Computational & Mathematical Statistics - Fall 2023
Assignment 2

Information criteria; Forward-Backward; Penalized estimation

Ioannis Maris, math1p0004

University of Crete,
Department of Mathematics & Applied Mathematics,
Department of Computer Science,
FORTH - Foundation for Research & Technology - Institute of Applied and Computational Mathematics, Institute of Computer Science.

November 2023

Assignment 2 Part 1

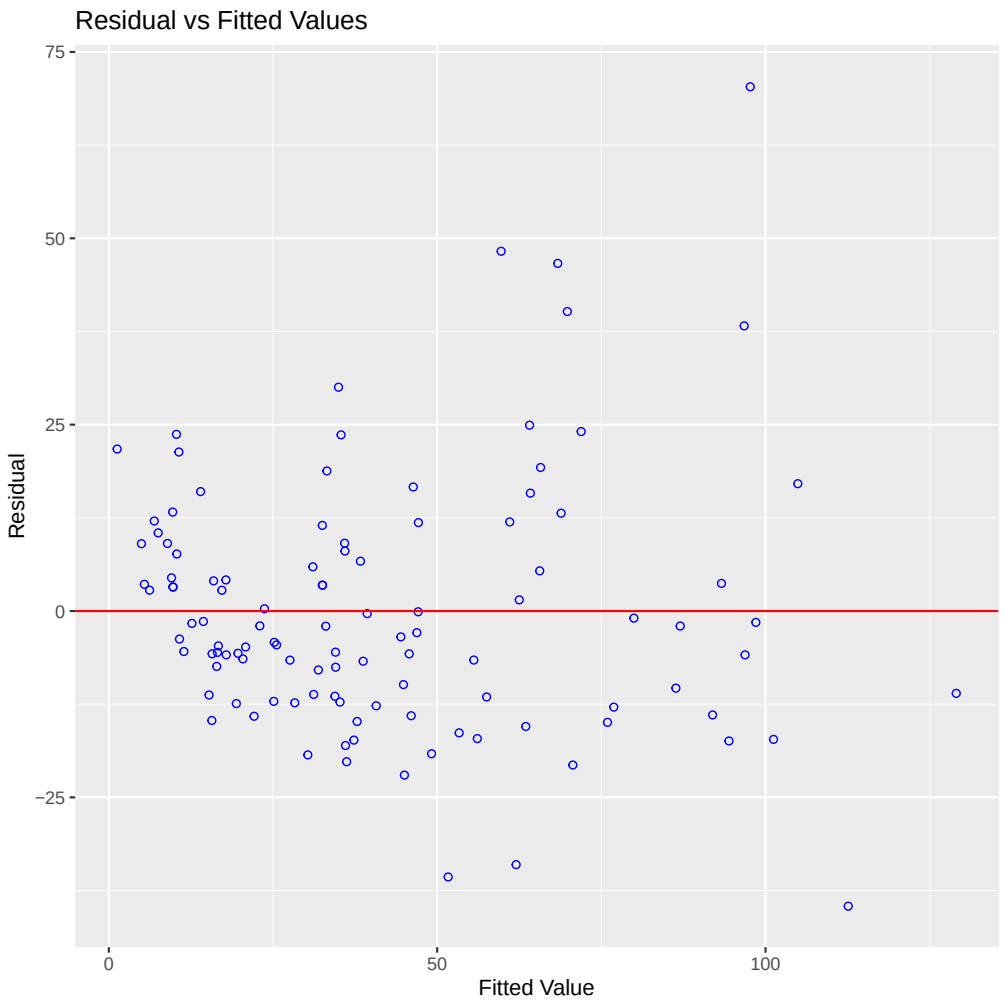
- Perform residual diagnostics on the full second order model M3 from Assignment 1.

```
library(olsrr)  # OLS diagnostics
library(L1pack) #LAD
library(caret)

data <- airquality
# Create the model with all the second and first-order terms
M3_formula <- Ozone ~ Solar.R + Wind + Temp + I(Solar.R^2)+
                      I(Wind^2) + I(Temp^2) + Solar.R:Wind+
                      Solar.R:Temp + Wind:Temp

set.seed(4)
# Remove rows with NA values
data_ <- na.omit(data)
# Shuffle the cleaned data
data_ <- data_[sample(nrow(data_)), ]
M3.ols <- lm(M3_formula, data = data_)
M3.lad <- lad(M3_formula, data = data_)
```

FORTH

Residuals vs Fitted Plot

```
ols_plot_resid_fit(M3.ols)
```

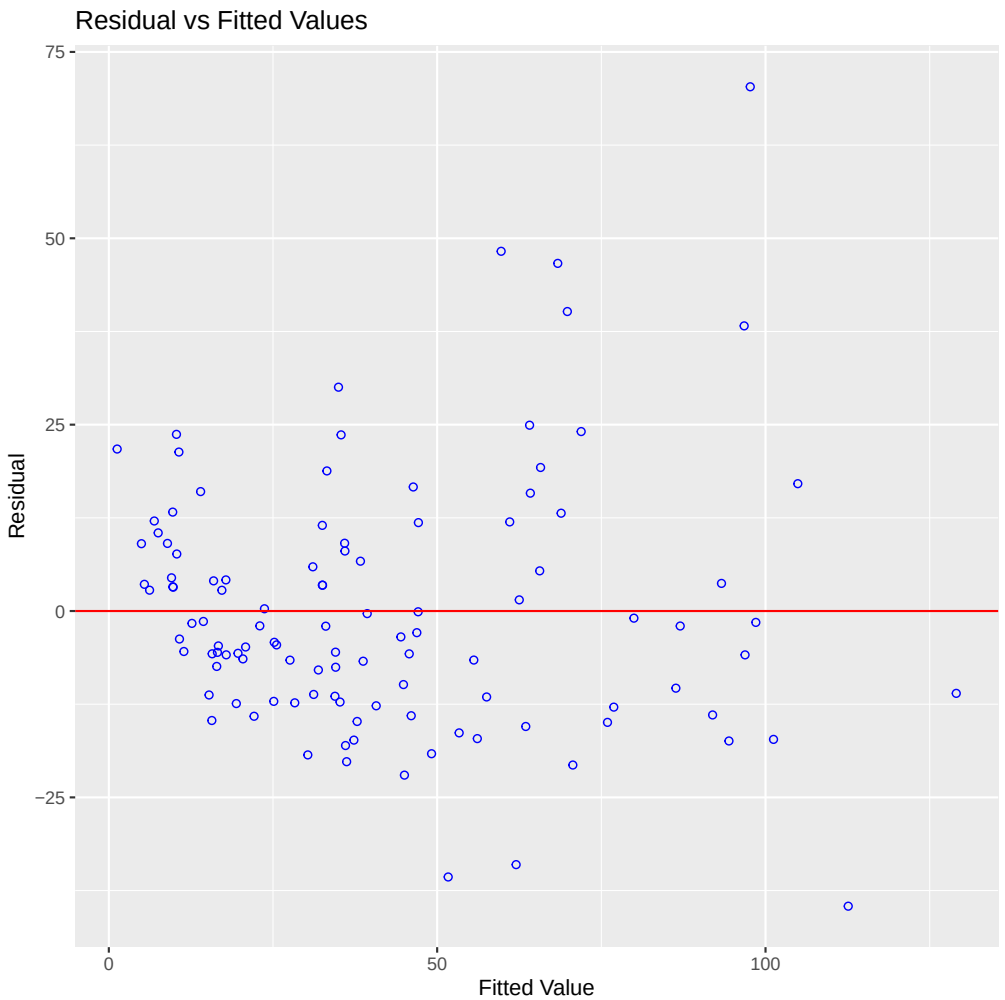


Model Diagnostics

- This plot helps assess whether the residuals have non-constant variance (heteroscedasticity).
- Scatter plot randomness around the zero line indicates the model's assumptions are likely correct, suggesting no non-linearity or misspecification.

