Assignments_4 Course

Mathematical and Computational Statistics

Achladianakis Minas

A report presented for the Course Mathematical and Computational Statistics

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
UNIVERSITY OF CRETE

Department of Applied Mathematics

University of Crete

Greece, November 10, 2023

Abstract

This assignment is part of my evaluation for the course: Mathematical and Computational Statistics.

List of Figures

Assignment 2

2.1 Part 1: Residual Diagnostics on Model M3

Performing residual diagnostics on the full second-order model M3, including the evaluation of serial correlation in the residuals and the alignment with the hypotheses of the Gauss-Markov Theorem. Selected figures will support discussion of findings. We start by creating our M3 model as in assignment 1:

```
data("airquality")

# Function to replace missing values with the mean of the column
replace_na_with_mean <- function(x) {
  if (is.numeric(x)) {
    x[is.na(x)] <- mean(x, na.rm = TRUE)
  }
  return(x)
}

# Function to clean each column of the airquality dataset
airquality_clean <- apply(airquality, 2, replace_na_with_mean)

# Convert the result back to a data frame
airquality_clean <- as.data.frame(airquality_clean)


# The full second-order model M3
M3 <- lm(Ozone ~ Solar.R + Wind + Temp + I(Solar.R^2) + I(Wind^2) +
↪  I(Temp^2) +
           Solar.R:Wind + Solar.R:Temp + Wind:Temp, data =
           ↪  airquality_clean)

summary(M3)
```

with summary:

```
    Call:
lm(formula = Ozone ~ Solar.R + Wind + Temp + I(Solar.R^2) + I(Wind^2) +
    I(Temp^2) + Solar.R:Wind + Solar.R:Temp + Wind:Temp, data =
    ↪  airquality_clean)

Residuals:
    Min      1Q  Median      3Q     Max
```

```
-37.242 -11.536  -2.992   9.319  84.164

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    2.415e+02  1.439e+02   1.678  0.09550 .
Solar.R        2.198e-01  1.908e-01   1.152  0.25132
Wind          -8.645e+00  7.024e+00  -1.231  0.22047
Temp          -5.764e+00  3.174e+00  -1.816  0.07149 .
I(Solar.R^2) -2.964e-04  2.242e-04  -1.322  0.18841
I(Wind^2)      3.847e-01  1.155e-01   3.329  0.00111 **
I(Temp^2)      4.433e-02  1.854e-02   2.391  0.01811 *
Solar.R:Wind -1.115e-02  5.592e-03  -1.994  0.04808 *
Solar.R:Temp  8.931e-04  2.265e-03   0.394  0.69396
Wind:Temp    -1.486e-03  7.155e-02  -0.021  0.98346
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 18.25 on 143 degrees of freedom
Multiple R-squared:  0.6195,        Adjusted R-squared:  0.5956
F-statistic: 25.87 on 9 and 143 DF,  p-value: < 2.2e-16
```

2.1.1 Residual Diagnostics on Model M3:

- **Linearity and Homoscedasticity Checks:**
  Using the below code we create the plot (2.1) which aid in checks for Linearity and Homoscedasticity.

```r
# Plot residuals vs fitted values
plot(M3$fitted.values, resid(M3), xlab = "Fitted Values", ylab =
↪  "Residuals", main = "Residuals vs Fitted")
abline(h = 0, col = "red")
```

- **Normality Check of Residuals:**
  Using the QQ test we can check whether the residuals are approximatelly normally distributed using the plot (2.2).

```r
qqnorm(resid(M3))
qqline(resid(M3), col = "red")
```

- **Correlation in Residuals:**
  To check for Serial Correlation I will use the Durbin-Watson test.

```r
#test for serial correlation
library(lmtest)
dwtest(M3)
```

```
    Durbin-Watson test

data:  M3
DW = 1.7516, p-value = 0.04162
alternative hypothesis: true autocorrelation is greater than 0
```

As we can see from the plot (2.1) there is some concern because of a line formation from around the 15_{nth} to the 70_{nth} Fitted value which indicates a linear pattern and could present an issue with the non-linearity assumption additionally from the variance the rest of the values have we can solidifie our assumption of heteroscedasticity. The QQ Plot (2.2) seems to fall in line suggesting that the residuals are mostly normally distributed. Finally the Durbin-Watson Test results provide a wd close to 2 with a p-value less than 0.05 that could suggest a statistical significance in positive autocorrelation by some statisticians, thought for as is just an indicator to do more analysis.

Conclusion remarks: The residuals plot's indication of heteroscedasticity and non-linearity raises the possibility that the model would benefit from variable transformation or the addition of more pertinent variables even thought the normality of residuals is a positive aspect and the positive autocorrelation found by Durbin-Watson suggests that the residuals might not be completely independent.

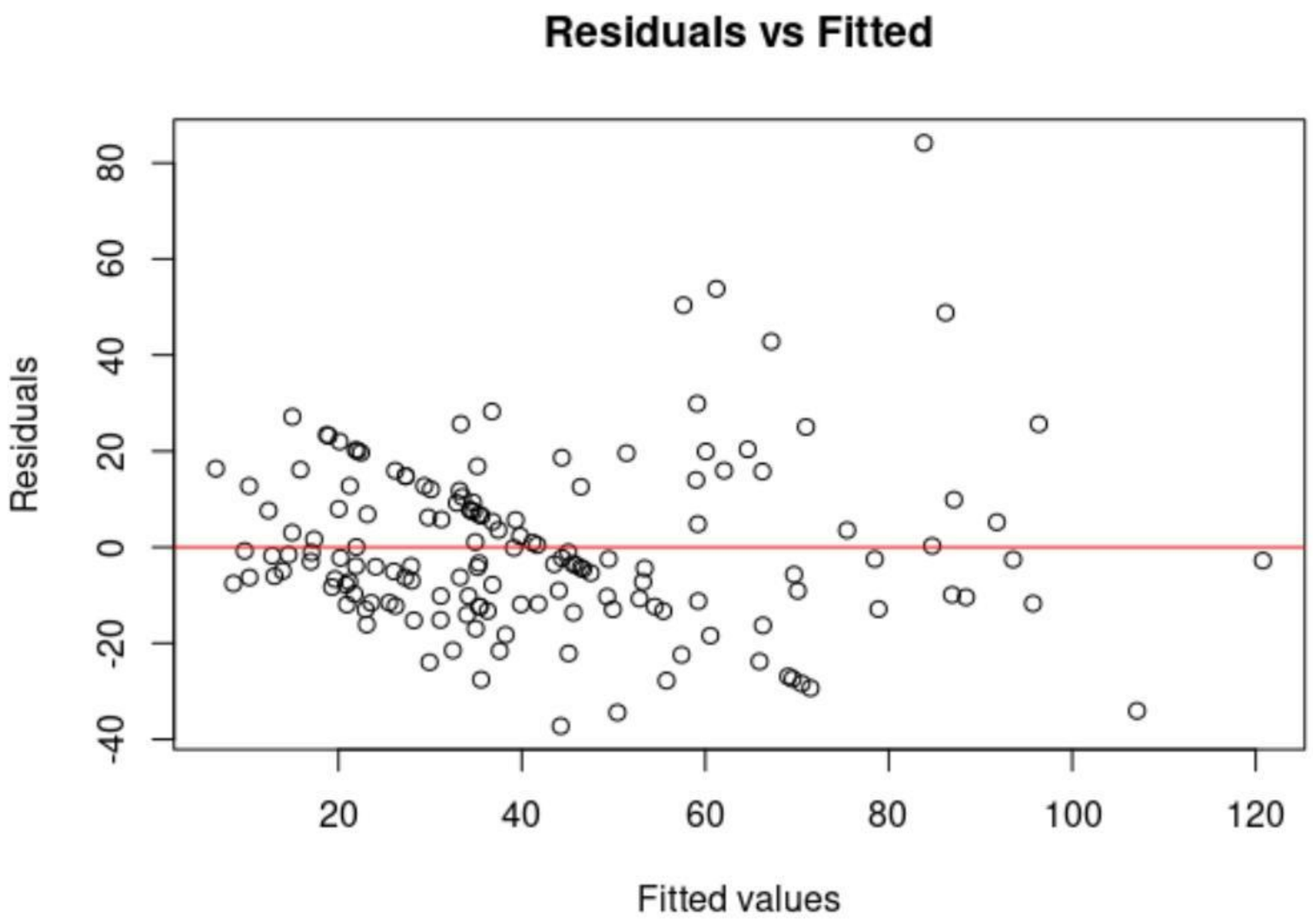

Figure 2.1: Residuals vs values

Before we proceed I will also apply the VIF, through a custom function [1] **:**

```
# Function to calculate VIF
calc_vif <- function(model) {
  # Get the model matrix
  model_matrix <- model.matrix(model)

  # Initialize a vector to store VIF values
  vif_values <- numeric(ncol(model_matrix) - 1)
```

