_seq))
plot_data <- data.frame(fertilizer_conc = conc_seq, predicted_yield = predicted_yield)

# Plot regression curve
ggplot(data, aes(x = fertilizer_conc, y = cress_yield)) +
  geom_point(color = "blue", size = 3) +
  geom_line(data = plot_data, aes(x = fertilizer_conc, y = predicted_yield), color = "red", size = 1) +
  labs(title = "Cubic Regression: Yield vs. Fertilizer Concentration",
       x = "Fertilizer Concentration (%)", y = "Cress Yield (mg)") +
  theme_minimal()

# ---- (c) Manual Bootstrap for 95% Confidence Interval ---- #

# Function for bootstrapping
bootstrap_coeffs <- function(data, indices) {
  sample_data <- data[indices, ]
  model <- lm(cress_yield ~ fertilizer_conc + I(fertilizer_conc^2) + I(fertilizer_conc^3), data = sampl
  return(coef(model))
}

# Perform bootstrap manually
set.seed(123)
n_replicates <- 10000
boot_estimates <- matrix(NA, nrow = n_replicates, ncol = 4)

for (i in 1:n_replicates) {
  boot_indices <- sample(1:nrow(data), nrow(data), replace = TRUE)
  boot_estimates[i, ] <- bootstrap_coeffs(data, boot_indices)
}


beta1_ci <- quantile(boot_estimates[, 2], probs = c(0.025, 0.975))
print(beta1_ci)

# Plot bootstrap distribution
hist(boot_estimates[, 2], breaks = 50, col = "lightblue", border = "black",
     main = "Bootstrap Distribution of Slope Estimates",
     xlab = "Bootstrapped Slope Estimates",
     freq = FALSE)
lines(density(boot_estimates[, 2]), col = "red", lwd = 2)
abline(v = coef(cubic_model)[2], col = "blue", lwd = 2, lty = 2)
abline(v = beta1_ci, col = "darkgreen", lwd = 2, lty = 2)

# ---- (d) Bootstrap Confidence Intervals Using `boot` Package ---- #

# Define function for boot package
boot_coeffs <- function(data, indices) {
  sample_data <- data[indices, ]
  model <- lm(cress_yield ~ fertilizer_conc + I(fertilizer_conc^2) + I(fertilizer_conc^3), data = sampl
  return(coef(model))
}
```

```r
# Run bootstrap with 10,000 resamples
set.seed(123)
boot_results <- boot(data, boot_coeffs, R = 10000)

# Compute bootstrap confidence intervals
boot_ci_perc <- boot.ci(boot_results, type = "perc", index = 2)
boot_ci_bca <- boot.ci(boot_results, type = "bca", index = 2)

# Print bootstrap confidence intervals
print(boot_ci_perc$percent[4:5])
print(boot_ci_bca$bca[4:5])

# ---- (e) Overlay Bootstrap Regression Curve ---- #

# Compute mean bootstrap coefficients
boot_coeff_mean <- colMeans(boot_results$t)

# Compute predicted values using bootstrapped coefficients
predicted_yield_boot <- boot_coeff_mean[1] + boot_coeff_mean[2] * conc_seq +
  boot_coeff_mean[3] * conc_seq^2 + boot_coeff_mean[4] * conc_seq^3

# Create a data frame for the bootstrap regression line
plot_data_boot <- data.frame(conc_seq, predicted_yield_boot)

# Overlay bootstrap regression curve
ggplot(data, aes(x = fertilizer_conc, y = cress_yield)) +
  geom_point(color = "blue", size = 3) +
  geom_line(data = plot_data, aes(x = conc_seq, y = predicted_yield), color = "black", size = 1, linetyp
  geom_line(data = plot_data_boot, aes(x = conc_seq, y = predicted_yield_boot), color = "red", size = 1
  labs(title = "Regression with Bootstrapped Prediction",
       x = "Fertilizer Concentration (%)", y = "Yield (mg)") +
  theme_minimal()
```

4.2 Question 2

```r
generate_gumbel = function(n, mu, c = log(log(2))) {
  U <- runif(n)
  X <- mu + c - log(-log(U))
  return(X)
}

sign_test = function(mu){
  x = generate_gumbel(n=13, mu)
  s = sum(x>0)
  p_value = binom.test(s, n, p = 0.5, alternative = "greater")$p.value
  return(p_value)
}
# power = P(reject H0| Ha is true)
compute_power = function(mu, n = 13, reps=10000, alpha = 0.05) {
  reject_count = sum(replicate(reps, sign_test(generate_gumbel(n, mu))) < alpha)
  # repeat the simulation for reps times
  return(reject_count / reps)
}

# H0: mu=0
set.seed(123)
n = 13
data_H0 = round(generate_gumbel(n, mu = 0), 3)
# Ha: mu>0
data_Ha = round(generate_gumbel(n, mu = 0.5), 3)
# a appropriate grid values for mu

# calculate density
# work for presentation
denH0 = density(data_H0)
denHa = density(data_Ha)

plot(denH0, col = "blue", lwd = 2, main = "Smoothed Density Curves for both Data", xlab = "Value", ylab
lines(denHa, col = "red", lwd = 2)
legend("topright", legend = c("H0: mu=0", "Ha: mu=0.5"), col = c("blue", "red"), lwd = 2)

# sign test
# set grid values
mu = seq(from=0, to=2, length.out=20)
power_values = sapply(mu, compute_power)
# reps has been defined in the definition of compute power function as 1000
plot(mu, power_values, type = "l", col = "blue", lwd = 2,
     xlab = expression(mu), ylab = "Power", main = "Power Curve")
```