Residuals vs Fitted Plot

```
ols_plot_resid_fit(M3.ols)
```



- In the provided plot, we would look for the absence of patterns and a roughly equal spread of residuals across the entire range of fitted values to satisfy the assumptions for the OLS.
- However, the plot seems to indicate a potential issue with **heteroscedasticity**, as indicated by a possible increasing spread in residuals as the fitted values increase.
- This would need to be further investigated.

Breusch Pagan Test for Heteroskedasticity

```
# Test for heteroscedasticity
ols_test_breusch_pagan(M3.ols)
>>
Breusch Pagan Test for
    Heteroskedasticity
 ---------------------------------
 Ho: the variance is constant
 Ha: the variance is not constant

               Data
 ---------------------------------
 Response : Ozone
 Variables: fitted values of Ozone

         Test Summary
 ---------------------------------
 DF              =    1
 Chi2            =    30.75916
 Prob > Chi2     =    2.92122e-08
```

Breusch Pagan Test[ab]

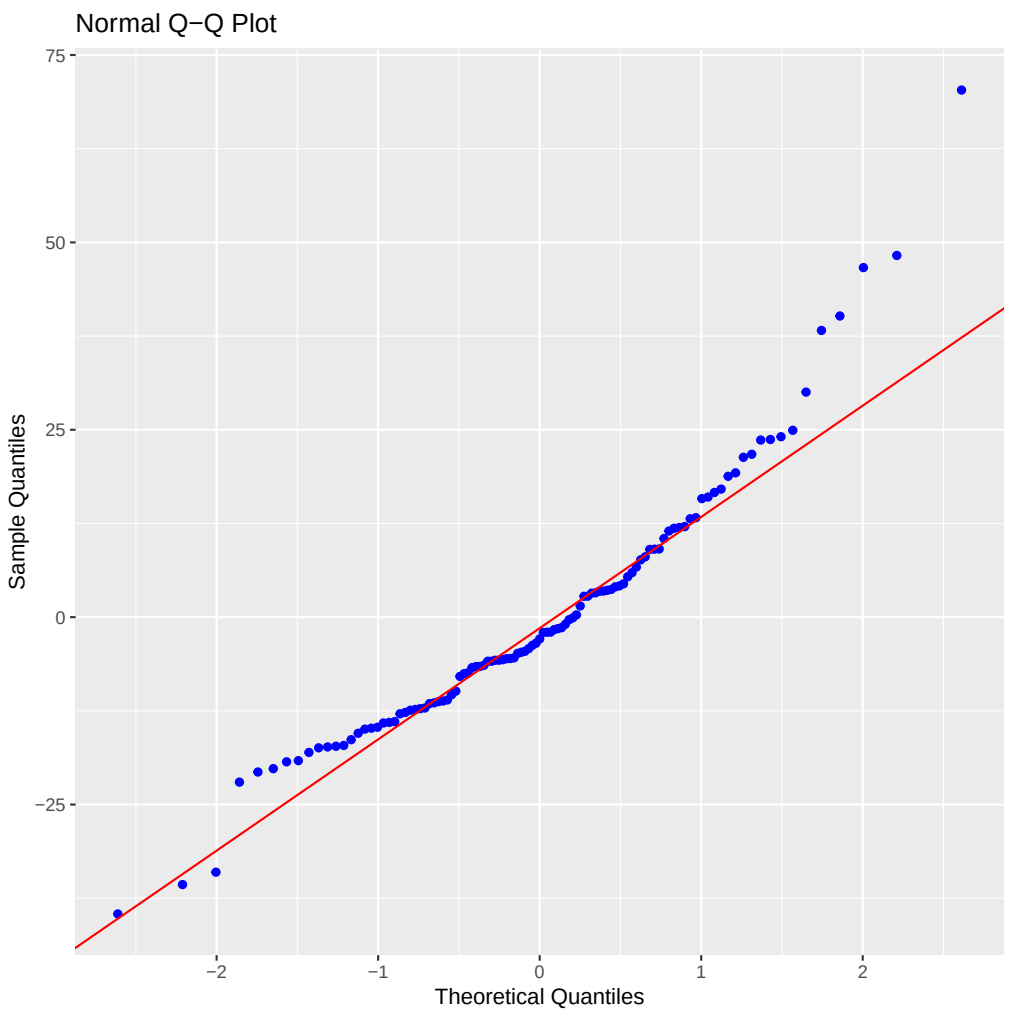
- One of the key assumptions of linear regression is that the residuals are distributed with equal variance at each level of the predictor variable. AKA **homoscedasticity**.
- When this assumption is violated, we say that **heteroscedasticity** is present in the residuals.

[a] H_0: **Homoscedasticity** is present (the residuals are distributed with equal variance)

[b] H_A: **Heteroscedasticity** is present (the residuals are not distributed with equal variance)

Normal Q-Q Plot

```
ols_plot_resid_qq(M3.ols)
```

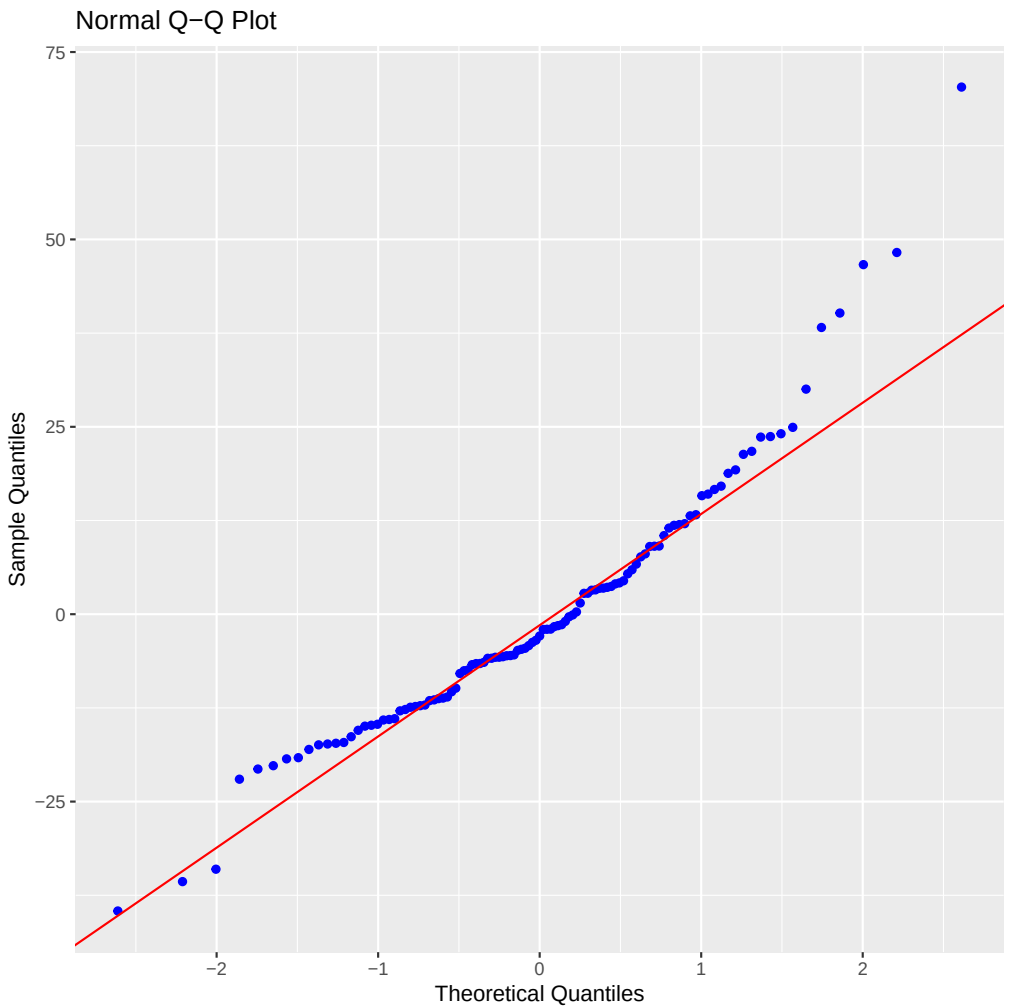


Normality Check

- This plot is used to determine if the residuals are approximately normally distributed.
- The residuals are plotted against a theoretical normal distribution in such a way that if the residuals are normally distributed, they will approximately lie on the diagonal line.