[1] I faced a problem with olsrr dependencies, specifically with car

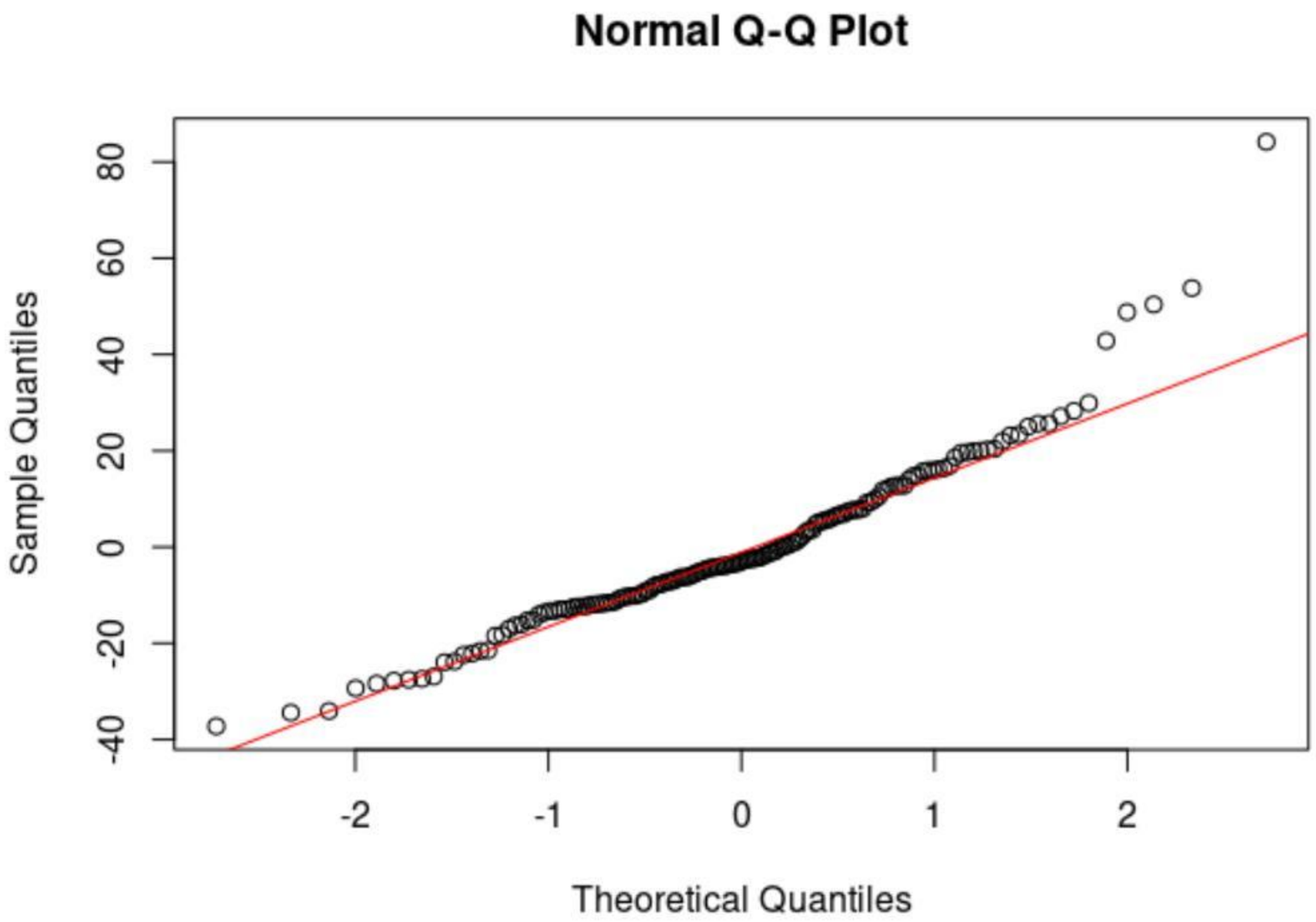

Figure 2.2: Normality Check

```
# Loop through each predictor
  for (i in 2:ncol(model_matrix)) {
    # Model excluding the predictor
    mod <- lm(model_matrix[, i] ~ model_matrix[, -i])

    # Calculate VIF
    vif_values[i - 1] <- 1 / (1 - summary(mod)$r.squared)
  }

  # Return the VIF values
  names(vif_values) <- names(model_matrix)[2:ncol(model_matrix)]
  return(vif_values)
}

> calc_vif(M3)
[1] 128.56420 279.55016 412.05497  21.06646  35.57512 327.53901  19.62788
↪  122.13576 141.75817
```

These results indicate varying degrees of multicollinearity among the predictors with values like 128.56, 279.55, and 412.05 extremely high and even the lower values pass the threshold of 5 or 10, suggesting that the predictors in my model are highly correlated with each other.

2.1.2 Box-Cox Transformation:

From the Box-Cox plot (2.3) we can observe the dotted vertical lines represent a confidence interval for the λ value which maximizes log-likelihood and can be chosen

for the transformation. The true λ value lies within this interval with 95% confidence level.

```
# Box-Cox Transformation
bc<- boxcox(M3)

# The optimal lambda value
lambda_opt <- bc$x[which.max(bc$y)]

> lambda_opt
[1] 0.4242424
```

Now based on the optimal λ value we apply the BC to the response variable, we re-fit the model and perform residual diagnostics (2.4).

```
# Adding a small constant to avoid log of zero
airquality_clean$Ozone_transformed <- ifelse(lambda_opt == 0,
                                             log(airquality_clean$Ozone +
                                             ↪  1),

                                             ↪  (airquality_clean$Ozone^lambda_opt
                                             ↪  - 1) / lambda_opt)

# The model with  transformed response variable
M3_transformed <- lm(Ozone_transformed ~ Solar.R + Wind + Temp +
↪  I(Solar.R^2) + I(Wind^2) + I(Temp^2) +
                     Solar.R:Wind + Solar.R:Temp + Wind:Temp, data =
                     ↪  airquality_clean)

# Residual diagnostics
par(mfrow=c(2,2))
plot(M3_transformed, which = 1:4)
```

From the residuals vs fitted plot we observe homoscedasticity. From what I other the scale location plot shows that the residuals are homogenous due to the lack of pattern and finally Cooks's distance indicates no large distances which implies none influential points affecting the model.

2.1.3 Noise:

To create three new predictors that have a correlation of approximately 0.8 with the original predictors we will mix the original variable with some noise, by taking a weighted sum of the original variable and a noise component with weights determined by the desired correlation coefficients. Using scale to standardize the original variables we get:

```
n <- nrow(airquality_clean)

# Correlation
rho <- 0.8

# Generate noise with the desired correlation
```

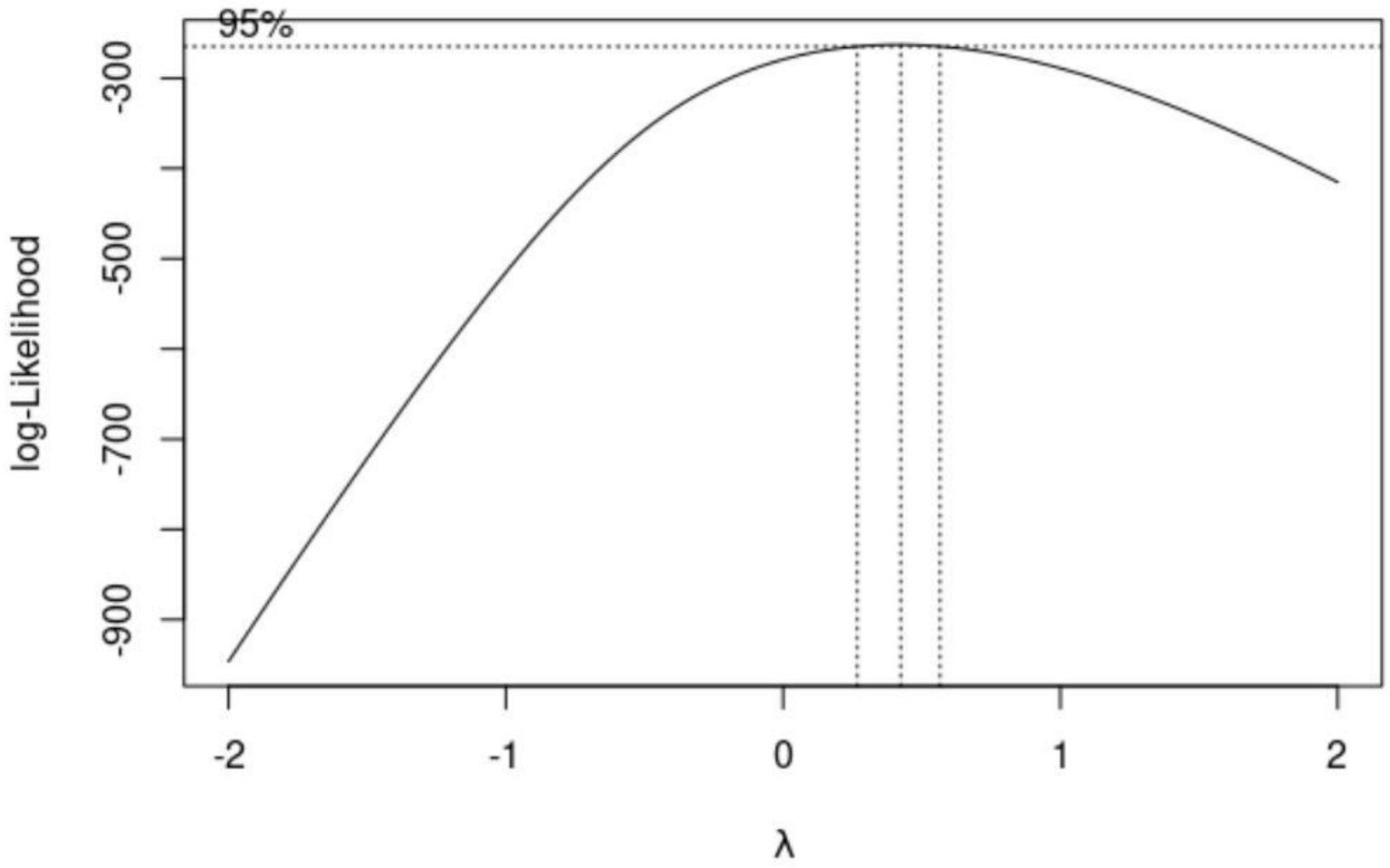

Figure 2.3: Box-Cox

```
noise_data <- mvrnorm(n, mu = c(0, 0, 0), Sigma = matrix(c(1, rho, rho,
↪  rho, 1, rho, rho, rho, 1), nrow = 3))

# Create new predictors by adding the noise to the scaled original
↪  variables
airquality_clean$Z1 <- scale(airquality_clean$Solar.R) * sqrt(rho) +
↪  noise_data[, 1] * sqrt(1 - rho)
airquality_clean$Z2 <- scale(airquality_clean$Wind) * sqrt(rho) +
↪  noise_data[, 2] * sqrt(1 - rho)
airquality_clean$Z3 <- scale(airquality_clean$Temp) * sqrt(rho) +
↪  noise_data[, 3] * sqrt(1 - rho)

# The correlation
cor(airquality_clean$Solar.R, airquality_clean$Z1) # 0.9012986
cor(airquality_clean$Wind, airquality_clean$Z2)    # 0.9115918
cor(airquality_clean$Temp, airquality_clean$Z3)    # 0.8979545
```

and now we create M4 as:

```
M4 <- lm(Ozone ~ Solar.R + Wind + Temp + I(Solar.R^2) + I(Wind^2) +
↪  I(Temp^2) +
         Solar.R:Wind + Solar.R:Temp + Wind:Temp + Z1 + Z2 + Z3,
         data = airquality_clean)
```

with summary:

```
Call:
lm(formula = Ozone ~ Solar.R + Wind + Temp + I(Solar.R^2) + I(Wind^2) +
    I(Temp^2) + Solar.R:Wind + Solar.R:Temp + Wind:Temp + Z1 +
```

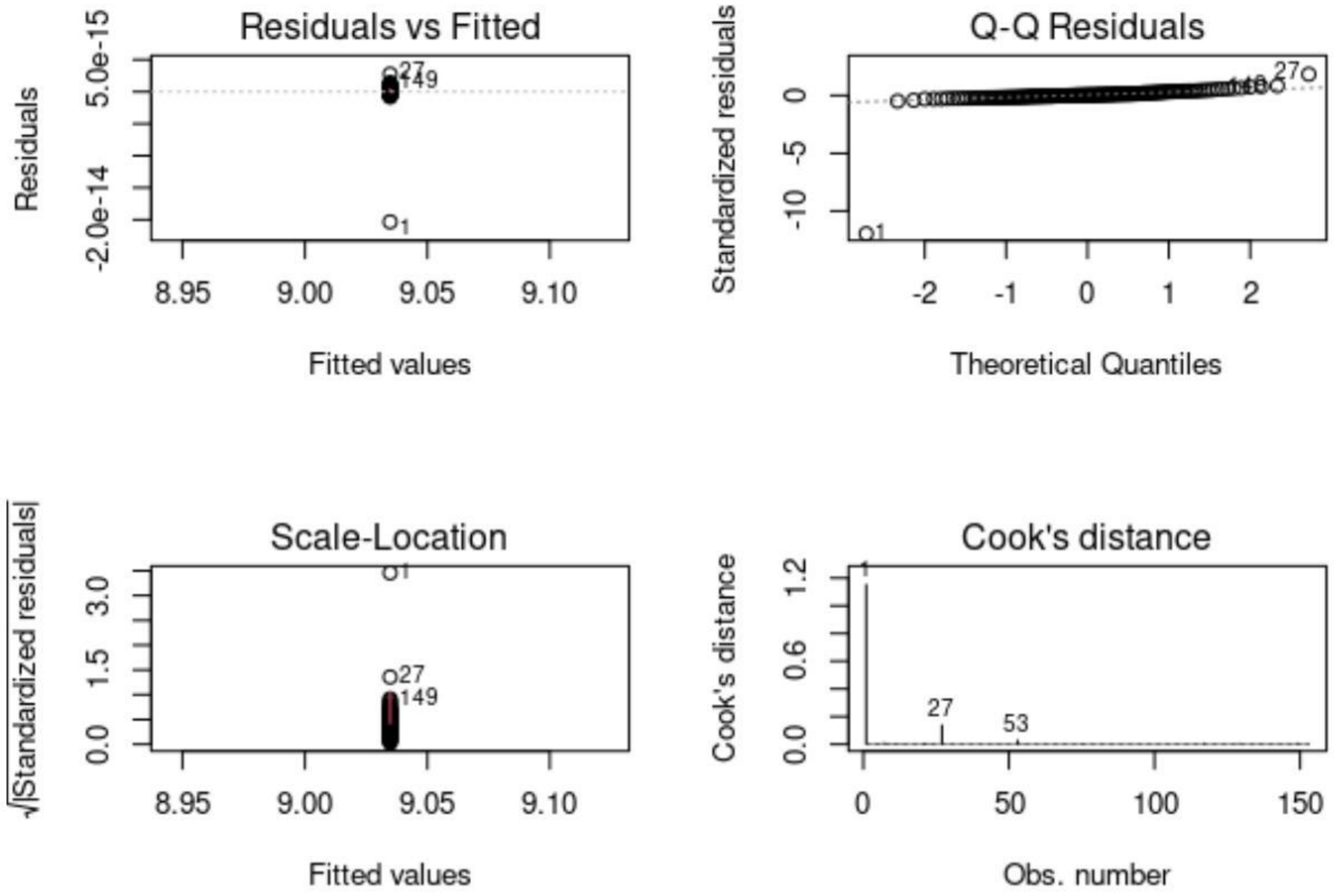

Figure 2.4: Residual diagnostics

```
    Z2 + Z3, data = airquality_clean)

Residuals:
    Min      1Q  Median      3Q     Max
-38.754 -10.893  -2.601  11.736  84.831

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.876e+02  1.478e+02   1.270  0.20636
Solar.R       2.089e-01  1.964e-01   1.064  0.28936
Wind         -9.747e+00  7.047e+00  -1.383  0.16881
Temp         -4.606e+00  3.233e+00  -1.424  0.15654
I(Solar.R^2) -3.339e-04  2.266e-04  -1.474  0.14285
I(Wind^2)     3.675e-01  1.168e-01   3.146  0.00202 **
I(Temp^2)     4.271e-02  1.873e-02   2.280  0.02412 *
Z1            3.065e+00  6.401e+00   0.479  0.63281
Z2            8.966e+00  6.210e+00   1.444  0.15100
Z3           -8.508e+00  6.766e+00  -1.258  0.21065
Solar.R:Wind -1.072e-02  5.650e-03  -1.897  0.05986 .
Solar.R:Temp  7.805e-04  2.281e-03   0.342  0.73276
Wind:Temp    -1.317e-02  7.272e-02  -0.181  0.85656
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Residual standard error: 18.22 on 140 degrees of freedom
Multiple R-squared:  0.6284,        Adjusted R-squared:  0.5966
F-statistic: 19.73 on 12 and 140 DF,  p-value: < 2.2e-16
```

2.1.4 Tofallis

The paper [Tof15] discusses a method for model selection by minimizing the sum of squares of $\log Q$ (the log of the accuracy ratio) , the method fits models that predict the geometric mean providing a form of regression that is less likely to underpredict compared to other models that minimize the mean absolute percentage error. The code below will transform the response, fit the model and perform diagnostics.

```
airquality_clean$Ozone_log <- log(airquality_clean$Ozone)

# Initial Model
initial_model <- lm(Ozone ~ Solar.R + Wind + Temp + I(Solar.R^2) +
↪  I(Wind^2) + I(Temp^2) +
                      Solar.R:Wind + Solar.R:Temp + Wind:Temp,
                    data = airquality_clean)


# Predict values using the initial model
predicted_values <- predict(initial_model, newdata = airquality_clean)

# Compute the ratio Q (Predicted / Actual)
Q <- predicted_values / airquality_clean$Ozone

# Apply the logarithm to Q (avoid division by zero)
ln_Q <- log(Q + 1e-6)  # Adding a small constant to avoid log(0)

# Fit the model M5 using ln_Q
M5 <- lm(ln_Q ~ Solar.R + Wind + Temp + I(Solar.R^2) + I(Wind^2) +
↪  I(Temp^2) +
           Solar.R:Wind + Solar.R:Temp + Wind:Temp,
         data = airquality_clean)

# Residual diagnostics on M5
par(mfrow=c(2,2))
plot(M5, which = 1:4)
```

with summary:

```
Call:
lm(formula = Ozone_log ~ Solar.R + Wind + Temp + I(Solar.R^2) +
    I(Wind^2) + I(Temp^2) + Solar.R:Wind + Solar.R:Temp + Wind:Temp +
    Z1 + Z2 + Z3, data = airquality_clean)

Residuals:
     Min       1Q   Median       3Q      Max
-1.78457 -0.29611  0.03896  0.33011  1.20700

Coefficients:
```

```
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    1.016e+00  4.134e+00   0.246  0.80617
Solar.R        1.126e-02  5.492e-03   2.049  0.04229 *
Wind          -1.454e-01  1.971e-01  -0.738  0.46199
Temp          -1.004e-02  9.043e-02  -0.111  0.91175
I(Solar.R^2)  -1.871e-05  6.338e-06  -2.952  0.00370 **
I(Wind^2)      5.064e-03  3.267e-03   1.550  0.12342
I(Temp^2)      7.275e-04  5.239e-04   1.389  0.16715
Z1             1.799e-01  1.790e-01   1.005  0.31659
Z2             3.554e-01  1.737e-01   2.046  0.04262 *
Z3            -5.070e-01  1.892e-01  -2.679  0.00826 **
Solar.R:Wind  -5.730e-05  1.580e-04  -0.363  0.71742
Solar.R:Temp  -4.937e-05  6.381e-05  -0.774  0.44039
Wind:Temp     -1.049e-03  2.034e-03  -0.516  0.60697
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5097 on 140 degrees of freedom
Multiple R-squared:  0.5915,        Adjusted R-squared:  0.5565
F-statistic:  16.9 on 12 and 140 DF,  p-value: < 2.2e-16
```

The residual vs fitted plot indicates no pattern within our residuals as they are randomly dispersed thus the model does not suffer from non-linearity issues and seems that the homoskedastic assumption holds. The Q-Q plot points mostly follow the reference line and in the scale-loc plot, the spread is fairly even suggesting that the variance of the residuals is relatively constant. Finally, there are no points with high Cooks distance, thus no major influential points.
The transformation is different from BC as the λ was close to 0.5 the BC provided a power transformation less extreme than the logarithmic due to λ. So while the transformations are conceptually similar because they both work on the principle of power transformation they are technically different as long as λ is not too close to 0. Though most of the above do hold for M4 2.6 the scale of residuals is much larger than M5's.

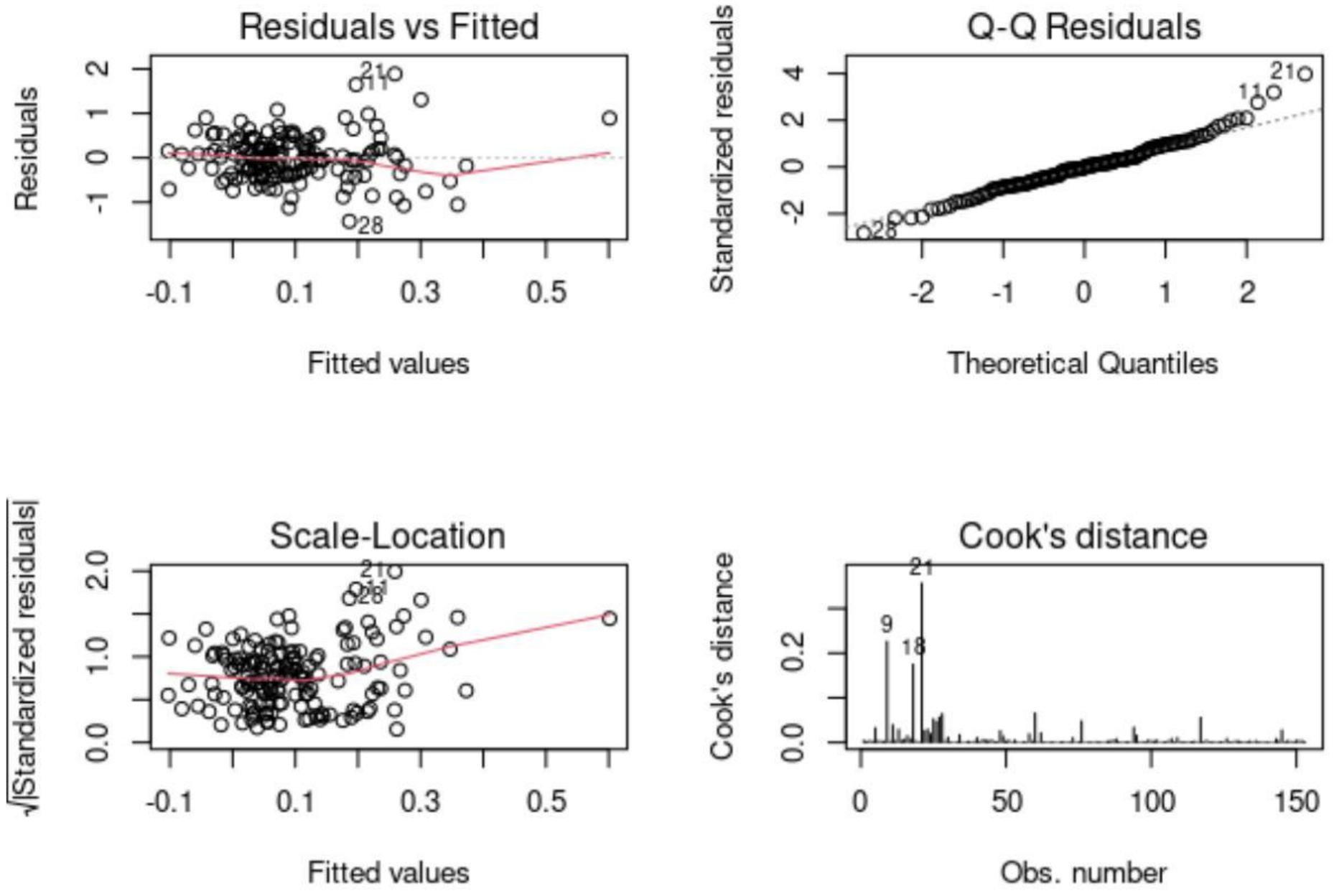

Figure 2.5: Diagnostics on M5

2.2 Part 2: Predictor Selection Tools on Model M5

Various tools for predictor selection on Model M5 will be examined. This includes BIC-based forward and backward selection, comparison with AIC-based methods, and the application of LASSO and ridge regression. The efficacy of each method will be assessed.