Normal Q-Q Plot

```
ols_plot_resid_qq(M3.ols)
```

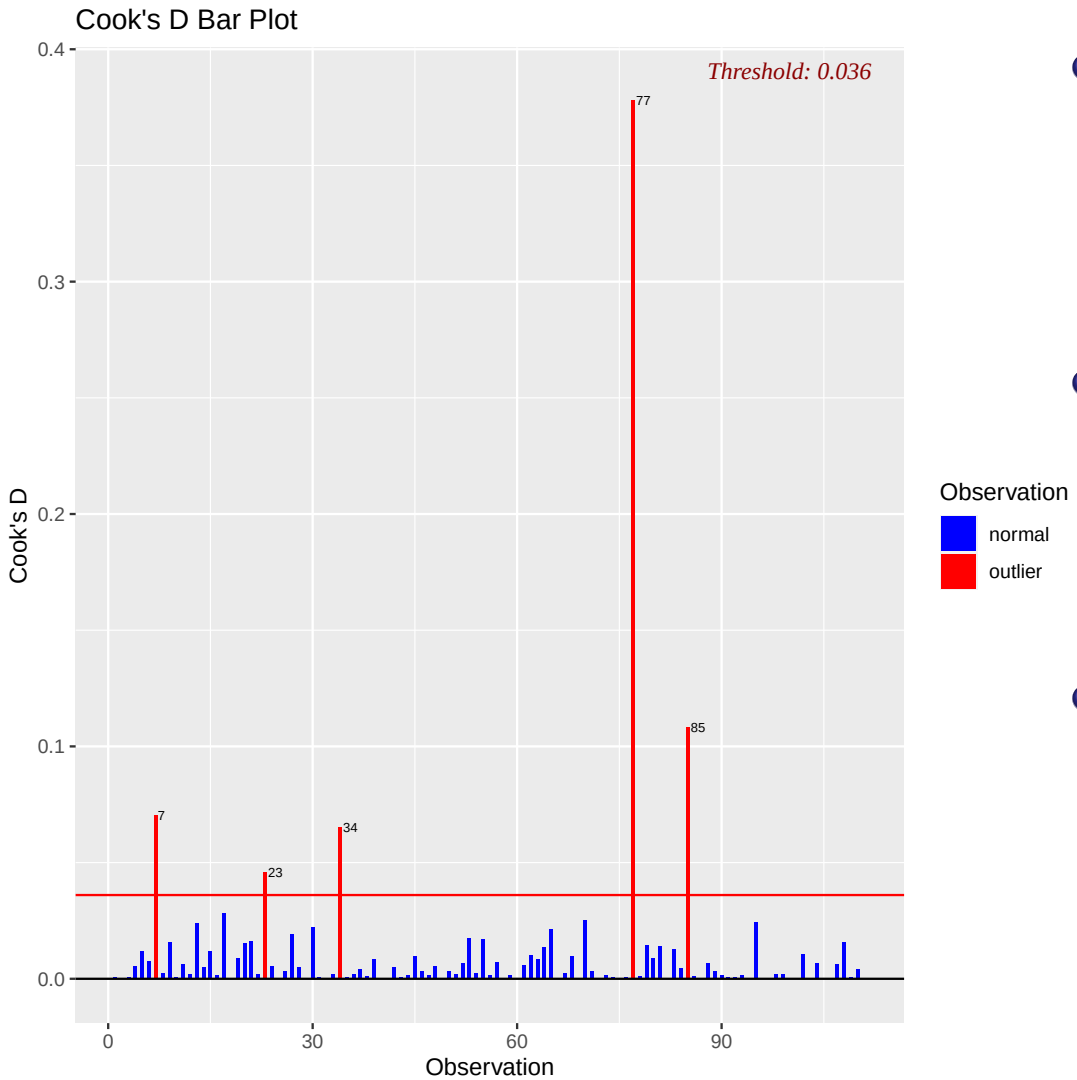


Normality Check

- The x-axis, labeled "Theoretical Quantiles", represents the quantiles from a standard normal distribution. These quantiles are the expected values if the residuals were normally distributed.
- The y-axis, labeled "Sample Quantiles", represents the quantiles of the residuals from our regression model. These are essentially the sorted values of the residuals, scaled to a normal distribution for comparison.

Cook's Distance Plot

```
ols_plot_cooksd_bar(M3.ols)
```

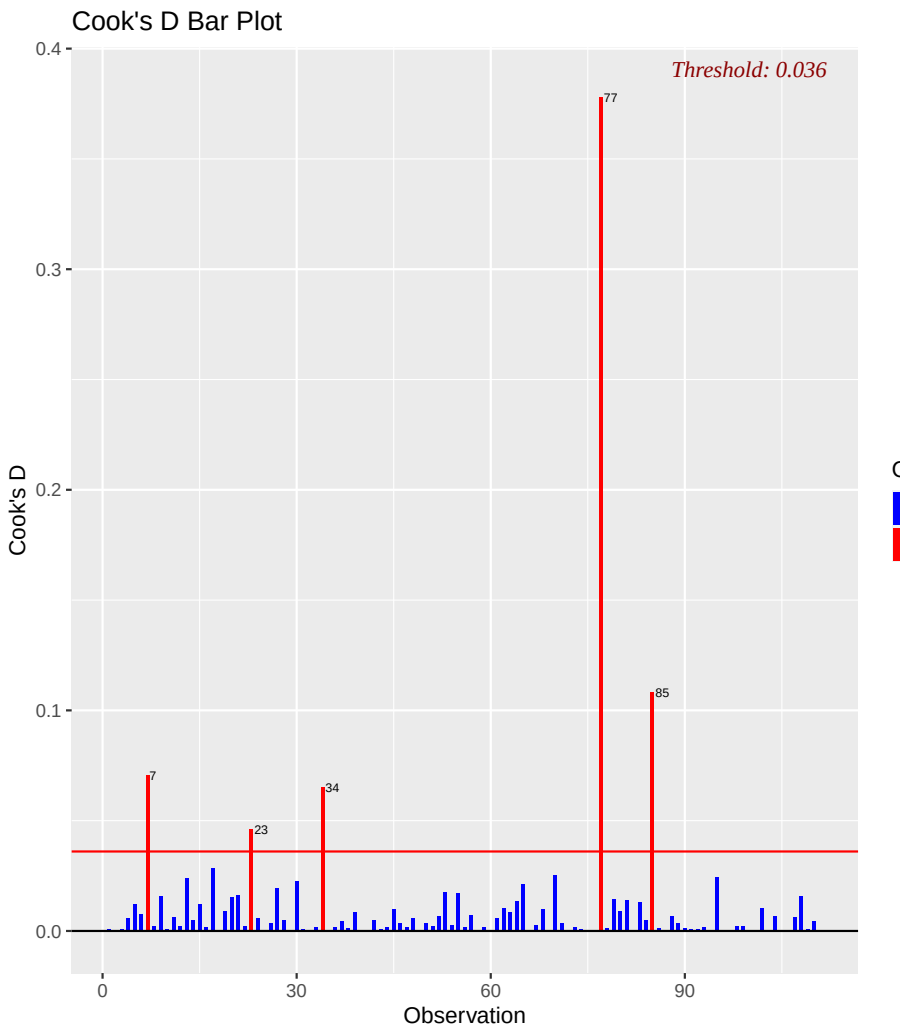


Influence Analysis

- The `'Cook's D Bar Plot'` is utilized to detect influential observations in our regression model.
- Cook's D (Cook's Distance) measures the influence of individual data points on the regression coefficients.
- The threshold of 0.036 is used to identify points that have a potentially influential impact on the model.