2.2.1 BIC-based Forward and Backward Selection

```
# Forward selection using BIC
f_bic <- regsubsets(Ozone ~ Solar.R + Wind + Temp + I(Solar.R^2) +
↪  I(Wind^2) + I(Temp^2) +
                    Solar.R:Wind + Solar.R:Temp + Wind:Temp + Z1 + Z2 +
                    ↪  Z3,
                    data = airquality_clean, nbest = 1, method =
                    ↪  "forward", really.big = TRUE)

# Backward selection using BIC
b_bic <- regsubsets(Ozone ~ Solar.R + Wind + Temp + I(Solar.R^2) +
↪  I(Wind^2) + I(Temp^2) +
                    Solar.R:Wind + Solar.R:Temp + Wind:Temp + Z1 + Z2 +
                    ↪  Z3,
```

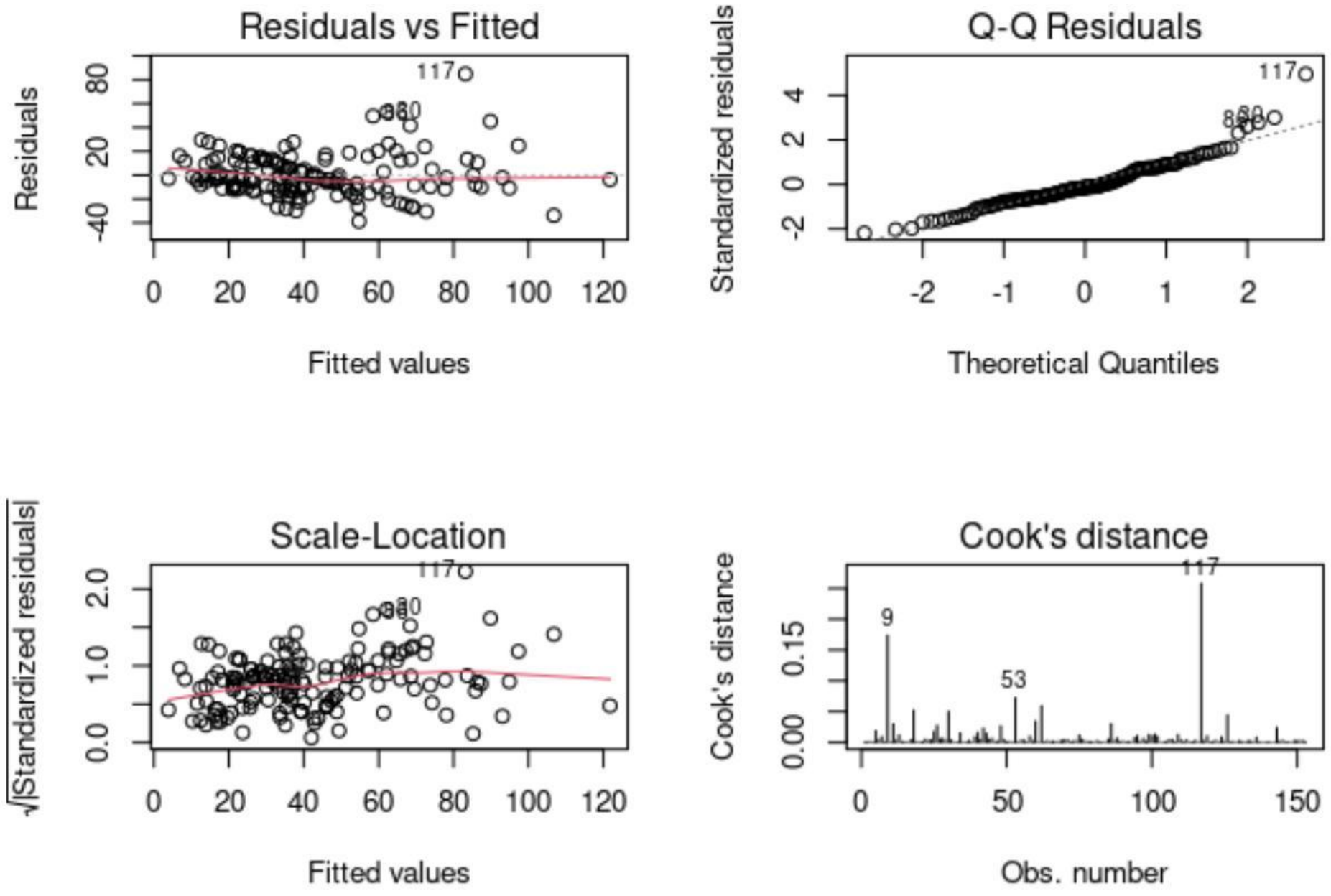

Figure 2.6: Diagnostics on M4

```
                    data = airquality_clean, nbest = 1, method =
                    ↪  "backward", really.big = TRUE)
```

with summaries:

```
> summary(f_bic)
Subset selection object
Call: regsubsets.formula(Ozone ~ Solar.R + Wind + Temp + I(Solar.R^2) +
    I(Wind^2) + I(Temp^2) + Solar.R:Wind + Solar.R:Temp + Wind:Temp +
    Z1 + Z2 + Z3, data = airquality_clean, nbest = 1, method = "forward",
    really.big = TRUE)
12 Variables  (and intercept)
             Forced in Forced out
Solar.R          FALSE      FALSE
Wind             FALSE      FALSE
Temp             FALSE      FALSE
I(Solar.R^2)     FALSE      FALSE
I(Wind^2)        FALSE      FALSE
I(Temp^2)        FALSE      FALSE
Z1               FALSE      FALSE
Z2               FALSE      FALSE
Z3               FALSE      FALSE
Solar.R:Wind     FALSE      FALSE
Solar.R:Temp     FALSE      FALSE
Wind:Temp        FALSE      FALSE
```

```
1 subsets of each size up to 8
Selection Algorithm: forward
         Solar.R Wind Temp I(Solar.R^2) I(Wind^2) I(Temp^2) Z1  Z2  Z3
          ↪  Solar.R:Wind Solar.R:Temp Wind:Temp
1  ( 1 ) " "     " "  " "  " "           " "         "*"         " " " " " " " " "
 ↪  "           " "            " "
2  ( 1 ) " "     " "  " "  " "           " "         "*"         " " " " " " " " "
 ↪  "           " "            "*"
3  ( 1 ) " "     " "  " "  " "           "*"         "*"         " " " " " " " " "
 ↪  "           " "            "*"
4  ( 1 ) " "     " "  " "  " "           "*"         "*"         "*" " " " " " " "
 ↪  "           " "            "*"
5  ( 1 ) " "     " "  "*"  " "           "*"         "*"         "*" " " " " " " "
 ↪  "           " "            "*"
6  ( 1 ) " "     " "  "*"  " "           "*"         "*"         "*" " " "*" " "
 ↪  "           " "            "*"
7  ( 1 ) " "     " "  "*"  " "           "*"         "*"         "*" "*" "*" " "
 ↪  "           " "            "*"
8  ( 1 ) " "     " "  "*"  " "           "*"         "*"         "*" "*" "*" " "
 ↪  "           "*"            "*"
> summary(b_bic)
Subset selection object
Call: regsubsets.formula(Ozone ~ Solar.R + Wind + Temp + I(Solar.R^2) +
    I(Wind^2) + I(Temp^2) + Solar.R:Wind + Solar.R:Temp + Wind:Temp +
    Z1 + Z2 + Z3, data = airquality_clean, nbest = 1, method = "backward",
    really.big = TRUE)
12 Variables  (and intercept)
             Forced in Forced out
Solar.R          FALSE      FALSE
Wind             FALSE      FALSE
Temp             FALSE      FALSE
I(Solar.R^2)     FALSE      FALSE
I(Wind^2)        FALSE      FALSE
I(Temp^2)        FALSE      FALSE
Z1               FALSE      FALSE
Z2               FALSE      FALSE
Z3               FALSE      FALSE
Solar.R:Wind     FALSE      FALSE
Solar.R:Temp     FALSE      FALSE
Wind:Temp        FALSE      FALSE
1 subsets of each size up to 8
Selection Algorithm: backward
         Solar.R Wind Temp I(Solar.R^2) I(Wind^2) I(Temp^2) Z1  Z2  Z3
          ↪  Solar.R:Wind Solar.R:Temp Wind:Temp
1  ( 1 ) " "     " "  " "  " "           " "         "*"         " " " " " " " " "
 ↪  "           " "            " "
2  ( 1 ) " "     "*"  " "  " "           " "         "*"         " " " " " " " " "
 ↪  "           " "            " "
3  ( 1 ) " "     "*"  " "  " "           "*"         "*"         " " " " " " " " "
 ↪  "           " "            " "
```

```
4  ( 1 ) "*"      "*"   " "   " "          "*"      "*"      " " " " " " " " "
↪   "          " "              " "
5  ( 1 ) "*"      "*"   "*"   " "          "*"      "*"      " " " " " " " " "
↪   "          " "              " "
6  ( 1 ) "*"      "*"   "*"   " "          "*"      "*"      " " " " " " " "
↪   "*"        " "                 " "
7  ( 1 ) "*"      "*"   "*"   " "          "*"      "*"      " " "*" " " " "
↪   "*"        " "                 " "
8  ( 1 ) "*"      "*"   "*"   "*"          "*"      "*"      " " "*" " " " "
↪   "*"        " "                 " "
```

The most parsimonious is the Forward BIC with model size 4. This is the result provided by comparing the summaries as it finds the model with that includes the fewest predictors while it provides adequate results.

```
f_bic_summary <- summary(f_bic)


b_bic_summary <- summary(b_bic)

# The model with the lowest BIC in forward selection
min_bic_value_forward <- min(f_bic_summary$bic)
min_bic_forward <- which.min(f_bic_summary$bic)

# The model with the lowest BIC in backward selection
min_bic_value_backward <- min(b_bic_summary$bic)
min_bic_backward <- which.min(b_bic_summary$bic)

# Compare
most_parsimonious <- ifelse(min_bic_value_forward <
↪  min_bic_value_backward,
                            paste("Forward BIC with model size",
                            ↪  min_bic_forward),
                            paste("Backward BIC with model size",
                            ↪  min_bic_backward))

most_parsimonious
```