Cook's Distance Plot

```
ols_plot_cooksd_bar(M3.ols)
```



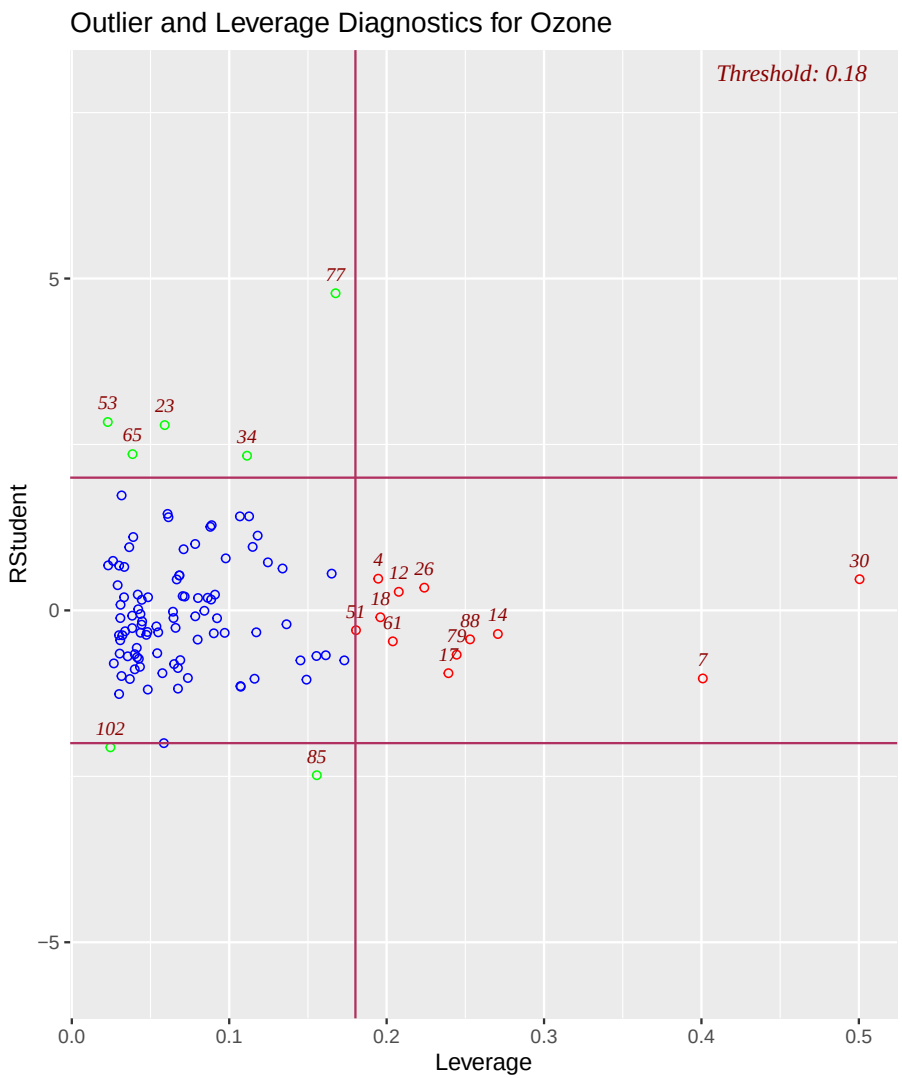
Influence Analysis

- Observation 77 is highly influential[a]a, with a Cook's D value that greatly exceeds the threshold, suggesting a significant impact on the model.

[a]The term "highly influential" in the context of Cook's D refers to the impact that the removal of that observation would have on the fitted regression model. If an observation has a high Cook's D value (beyond the threshold), it means that the estimated regression coefficients change significantly when that observation is excluded from the dataset.

Residuals vs Leverage Plot

```
ols_plot_resid_lev(M3.ols)
```



Outliers and Leverage[a]

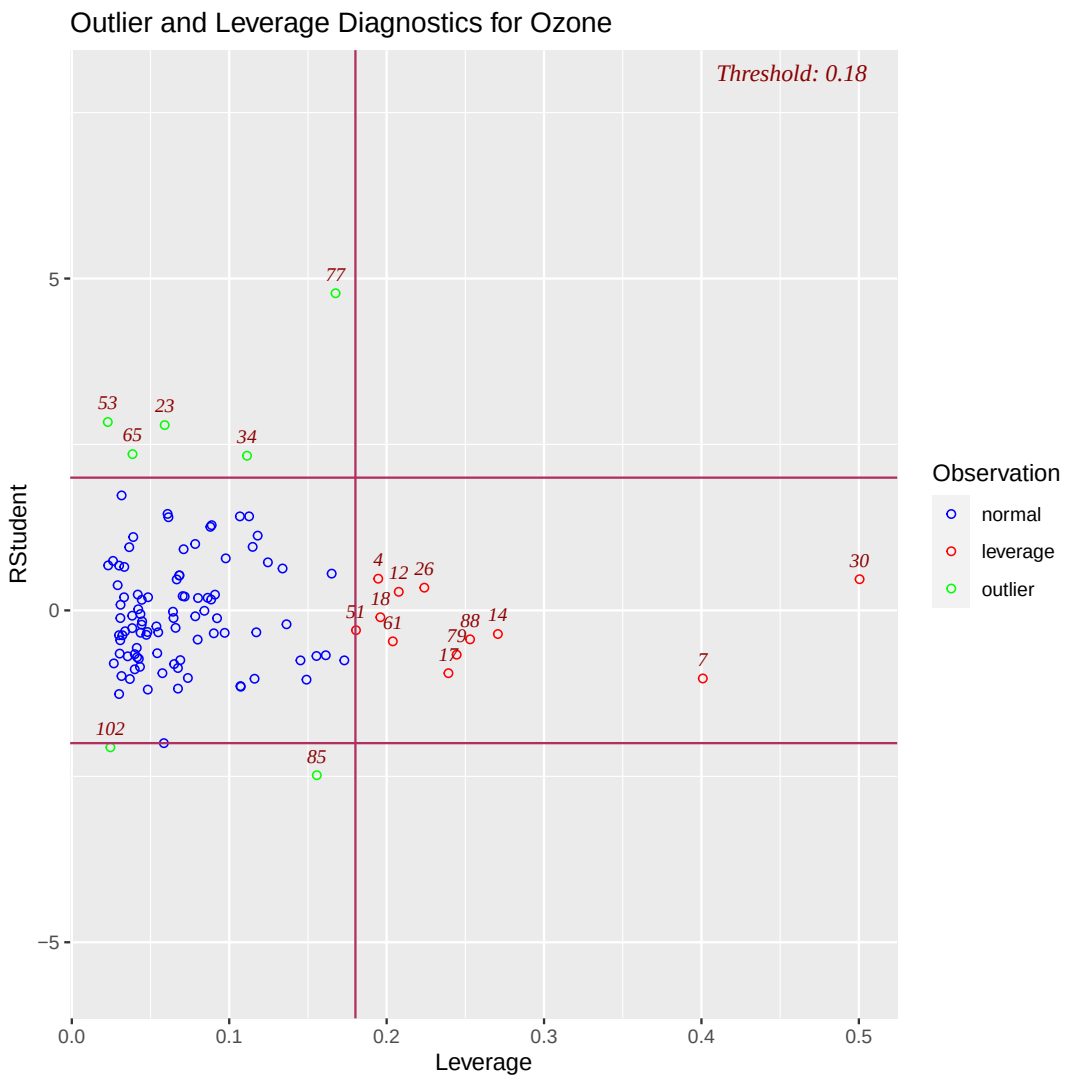
- The `'Residuals vs Leverage'` plot is a diagnostic tool used to identify influential observations.
- In general, the plot helps us understand which data points have the most influence on the calculation of the regression coefficients.