2.2.2 Compare F-BIC, B-BIC to F-AIC

```
# Forward selection using AIC
f_aic <- step(M5, direction = "forward", scope = list(upper = ~ Solar.R +
↪  Wind + Temp + I(Solar.R^2) + I(Wind^2) + I(Temp^2) +
                                                      Solar.R:Wind +
                                                      ↪  Solar.R:Temp +
                                                      ↪  Wind:Temp + Z1
                                                      ↪  + Z2 + Z3),
              trace = FALSE)
summary(f_aic)

# Backward selection using AIC
b_aic <- step(M5, direction = "backward", trace = FALSE)
```

with summary:

```
> summary(f_aic)

Call:
lm(formula = Ozone_log ~ Solar.R + Wind + Temp + I(Solar.R^2) +
    I(Wind^2) + I(Temp^2) + Solar.R:Wind + Solar.R:Temp + Wind:Temp +
    Z1 + Z2 + Z3, data = airquality_clean)

Residuals:
     Min       1Q   Median       3Q      Max
-1.78457 -0.29611  0.03896  0.33011  1.20700

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    1.016e+00  4.134e+00   0.246  0.80617
Solar.R        1.126e-02  5.492e-03   2.049  0.04229 *
Wind          -1.454e-01  1.971e-01  -0.738  0.46199
Temp          -1.004e-02  9.043e-02  -0.111  0.91175
I(Solar.R^2) -1.871e-05  6.338e-06  -2.952  0.00370 **
I(Wind^2)      5.064e-03  3.267e-03   1.550  0.12342
I(Temp^2)      7.275e-04  5.239e-04   1.389  0.16715
Z1             1.799e-01  1.790e-01   1.005  0.31659
Z2             3.554e-01  1.737e-01   2.046  0.04262 *
Z3            -5.070e-01  1.892e-01  -2.679  0.00826 **
Solar.R:Wind -5.730e-05  1.580e-04  -0.363  0.71742
Solar.R:Temp -4.937e-05  6.381e-05  -0.774  0.44039
Wind:Temp    -1.049e-03  2.034e-03  -0.516  0.60697
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5097 on 140 degrees of freedom
Multiple R-squared:  0.5915,        Adjusted R-squared:  0.5565
F-statistic:  16.9 on 12 and 140 DF,  p-value: < 2.2e-16

>summary(b_aic)

Call:
lm(formula = Ozone_log ~ Solar.R + Wind + I(Solar.R^2) + I(Wind^2) +
    I(Temp^2) + Z2 + Z3, data = airquality_clean)

Residuals:
     Min       1Q   Median       3Q      Max
-1.89171 -0.31762  0.03153  0.31547  1.32672

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    1.696e+00  6.198e-01   2.737 0.006979 **
Solar.R        9.165e-03  2.079e-03   4.408 2.01e-05 ***
Wind          -2.832e-01  6.345e-02  -4.464 1.60e-05 ***
I(Solar.R^2) -2.003e-05  5.938e-06  -3.373 0.000954 ***
I(Wind^2)      6.484e-03  2.393e-03   2.710 0.007545 **
```

```
I(Temp^2)     4.949e-04  1.107e-04   4.469 1.57e-05 ***
Z2            4.272e-01  1.577e-01   2.710 0.007544 **
Z3           -4.500e-01  1.701e-01  -2.645 0.009068 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5063 on 145 degrees of freedom
Multiple R-squared:  0.5826,        Adjusted R-squared:  0.5625
F-statistic: 28.92 on 7 and 145 DF,  p-value: < 2.2e-16
```

Now comparing these models with F-BIC and B-BIC we observe that the BIC models both concluded in the usage of 8 predictors as opposed to the AIC's which kept 7 in the backwards and 12 in the forward. To determine the most parsimonious model, you would typically choose the one with the lowest AIC or BIC value. Given that BIC tends to penalize complexity more heavily than AIC, a model with a lower BIC might be considered more parsimonious, especially if the dataset is large, thus the original parsimonious model.

2.2.3 Comare to the F-P and B-P:

The forward and backward defer from AIC and BIC in their approach of including or excluding predictors, as they rely on statistical significance a term much misunderstood and faulty. In contrast BIC and AIC consider the overall model fit and complexity, with BIC preferring simpler models, especially in large datasets, while AIC is more lenient with model complexity and focuses on the goodness of fit. An other difference is in their assessment phase-time as F-P and B-P are assessing predictors one at a time, where the others evaluate the model as a whole.[2]

```
# Start with the intercept
model_fp <- lm(Ozone ~ 1, data = airquality_clean)

# Iteratively add predictors based on p-value
model_fp <- step(model_fp,
                 scope = list(lower = ~ 1,
                              upper = ~ Solar.R + Wind + Temp +
                              ↪  I(Solar.R^2) + I(Wind^2) + I(Temp^2) +
                                Solar.R:Wind + Solar.R:Temp + Wind:Temp +
                                ↪  Z1 + Z2 + Z3),
                 direction = "forward",
                 test = "F")
```

```
# Start with the full model
model_bp <- lm(Ozone ~ Solar.R + Wind + Temp + I(Solar.R^2) + I(Wind^2) +
↪  I(Temp^2) +
                 Solar.R:Wind + Solar.R:Temp + Wind:Temp + Z1 + Z2 + Z3,
               data = airquality_clean)

# Iteratively remove predictors based on p-value
model_bp <- step(model_bp, direction = "backward", test = "F")
```

[2]Due to a dependence error I will not use olsrr

with results:

```
> summary(model_fp)

Call:
lm(formula = Ozone ~ I(Temp^2) + Wind + I(Wind^2) + Z1 + Temp,
    data = airquality_clean)

Residuals:
    Min      1Q  Median      3Q     Max
-38.413 -10.509  -3.802  10.157  88.170

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 310.53726   81.79403   3.797 0.000214 ***
I(Temp^2)     0.05132    0.01438   3.569 0.000484 ***
Wind        -11.32402    1.94954  -5.809 3.75e-08 ***
I(Wind^2)     0.39753    0.08866   4.484 1.47e-05 ***
Z1            5.96760    1.69759   3.515 0.000584 ***
Temp         -6.62283    2.18880  -3.026 0.002928 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 18.45 on 147 degrees of freedom
Multiple R-squared:  0.5999,        Adjusted R-squared:  0.5863
F-statistic: 44.09 on 5 and 147 DF,  p-value: < 2.2e-16
```

```
> summary(model_bp)

Call:
lm(formula = Ozone ~ Solar.R + Wind + Temp + I(Wind^2) + I(Temp^2) +
    Solar.R:Wind, data = airquality_clean)

Residuals:
    Min      1Q  Median      3Q     Max
-35.020 -11.414  -3.319   9.457  84.364

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)  249.234923  85.949007   2.900 0.004311 **
Solar.R        0.190783   0.056615   3.370 0.000962 ***
Wind          -8.809996   2.139751  -4.117 6.39e-05 ***
Temp          -6.002178   2.216168  -2.708 0.007570 **
I(Wind^2)      0.386813   0.087361   4.428 1.85e-05 ***
I(Temp^2)      0.047263   0.014536   3.251 0.001426 **
Solar.R:Wind  -0.011666   0.004881  -2.390 0.018123 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 18.17 on 146 degrees of freedom
Multiple R-squared:  0.6148,        Adjusted R-squared:  0.599
F-statistic: 38.84 on 6 and 146 DF,  p-value: < 2.2e-16
```

From their summaries we see that F-P includes 5 predictors while B-P includes 6 while both have similar Residual Standart Errors.

2.2.4 LASSO and Ridge

The model matrix you see below in the R code was chosen to reflect the same predictors as used in model M5 per our instructions. This consistency ensures also that the effects of LASSO and Ridge regularization are applied to the same set of variables, allowing for the models comparison.

```
library(glmnet)
x <- model.matrix(Ozone ~ Solar.R + Wind + Temp + I(Solar.R^2) + I(Wind^2)
↪  + I(Temp^2) +
                    Solar.R:Wind + Solar.R:Temp + Wind:Temp + Z1 + Z2 +
                     ↪  Z3,
                  data = airquality_clean)[, -1] # remove intercept
y <- log(airquality_clean$Ozone)

# LASSO
lasso_cv <- cv.glmnet(x, y, alpha = 1)
par(mfrow = c(1, 1))
plot(lasso_cv)

# Ridge
ridge_cv <- cv.glmnet(x, y, alpha = 0)
par(mfrow = c(1, 1))
plot(ridge_cv)
```

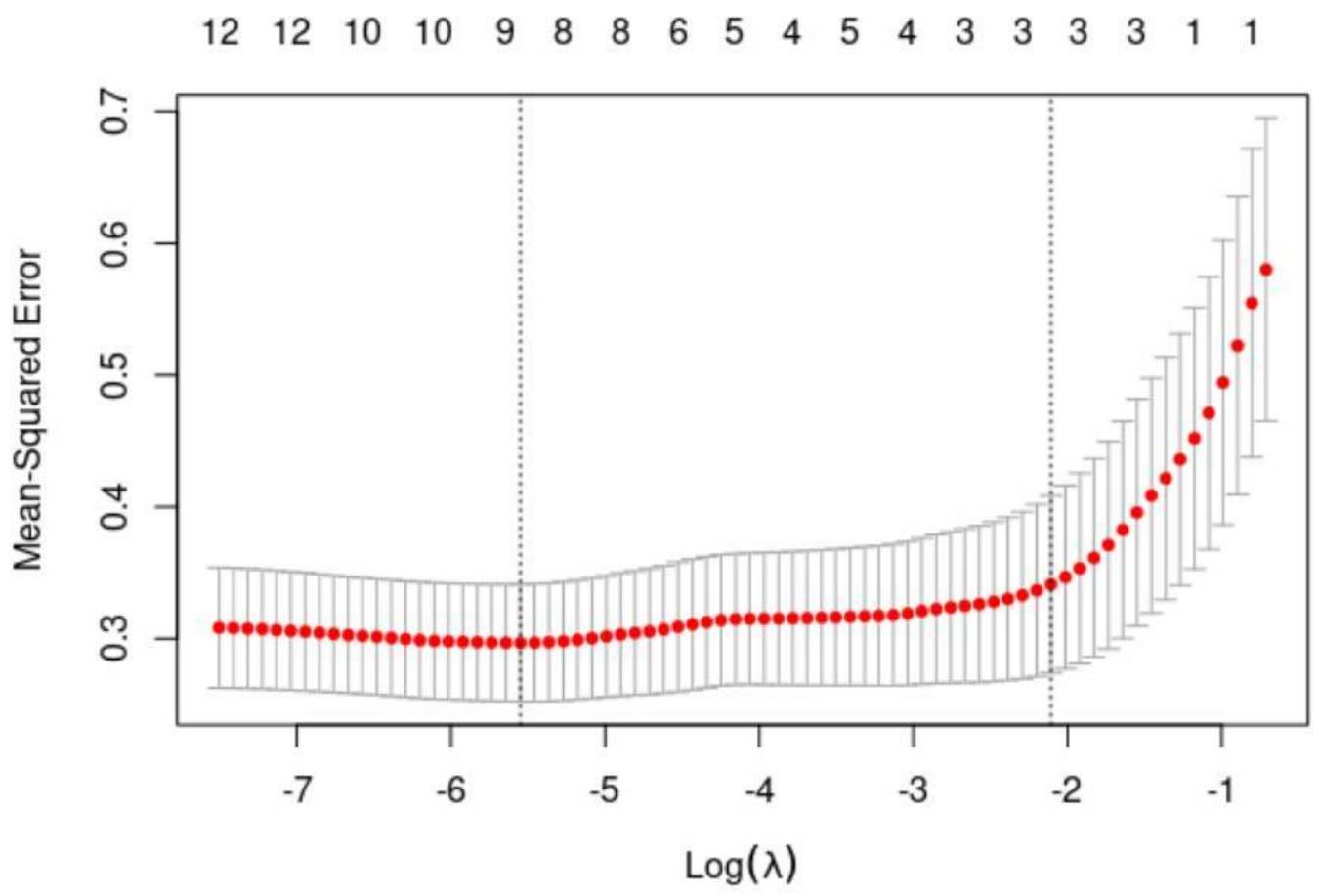

Figure 2.7: LASSO

plotted in (2.7) and (2.8) and with summaries:

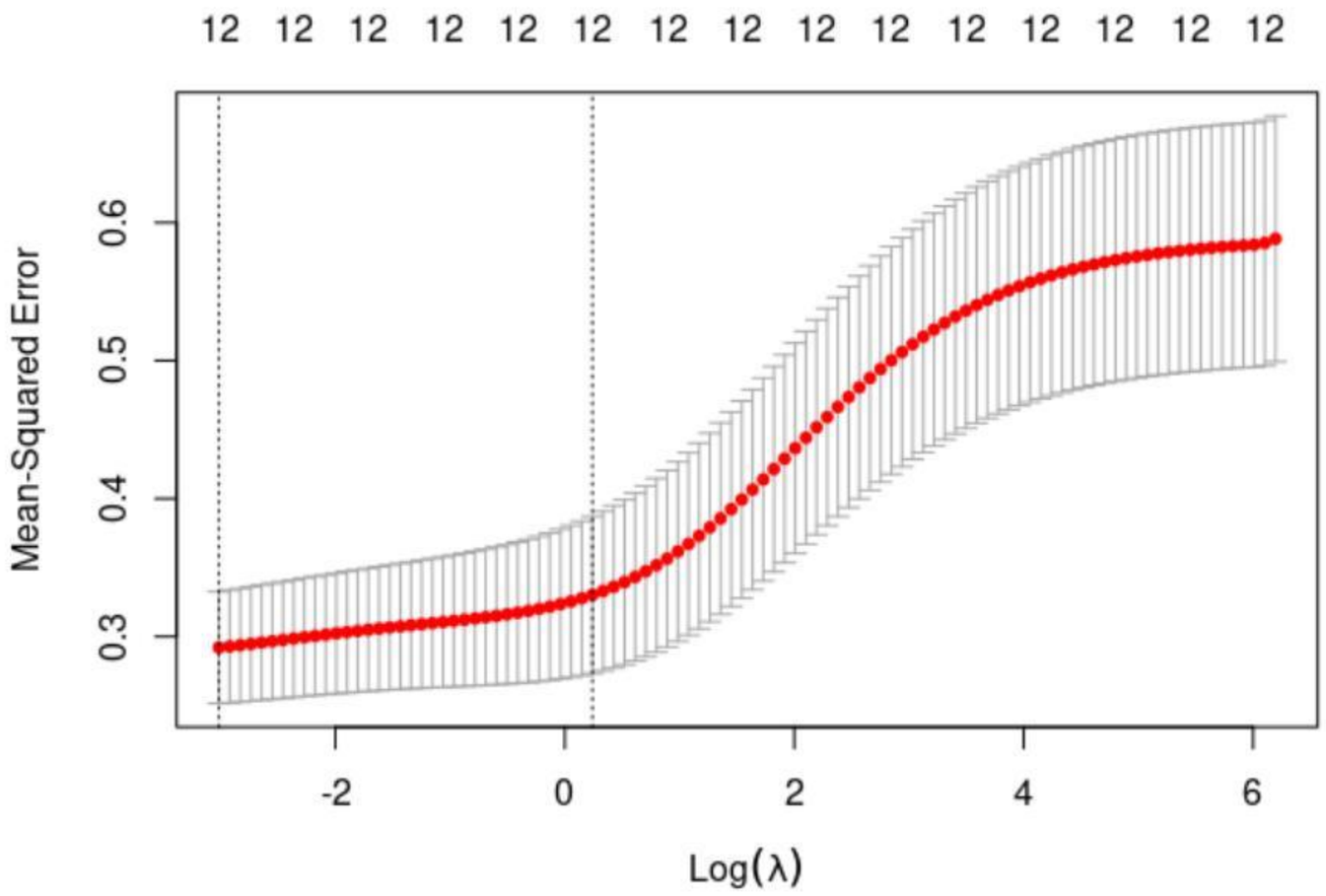

Figure 2.8: Ridge

```
> summary(ridge_cv)
           Length Class  Mode
lambda     100    -none- numeric
cvm        100    -none- numeric
cvsd       100    -none- numeric
cvup       100    -none- numeric
cvlo       100    -none- numeric
nzero      100    -none- numeric
call         4    -none- call
name         1    -none- character
glmnet.fit  12    elnet  list
lambda.min   1    -none- numeric
lambda.1se   1    -none- numeric
index        2    -none- numeric

> summary(lasso_cv)
           Length Class  Mode
lambda     74     -none- numeric
cvm        74     -none- numeric
cvsd       74     -none- numeric
cvup       74     -none- numeric
cvlo       74     -none- numeric
nzero      74     -none- numeric
call        4     -none- call
name        1     -none- character
glmnet.fit 12     elnet  list
lambda.min  1     -none- numeric
lambda.1se  1     -none- numeric
index       2     -none- numeric
```

Now using the code below we can better observe our comparison:
An important note is that the **M5 is immensely better than M4** in AIC and BIC, and the M4 is very similar to F-P, B-P F-BIC, and B-BIC. The difference is enough that could indicate a different scale of measurement, or a different model structure altogether. This is exactly the case as M5 used the logarithm of our data as did LASSO and Ridge so an immediate comparison can not be made, we can re-create M4 and the others, trained on the same logarithmic data for a clearer perspective. The above statements are all wrong in that sense...

```
# The re-creation
airquality_clean$Log_Ozone <- log(airquality_clean$Ozone)

M4_log <- lm(Log_Ozone ~ Solar.R + Wind + Temp + I(Solar.R^2) + I(Wind^2)
↪  + I(Temp^2) +
               Solar.R:Wind + Solar.R:Temp + Wind:Temp + Z1 + Z2 + Z3,
             data = airquality_clean)


# F-P
model_fp_log <- step(lm(Log_Ozone ~ 1, data = airquality_clean),
                     scope = list(lower = ~ 1,
                                  upper = ~ Solar.R + Wind + Temp +
                                  ↪  I(Solar.R^2) + I(Wind^2) + I(Temp^2)
                                  ↪  +
                                    Solar.R:Wind + Solar.R:Temp +
                                    ↪  Wind:Temp + Z1 + Z2 + Z3),
                     direction = "forward",
                     test = "F")

# B-P
model_bp_log <- step(lm(Log_Ozone ~ Solar.R + Wind + Temp + I(Solar.R^2) +
↪  I(Wind^2) + I(Temp^2) +
                          Solar.R:Wind + Solar.R:Temp + Wind:Temp + Z1 +
                          ↪  Z2 + Z3,
                        data = airquality_clean),
                     direction = "backward",
                     test = "F")


library(leaps)

# F-BIC
f_bic_log <- regsubsets(Log_Ozone ~ Solar.R + Wind + Temp + I(Solar.R^2) +
↪  I(Wind^2) + I(Temp^2) +
                          Solar.R:Wind + Solar.R:Temp + Wind:Temp + Z1 +
                          ↪  Z2 + Z3,
                        data = airquality_clean, nbest = 1, method =
                        ↪  "forward", really.big = TRUE)

# B-BIC
```

```
b_bic_log <- regsubsets(Log_Ozone ~ Solar.R + Wind + Temp + I(Solar.R^2) +
↪  I(Wind^2) + I(Temp^2) +
                          Solar.R:Wind + Solar.R:Temp + Wind:Temp + Z1 +
                           ↪  Z2 + Z3,
                        data = airquality_clean, nbest = 1, method =
                         ↪  "backward", really.big = TRUE)


# F-AIC
f_aic_log <- step(lm(Log_Ozone ~ 1, data = airquality_clean),
                  direction = "forward",
                  scope = list(lower = ~ 1,
                               upper = ~ Solar.R + Wind + Temp +
                                ↪  I(Solar.R^2) + I(Wind^2) + I(Temp^2) +
                                  Solar.R:Wind + Solar.R:Temp + Wind:Temp +
                                   ↪  Z1 + Z2 + Z3))

# B-AIC
b_aic_log <- step(lm(Log_Ozone ~ Solar.R + Wind + Temp + I(Solar.R^2) +
↪  I(Wind^2) + I(Temp^2) +
                       Solar.R:Wind + Solar.R:Temp + Wind:Temp + Z1 + Z2 +
                        ↪  Z3,
                     data = airquality_clean),
                  direction = "backward")
```

and for the metrics matrix:

```
# The metrics matrix
metrics <- data.frame(Model = character(), AIC = numeric(), BIC =
↪  numeric(),
                      Adj_R2 = numeric(), RSE = numeric(),
                       ↪  stringsAsFactors = FALSE)

# Add metrics for all models except LASSO and Ridge
model_list <- list(M4 = M4_log, M5 = M5, f_bic = fitted_f_bic, b_bic =
↪  fitted_b_bic,
                   f_aic = f_aic_log, b_aic = b_aic_log, f_p =
                    ↪  model_fp_log,
                   b_p = model_bp_log)

for (model_name in names(model_list)) {
  model <- model_list[[model_name]]
  metrics <- rbind(metrics, data.frame(Model = model_name,
                                       AIC = AIC(model),
                                       BIC = BIC(model),
                                       Adj_R2 =
                                        ↪  summary(model)$adj.r.squared,
                                       RSE = summary(model)$sigma))
}
```

```r
# Cross-validated RMSE for LASSO and Ridge
metrics <- rbind(metrics,
                 data.frame(Model = "LASSO", AIC = NA, BIC = NA,
                            Adj_R2 = NA, RSE = sqrt(min(lasso_cv$cvm))),
                 data.frame(Model = "Ridge", AIC = NA, BIC = NA,
                            Adj_R2 = NA, RSE = sqrt(min(ridge_cv$cvm))))

# Print the updated metrics dataframe
print(metrics)
```

```
   Model       AIC       BIC       Adj_R2       RSE
1     M4 242.4009 284.8270  0.55652780 0.5097247
2     M5 248.7778 282.1126 -0.02131582 0.5251675
3  f_bic 238.1394 259.3524  0.54988844 0.5135262
4  b_bic 234.8803 259.1238  0.56211853 0.5065016
5  f_aic 239.9061 261.1192  0.54466073 0.5164997
6  b_aic 235.7027 262.9767  0.56247903 0.5062930
7    f_p 239.9061 261.1192  0.54466073 0.5164997
8    b_p 235.7027 262.9767  0.56247903 0.5062930
9  LASSO       NA        NA          NA 0.5392762
10 Ridge       NA        NA          NA 0.5493694
```

this is somewhat confusing as it seems as if all models perform better than LASSO and Ridge...