---

[a] **leverage** is a measure of how far away the independent variable values of an observation are from those of the other observations. (**Source**)

Residuals vs Leverage Plot

```
ols_plot_resid_lev(M3.ols)
```

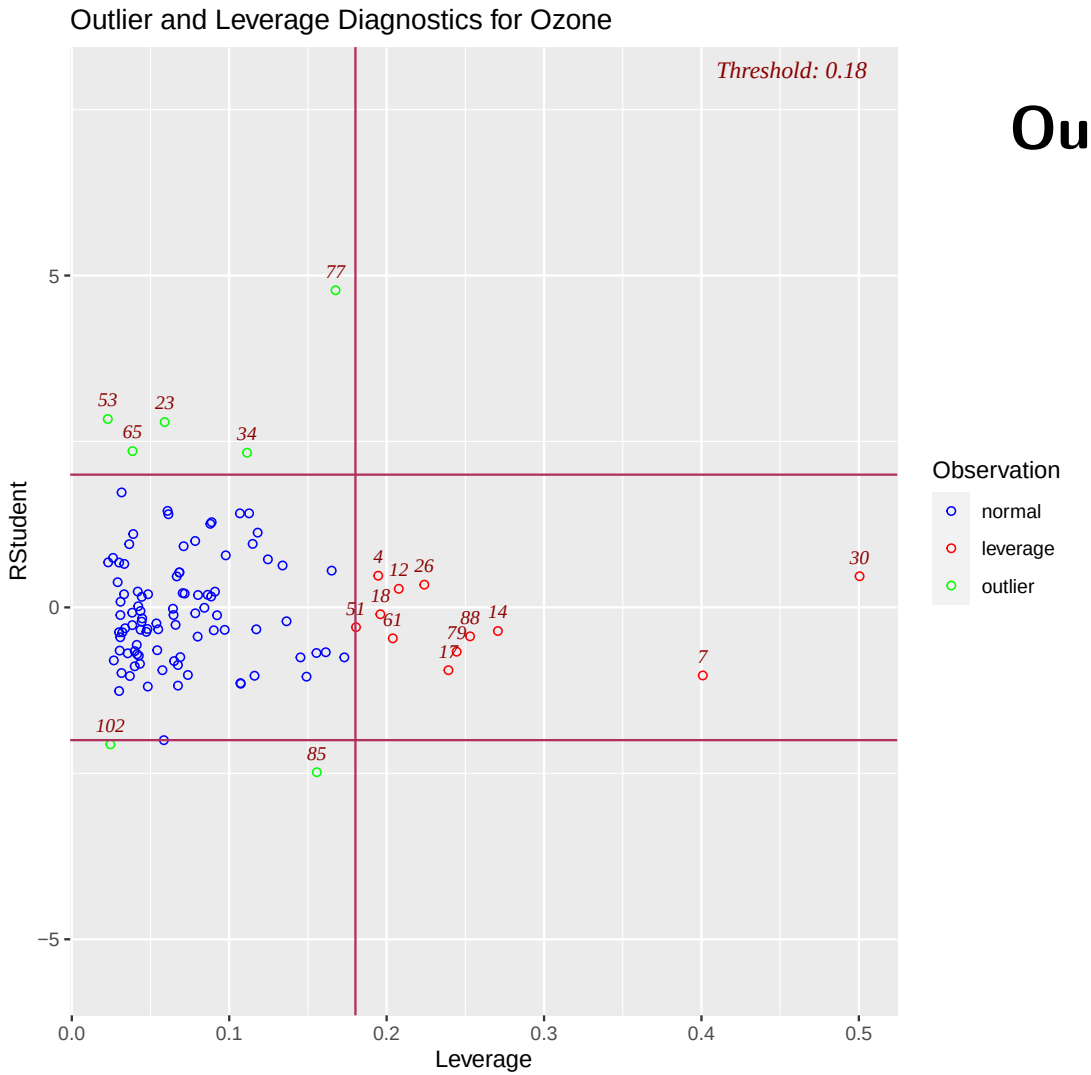


Outliers and Leverage

- The vertical red line represents a threshold for leverage, with points to the right deemed to have high leverage.
- The horizontal lines represent thresholds for standardized residuals, with points outside these lines considered to be outliers.
- The `Rstudent` axis represents studentized residuals, which are the residuals divided by their estimated standard deviation.