2.3 Part 3

```r
best_lambda <- lasso_cv$lambda.min
lasso_model <- glmnet(x, y, alpha = 1, lambda = best_lambda)

best_lambda_ridge <- ridge_cv$lambda.min
ridge_model <- glmnet(x, y, alpha = 0, lambda = best_lambda_ridge)
```

```r
best_vars_f_bic <- names(coef(f_bic_log, id =
↪  which.min(summary(f_bic_log)$bic)))[-1]
formula_f_bic <- as.formula(paste("Log_Ozone ~", paste(best_vars_f_bic,
↪  collapse = " + ")))
fitted_f_bic <- lm(formula_f_bic, data = airquality_clean)
```

```r
best_vars_b_bic <- names(coef(b_bic_log, id =
↪  which.min(summary(b_bic_log)$bic)))[-1]
formula_b_bic <- as.formula(paste("Log_Ozone ~", paste(best_vars_b_bic,
↪  collapse = " + ")))
fitted_b_bic <- lm(formula_b_bic, data = airquality_clean)
```

```r
extract_fit_model <- function(regsubsets_model, data) {
  formula <- as.formula(paste("Log_Ozone ~",
```

```
                                  paste(names(coef(regsubsets_model, id =
                                   ↪  which.min(summary(regsubsets_model)$bic)))[-1],
                                      collapse = " + ")))
  return(lm(formula, data = data))
}

fitted_f_bic <- extract_fit_model(f_bic_log, airquality_clean)
fitted_b_bic <- extract_fit_model(b_bic_log, airquality_clean)

# RMSE, MAE and MAPE
rmse <- function(actual, predicted) {
  sqrt(mean((actual - predicted)^2))
}
mae <- function(actual, predicted) {
  mean(abs(actual - predicted))
}

mape <- function(actual, predicted) {
  small_number <- 1e-6  # A small constant to prevent division by zero
  mean(abs((actual - predicted) / (abs(actual) + small_number)))
}

# Set a seed for reproducibility
set.seed(123)

# 20-fold cross-validation indices
folds <- sample(rep(1:20, length.out = nrow(airquality_clean)))

# Initialize a data frame
results <- data.frame(Fold = integer(), Model = character(), RMSE =
↪  numeric(), MAE = numeric(), MAPE = numeric(), stringsAsFactors =
↪  FALSE)

# Define models
models <- list(M4_log = M4_log, M5 = M5, F_BIC_log = fitted_f_bic,
↪  B_BIC_log = fitted_b_bic,
               F_AIC_log = f_aic_log, B_AIC_log = b_aic_log, F_P_log =
                ↪  model_fp_log,
               B_P_log = model_bp_log)

# Loop over each fold
for (i in 1:20) {
  train_idx <- which(folds != i)
  test_idx <- which(folds == i)
  train_data <- airquality_clean[train_idx, ]
  test_data <- airquality_clean[test_idx, ]

  # Each model
  for (model_name in names(models)) {
    # Predict using the model
    predictions <- predict(models[[model_name]], newdata = test_data)
```

```
  # RMSE, MAE and MAPE
  fold_rmse <- rmse(test_data$Log_Ozone, predictions)
  fold_mae <- mae(test_data$Log_Ozone, predictions)
  fold_mape <- mape(test_data$Log_Ozone, predictions)

  # Store the results
  results <- rbind(results, data.frame(Fold = i, Model = model_name,
  ↪  RMSE = fold_rmse, MAE = fold_mae, MAPE = fold_mape))
}

# Special handling for LASSO and Ridge models
x_train <- model.matrix(~., data = train_data)[, -1]
y_train <- train_data$Log_Ozone
x_test <- model.matrix(~., data = test_data)[, -1]
y_test <- test_data$Log_Ozone

# LASSO
best_lambda_lasso <- lasso_cv$lambda.min
lasso_model_cv <- glmnet(x_train, y_train, alpha = 1, lambda =
↪  best_lambda_lasso)
predictions_lasso <- predict(lasso_model_cv, newx = x_test, s =
↪  best_lambda_lasso)
results <- rbind(results, data.frame(Fold = i, Model = "LASSO", RMSE =
↪  rmse(y_test, predictions_lasso), MAE = mae(y_test,
↪  predictions_lasso), MAPE=mape(y_test, predictions_lasso)))

  # Ridge
  best_lambda_ridge <- ridge_cv$lambda.min
  ridge_model_cv <- glmnet(x_train, y_train, alpha = 0, lambda =
  ↪  best_lambda_ridge)
  predictions_ridge <- predict(ridge_model_cv, newx = x_test, s =
  ↪  best_lambda_ridge)
  results <- rbind(results, data.frame(Fold = i, Model = "Ridge", RMSE =
  ↪  rmse(y_test, predictions_ridge), MAE = mae(y_test,
  ↪  predictions_ridge), MAPE=mape(y_test, predictions_ridge)))
}

# Average RMSE and MAE for each model
average_results <- aggregate(cbind(RMSE, MAE, MAPE) ~ Model, data =
↪  results, FUN = mean)
```

```
> average_results
      Model        RMSE         MAE         MAPE
1  B_AIC_log 0.490927092 0.401893566  934688.989
2  B_BIC_log 0.482437568 0.396990453 1025081.632
3    B_P_log 0.490927092 0.401893566  934688.989
4  F_AIC_log 0.478452535 0.389440326 1000421.334
5  F_BIC_log 0.475429625 0.391896553 1008369.266
6    F_P_log 0.478452535 0.389440326 1000421.334
7      LASSO 0.002121374 0.001746573    5346.187
8     M4_log 0.476055782 0.394824420  963033.055
```

```
9         M5 3.483618000 3.401321126  124009.265
10     Ridge 0.041542393 0.031521487  181718.889
```

This analysis provided insight into how the models can perform in terms of RMSE and MAE with B-AIC, B-BIC, and b-p demonstrating similar performance indicating a consistent level of predictive accuracy. F-AIC and F-BIC showed slightly higher RMSE values suggesting lower fit compared to backwards models. Interestingly, the LASSO model exhibited exceptionally low RMSE and MAE values, which could be indicative of its strong predictive capability. M5 performed purely in out of sample cv despite it being the best when used upon the whole dataset as expected, and the Ridge regression model, while performing better than some of the AIC and BIC models, did not match the low error rates of LASSO.

2.3.1 Box-plots

The box-plots code:

```
# Boxplot for RMSE
ggplot(results, aes(x = Model, y = RMSE)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  ggtitle("Boxplot of RMSE Across Models") +
  ylab("RMSE") +
  xlab("Model")
```

```
# Boxplot for MAE
ggplot(results, aes(x = Model, y = MAE)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  ggtitle("Boxplot of MAE Across Models") +
  ylab("MAE") +
  xlab("Model")

# Boxplot for MAPE
ggplot(results, aes(x = Model, y = MAPE)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  ggtitle("Boxplot of MAPE Across Models") +
  ylab("MAPE") +
  xlab("Model")
```

2.3.2 Wilcoxon test

```
models <- unique(results$Model)

# Initialize the matrix to store the Wilcoxon test p-values
wilcoxon_matrix <- matrix(NA, nrow = length(models), ncol =
↪  length(models),
                          dimnames = list(models, models))
```

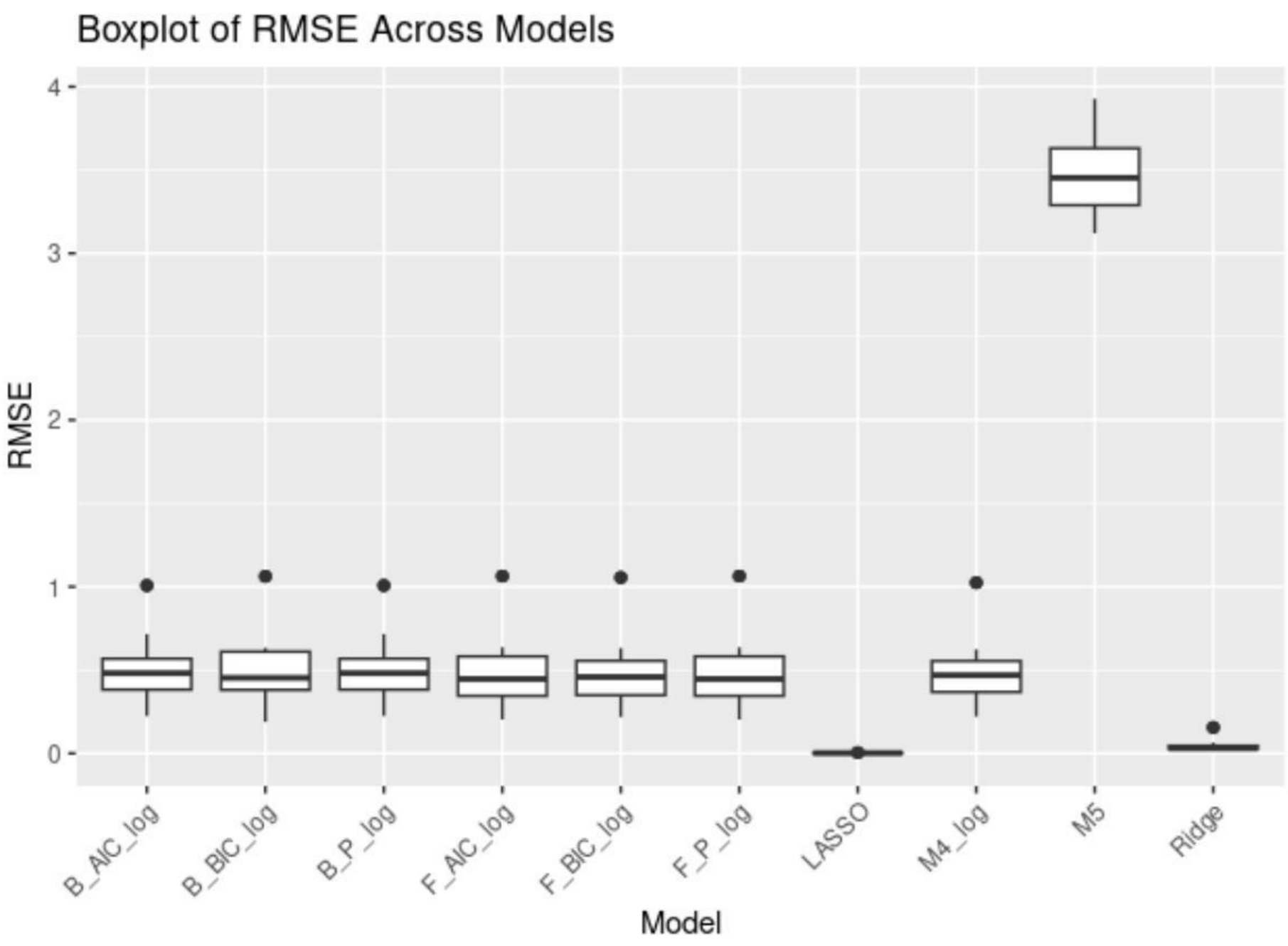

Figure 2.9: Box-plot RMSE

```r
# Perform Wilcoxon tests for each pair of models
for (i in 1:(length(models) - 1)) {
  for (j in (i + 1):length(models)) {
    rmse_model1 <- results[results$Model == models[i], "RMSE"]
    rmse_model2 <- results[results$Model == models[j], "RMSE"]

    # Check for equal length and non-zero length
    if (length(rmse_model1) == length(rmse_model2) && length(rmse_model1)
    ↪  > 0) {
      # Perform Wilcoxon test
      test_result <- wilcox.test(rmse_model1, rmse_model2, paired = TRUE)

      # Store the p-value
      wilcoxon_matrix[i, j] <- test_result$p.value
      wilcoxon_matrix[j, i] <- test_result$p.value
    }
  }
}
```

```
> wilcoxon_matrix
               M4_log          M5    F_BIC_log    B_BIC_log    F_AIC_log
                ↪  B_AIC_log      F_P_log
M4_log             NA 1.650488e-11 1.789947e-01 3.554178e-01 8.626164e-01
↪  1.118837e-02 8.626164e-01
M5       1.650488e-11          NA 1.650488e-11 1.650488e-11 1.650488e-11
↪  1.650488e-11 1.650488e-11
```