Residuals vs Leverage Plot

```
ols_plot_resid_lev(M3.ols)
```

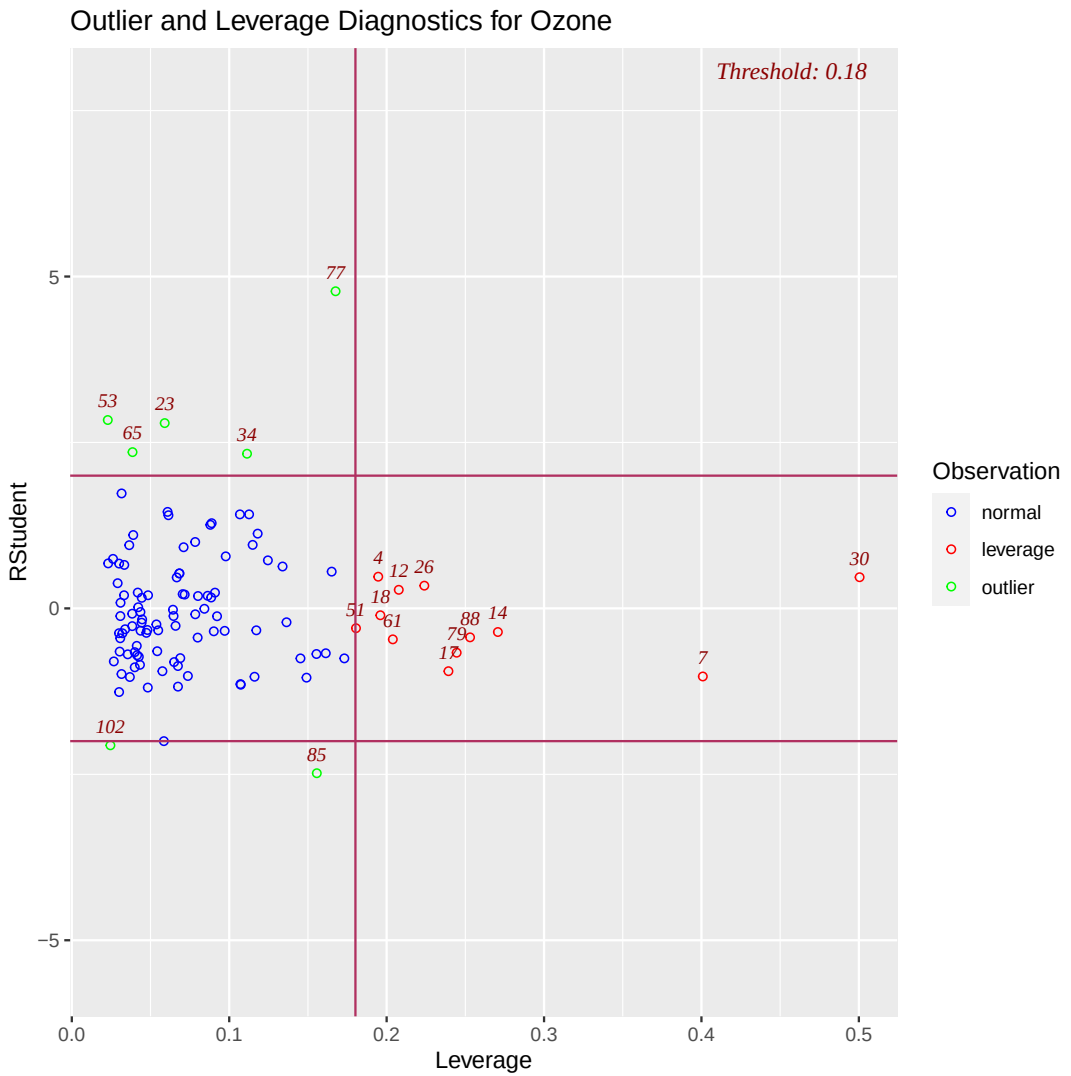


Outliers and Leverage

- The plot displays a few points that stand out:
  - Several points shown in blue are within the acceptable range for both leverage and studentized residuals, indicating they are neither high leverage points nor outliers.

Residuals vs Leverage Plot

```
ols_plot_resid_lev(M3.ols)
```

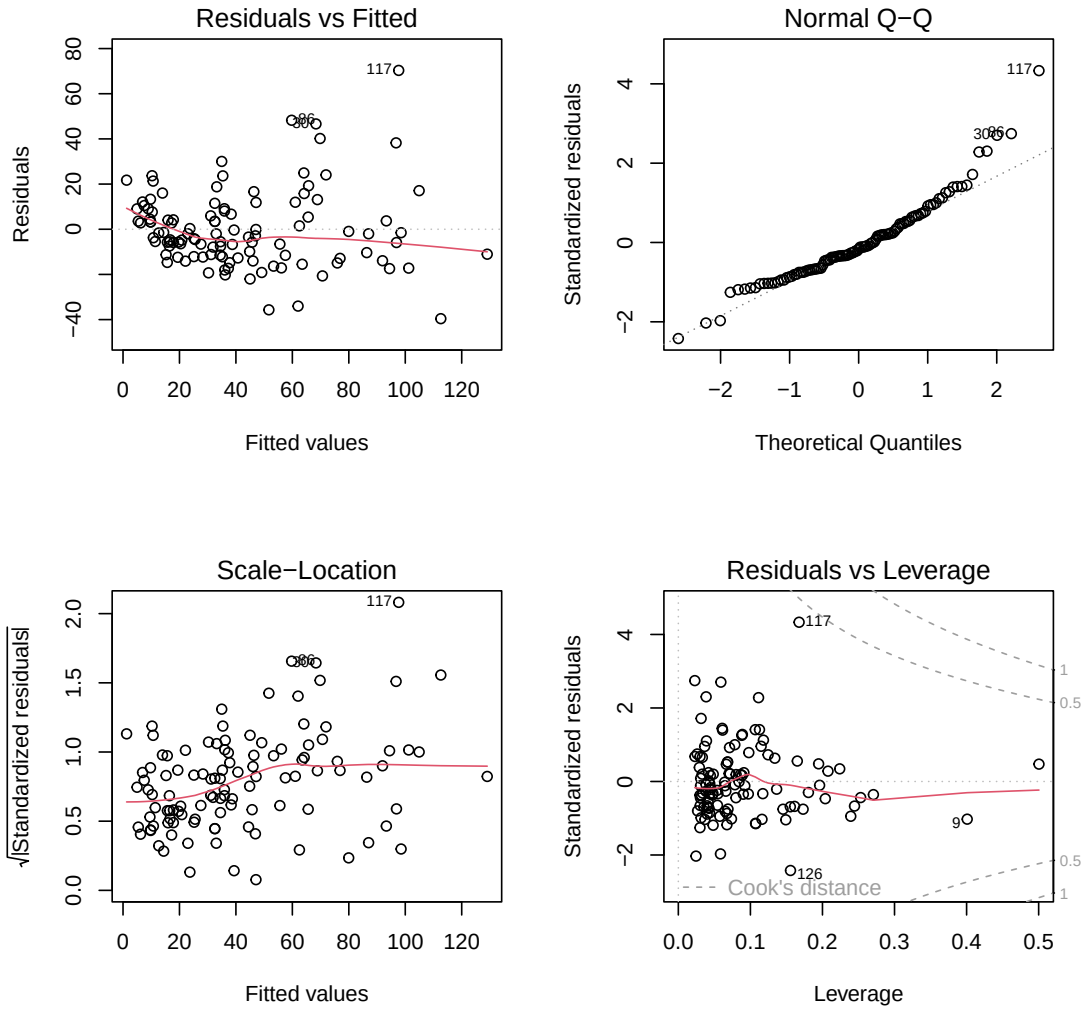


Outliers and Leverage

- Point 7 is beyond the threshold for high leverage but has a low stud. residual, indicating that while it is a leverage point, it does not greatly influence the model's predictions.
- Point 53 is an outlier in terms of the Ozone, having a high stud. residual but is not a point of high leverage, suggesting it may not be influential in terms of the leverage but is an outlier concerning the model's predictions.

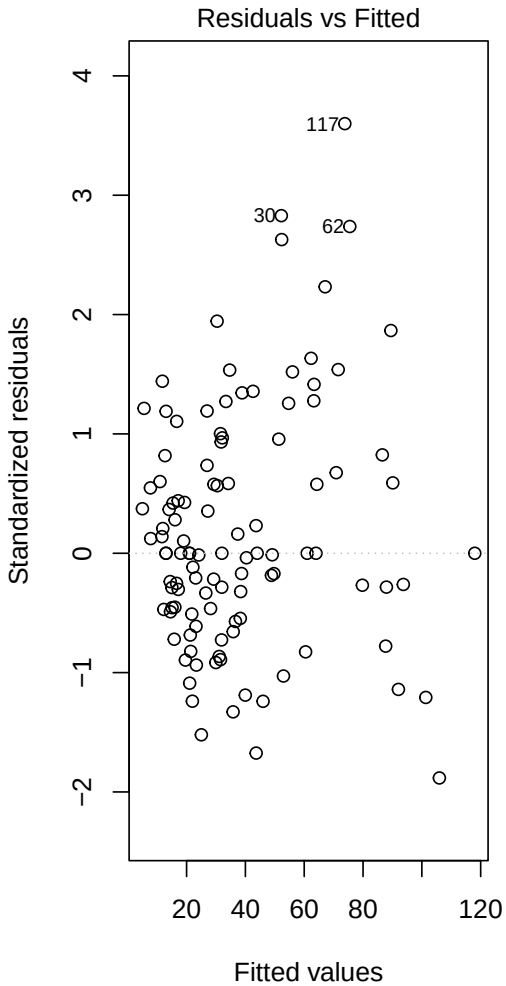
Assignment 2 Part 1 - Diagnostics with `plot()` (OLS)

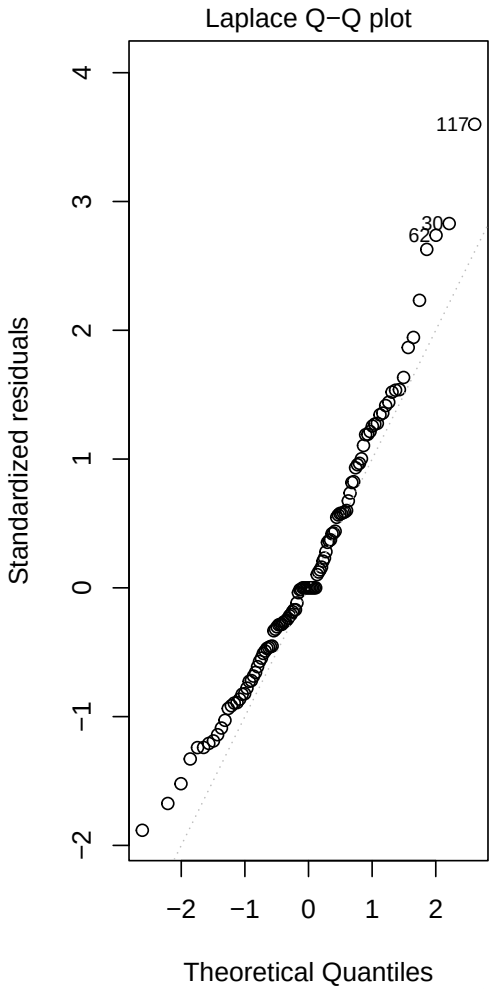
```
par(mfrow=c(2,2))
plot(M3.ols) # Alternative of olsrr
```



Assignment 2 Part 1 - Diagnostics with `plot()` (LAD)

```
par(mfrow=c(1,2))
plot(M3.lad)
```





Serial correlation

- Do not forget to evaluate whether your residuals are serially correlated.[1]

[1] Serial correlation (also known as autocorrelation) occurs when the residuals are not independent across observations, but instead one residual is correlated with previous residuals.

Serial correlation

- Do not forget to evaluate whether your residuals are serially correlated.[1]
- How can we address autocorrelation? One effective approach is to employ the **Durbin–Watson statistic**.

Durbin–Watson statistic

In statistics, the Durbin–Watson statistic is a test statistic used to detect the presence of autocorrelation. if e_t is the residual given by $e_t = \rho e_{t-1} + v_t$, the DW test statistic is

$$d = \frac{\sum_{t=2}^{T}(e_t - e_{t-1})^2}{\sum_{t=1}^{T} e_t^2},$$

where T is the number of observations. For large T, $d \approx 2(1 - \hat{\rho})$, where $\hat{\rho}$ is the sample autocorrelation of the residuals. (Source)

[1] Serial correlation (also known as autocorrelation) occurs when the residuals are not independent across observations, but instead one residual is correlated with previous residuals.

Serial correlation (autocorrelation)

```
# Package for the Durbin-Watson test
library(lmtest)

dw_test_result <- dwtest(M3.ols)
print(dw_test_result)
>>
    Durbin-Watson test

data:  M3.ols
DW = 1.8004, p-value = 0.1088
alternative hypothesis: true autocorrelation is greater than 0
```

- A value of approximately 2 indicating no autocorrelation[2].
- A value of less than 2 suggesting positive autocorrelation.
- A value of more than 2 suggesting negative autocorrelation.

1.8 is close to 2. To obtain a more accurate estimation for the statistic, we can utilize **bootstrapping**.

[2] A rule of thumb is that DW test statistic values in the range of 1.5 to 2.5 are relatively normal. (Source)

Bootstrapping serial correlation (autocorrelation)

```
set.seed(42)
B <- 10000
bagging_preds <- matrix(NA, nrow=nrow(data_), ncol=B)
boot.results <- data.frame(dw_statistic=numeric(B),
                           p_value=numeric(B))
# Bagging loop
for (b in 1:B) {
# Draw a bootstrap sample
    bootstrap_indices <- sample(1:nrow(data_),
                                size=nrow(data_),
                                replace=TRUE)
    bootstrap_data <- data_[bootstrap_indices, ]
    bootstrap_model <- lm(M3_formula, data = bootstrap_data)

    #Perform Durbin-Watson test for each bootstrap sample
    dw_test_result <- dwtest(bootstrap_model)

    boot.results$dw_statistic[b] <- dw_test_result$statistic
    boot.results$p_value[b] <- dw_test_result$p.value
    }
```

FORTH

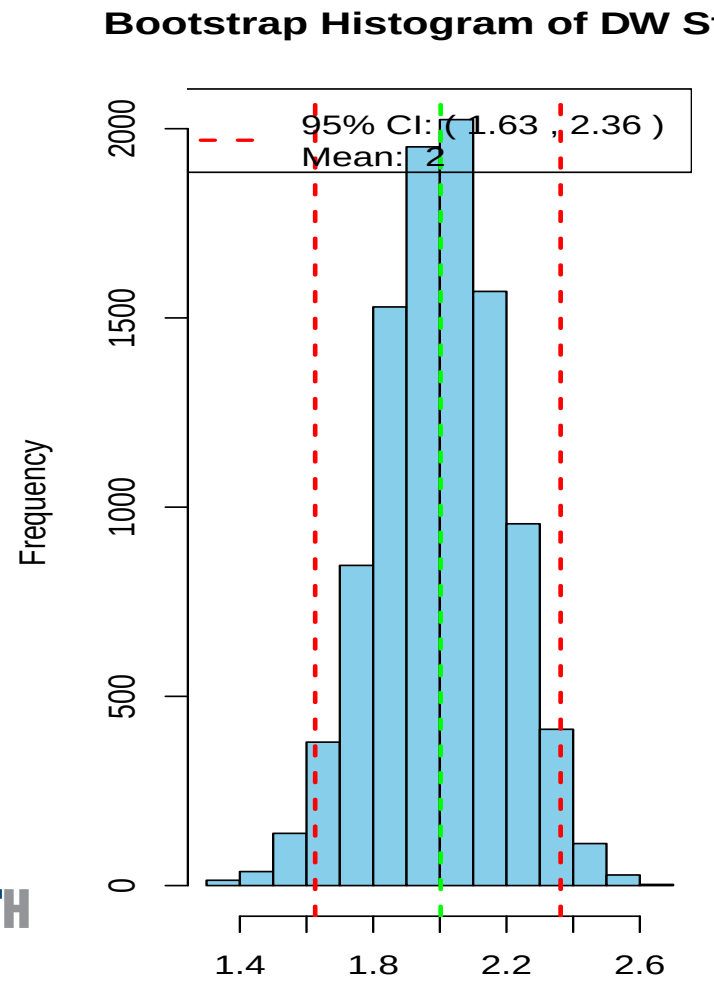
Bootstrapping serial correlation (autocorrelation)