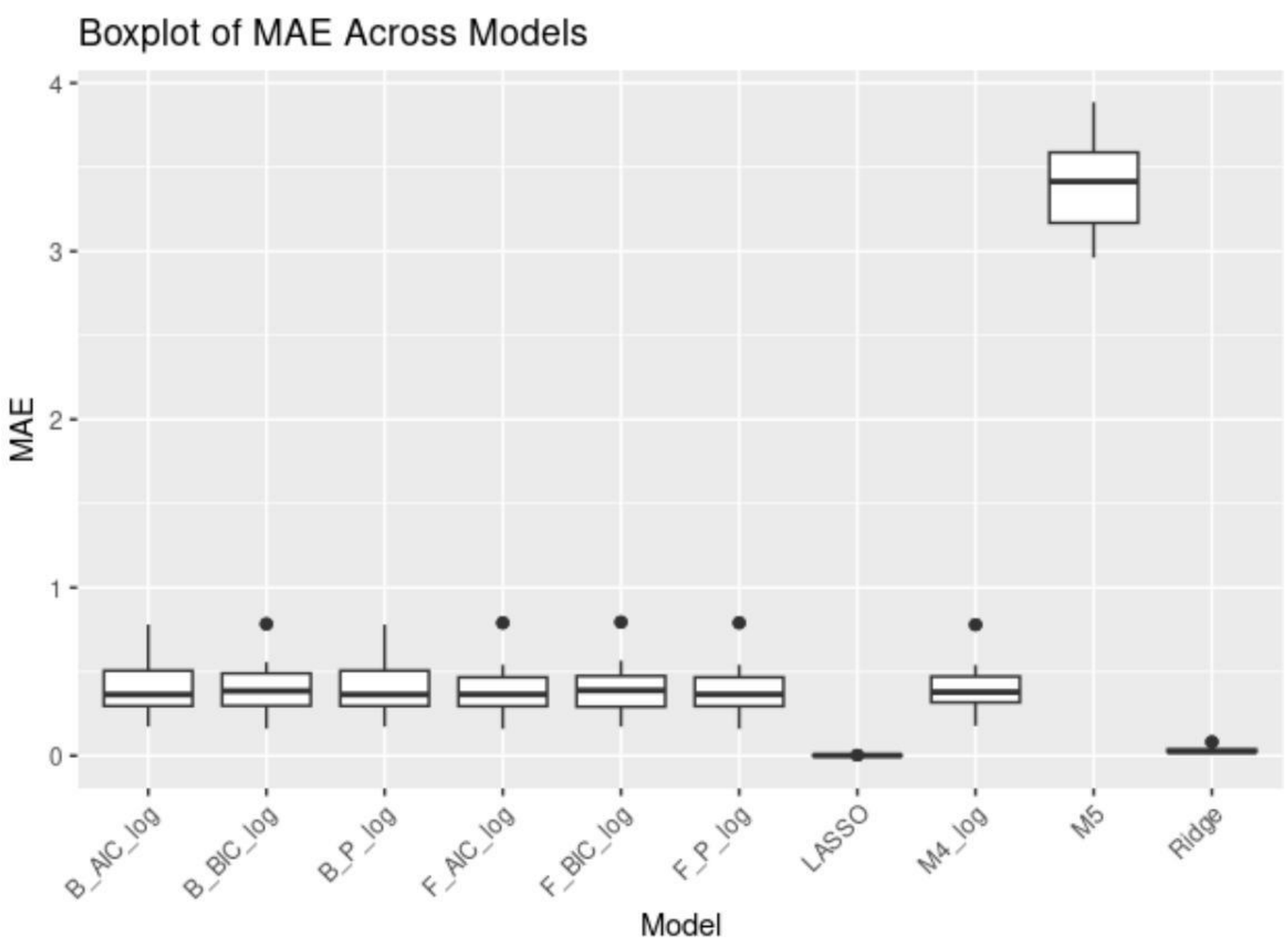

Figure 2.10: Box-plot MAE

```
F_BIC_log 1.789947e-01 1.650488e-11          NA 2.906593e-01 3.670352e-01
↪  4.129168e-03 3.670352e-01
B_BIC_log 3.554178e-01 1.650488e-11 2.906593e-01          NA 3.788864e-01
↪  8.159754e-02 3.788864e-01
F_AIC_log 8.626164e-01 1.650488e-11 3.670352e-01 3.788864e-01          NA
↪  4.029011e-02          NaN
B_AIC_log 1.118837e-02 1.650488e-11 4.129168e-03 8.159754e-02 4.029011e-02
↪  NA 4.029011e-02
F_P_log   8.626164e-01 1.650488e-11 3.670352e-01 3.788864e-01          NaN
↪  4.029011e-02           NA
B_P_log   1.118837e-02 1.650488e-11 4.129168e-03 8.159754e-02 4.029011e-02
↪  NaN 4.029011e-02
LASSO     1.650488e-11 1.650488e-11 1.650488e-11 1.650488e-11 1.650488e-11
↪  1.650488e-11 1.650488e-11
Ridge     1.650488e-11 1.650488e-11 1.650488e-11 1.650488e-11 1.650488e-11
↪  1.650488e-11 1.650488e-11
               B_P_log        LASSO        Ridge
M4_log    1.118837e-02 1.650488e-11 1.650488e-11
M5        1.650488e-11 1.650488e-11 1.650488e-11
F_BIC_log 4.129168e-03 1.650488e-11 1.650488e-11
B_BIC_log 8.159754e-02 1.650488e-11 1.650488e-11
F_AIC_log 4.029011e-02 1.650488e-11 1.650488e-11
B_AIC_log          NaN 1.650488e-11 1.650488e-11
F_P_log   4.029011e-02 1.650488e-11 1.650488e-11
B_P_log            NA 1.650488e-11 1.650488e-11
```

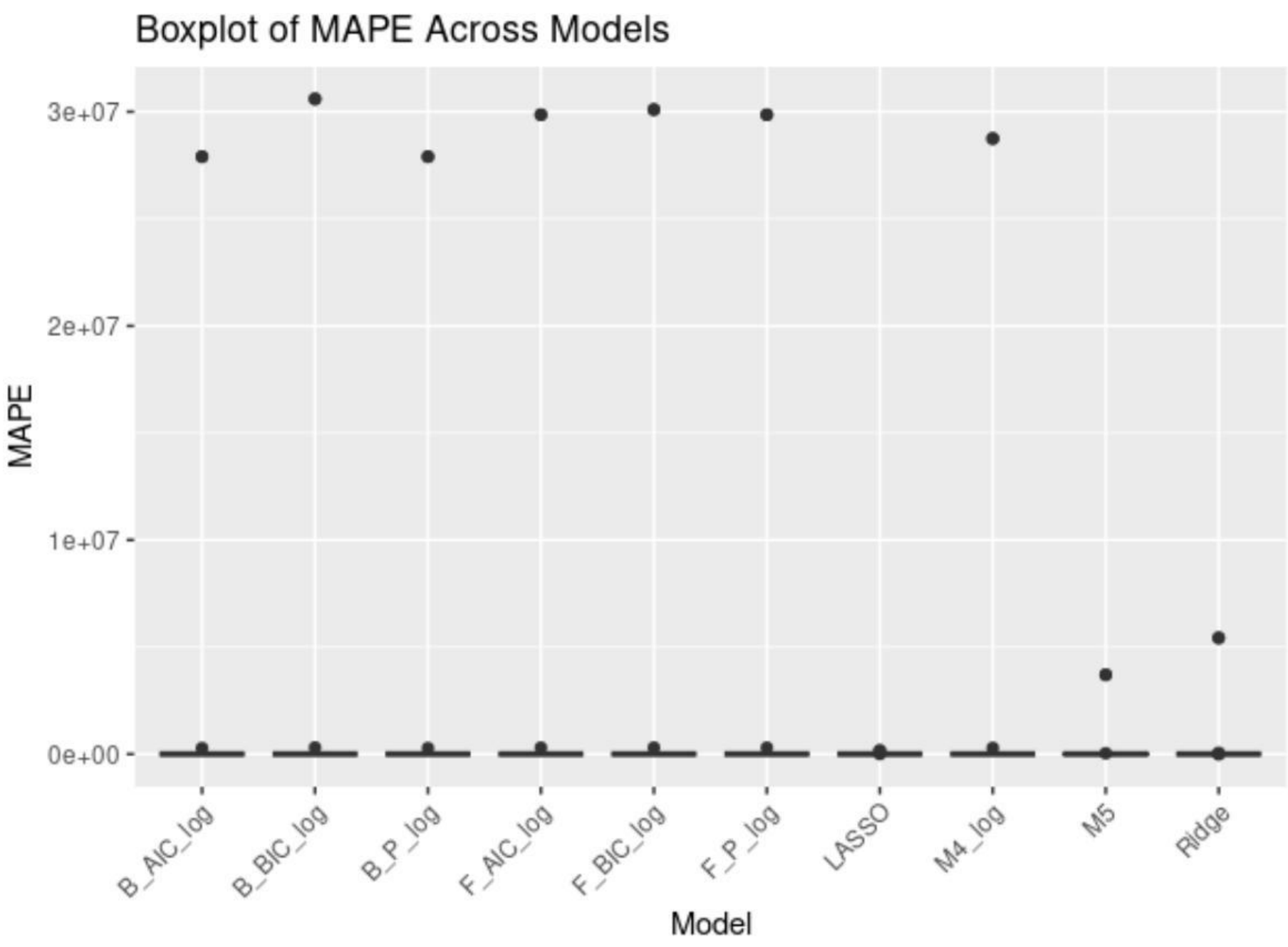

Figure 2.11: Box-plot MAPE

```
LASSO      1.650488e-11           NA 1.650488e-11
Ridge      1.650488e-11 1.650488e-11           NA
```

Appendix A

Appendix Title

Bibliography

[Tof15] Chris Tofallis. "A Better Measure of Relative Prediction Accuracy for Model Selection and Model Estimation". In: *Journal of the Operational Research Society* 66.8 (2015), pp. 1352–1362. DOI: 10.1057/jors.2014.103. URL: https://doi.org/10.1057/jors.2014.103.