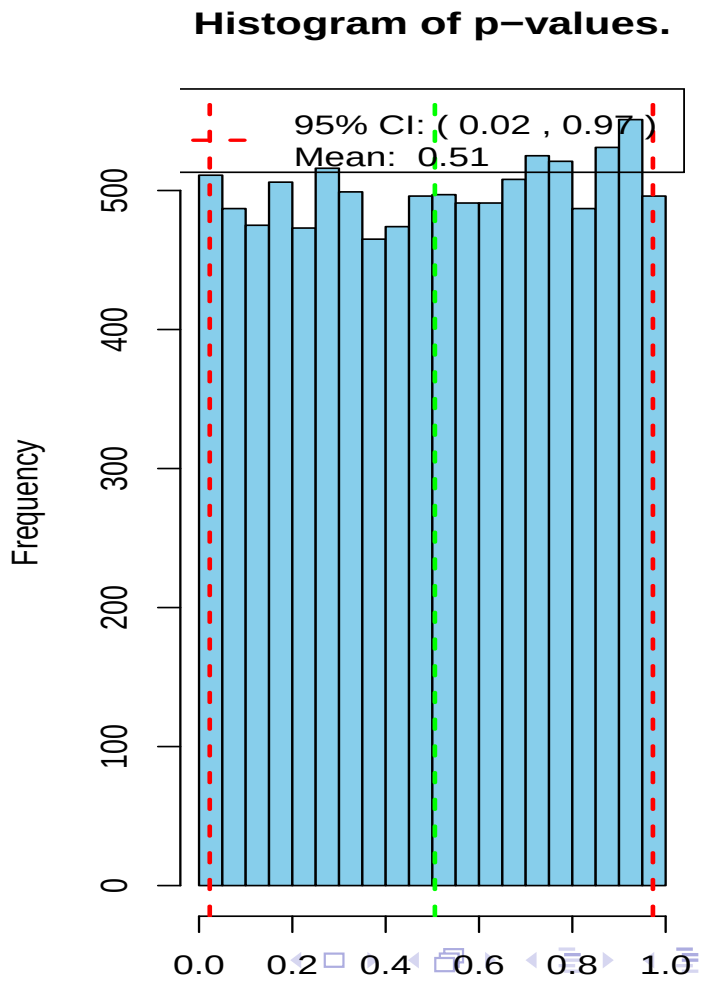
```r
# Create histogram with 95% CI and mean value lines
boot.histogram <- function(data_vector, main_title, xlab_title) {
  # Histogram of the data
  hist(data_vector,  main = main_title,
       xlab = xlab_title,  col = "skyblue", border = "black")
  # Calculate the 95% confidence interval
  ci <- quantile(data_vector, probs = c(0.025, 0.975))
  mean_val <- mean(data_vector)
  # Add dotted vertical lines for the 95% confidence interval
  abline(v = ci[1], col = "red", lwd = 2, lty = 2) # lower quantile
  abline(v = ci[2], col = "red", lwd = 2, lty = 2) # upper quantile
  # Add a dashed green line for the mean
  abline(v = mean_val, col = "green", lwd = 2, lty = 2)
  # Add a legend with the confidence interval values and mean value
  legend("topright",
         legend = paste("95% CI: (", round(ci[1], 2), ',',
                        round(ci[2], 2), ')', "\nMean: ",
                            round(mean_val, 2)),
         lty = 2, lwd = 2, col = c("red", "green"))
}
```

FORTH

Bootstrapping serial correlation (autocorrelation)

```
par(mfrow=c(1,2))
boot.histogram(boot.results$dw_statistic,
        'Bootstrap Histogram of DW Stat.', 'Durbin-Watson Statistic')
boot.histogram(boot.results$p_value, 'Histogram of DW Stat.',
               'Durbin-Watson Statistic')
```





Bootstrapping serial correlation (autocorrelation)

- **Observations:**
  - The 95% CI lies between $\underbrace{(1.5, 2.5)}_{\supset (1.63, 2.36)}$ (with a mean value of ≈ 2.)
  - Durbin–Watson Statistic approaches a normal distribution.
  - Durbin–Watson p-values are uniformly distributed[3].
- Hence, from our bootstrap analysis, we have substantial evidence to suggest that **serial correlation** does **not** present a concern in our dataset.

[3] Under the $\mathbf{H_0}$:
$\Pr(P < p) = \Pr(F(T) < p) = \Pr(T < F^{-1}(p)) = F(F^{-1}(p)) = p$. (Source)

Bootstrapping serial correlation (autocorrelation)

- **Observations:**
  - The 95% CI lies between $\underbrace{(1.5, 2.5)}_{\supset (1.63, 2.36)}$ (with a mean value of ≈ 2.)
  - Durbin–Watson Statistic approaches a normal distribution.
  - Durbin–Watson p-values are uniformly distributed[3].
- Hence, from our bootstrap analysis, we have substantial evidence to suggest that **serial correlation** does **not** present a concern in our dataset.
- ◇ How far is your analysis from the hypotheses of the Gauss-Markov Theorem?

[3] Under the $\mathbf{H_0}$:
$\Pr(P < p) = \Pr(F(T) < p) = \Pr(T < F^{-1}(p)) = F(F^{-1}(p)) = p$. (Source)

Gauss-Markov Theorem

Theorem

*The **Gauss-Markov Theorem** is a fundamental theorem in linear regression that provides the conditions under which the OLS estimator is the Best Linear Unbiased Estimator (**BLUE**). In other words, under certain conditions, OLS provides the most precise (lowest variance) estimates of the regression coefficients without bias.*

- For the OLS estimates to be BLUE, the following assumptions must be satisfied:
  1. **Linearity**: The relationship between the dependent variable and the independent variables is linear in the parameters.
  2. **Random** Sampling: The data is a random sample from the population.
  3. No Perfect **Multicollinearity**: The independent variables are not perfectly correlated with each other.
  4. **Homoscedasticity**: The variance of the residuals (errors) is constant across all levels of the independent variables.
  5. **No Autocorrelation**: The residuals (errors) are not correlated with each other (no serial correlation).
  6. **Zero Conditional Mean**: the errors are uncorrelated with the features

Linearity - Pairplot via GGally, ggplot2

```
library(ggplot2)
library(GGally)
# Add the second order terms to data_ (frame)
data_$Solar.R2 <- data_$Solar.R^2
data_$Wind2 <- data_$Wind^2
data_$Temp2 <- data_$Temp^2
data_$Solar.R_Wind <- data_$Solar.R * data_$Wind
data_$Solar.R_Temp <- data_$Solar.R * data_$Temp
data_$Wind_Temp <- data_$Wind * data_$Temp

ggpairs(data_[c(-5,-6)],
  upper = list(continuous = wrap("cor", size=3.2, color="darkblue")),
  lower = list(continuous = wrap("smooth", colour="pink",
                                 fill="lightblue", alpha=0.9)),
  diag = list(continuous = wrap("densityDiag", fill="cornflowerblue",
                                 alpha=0.85)),
        axisLabels = "show") +
  theme_bw() +
  theme(legend.position = "bottom", axis.text=element_text(size=12),
        plot.title = element_text(size=10, hjust=0.5))
```

Linearity - Pairplot via GGally, ggplot2

Gauss-Markov assumptions

1. **Linearity**: We can observe some linearity from the pairplot.
2. **Random Sampling**: We observed that in the scatter plots of the residuals versus predicted values, the residuals appear to be randomly scattered, suggesting random sampling.
3. **No** Perfect **Multicollinearity**: Some of the features are highly (Pearson) correlated with each other, as we can observe from the pairplot. For instance, `Solar.R:Temp` with $\texttt{I(Solar.R)}^2$ and so forth. This might raise concerns regarding multicollinearity issues.
4. **Homoscedasticity**: The **Breusch-Pagan** test yields a p-value that is very close to zero. Consequently, we can reject the null hypothesis, indicating the presence of heteroscedasticity.
5. **No Autocorrelation**: We have substantial evidence to suggest that the residuals are not serially correlated. (**DW statistic**≈ 2)
6. **Zero Cond. Mean** ($\mathbb{E}(\mu \mid \mathbf{X}) = 0$): We don't observe any patterns in the Residuals-Fitted values plot, thus indicating that this assumption is satisfied.

Box-Cox transformation

- Evaluate the suggested Box-Cox transformation of the response variable.

Box-Cox transformation

- Evaluate the suggested Box-Cox transformation of the response variable.

Definition

The **Box-Cox transformation** is defined as follows for a response variable Y and a transformation parameter λ:

$$Y(\lambda) = \begin{cases} \frac{Y^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0, \\ \log(Y) & \text{if } \lambda = 0. \end{cases}$$

- Y is the response variable that we want to transform.
- λ is the transformation parameter. The value of λ is chosen to achieve the best approximation to normality for the transformed variable $Y(\lambda)$.
- When $\lambda = 1$, the transformation is the identity transformation (no change).
- When $\lambda = 0$, it becomes a logarithmic transformation.
- The choice of λ can significantly impact the shape of the distribution of the transformed variable.

Purpose of Box-Cox transformation

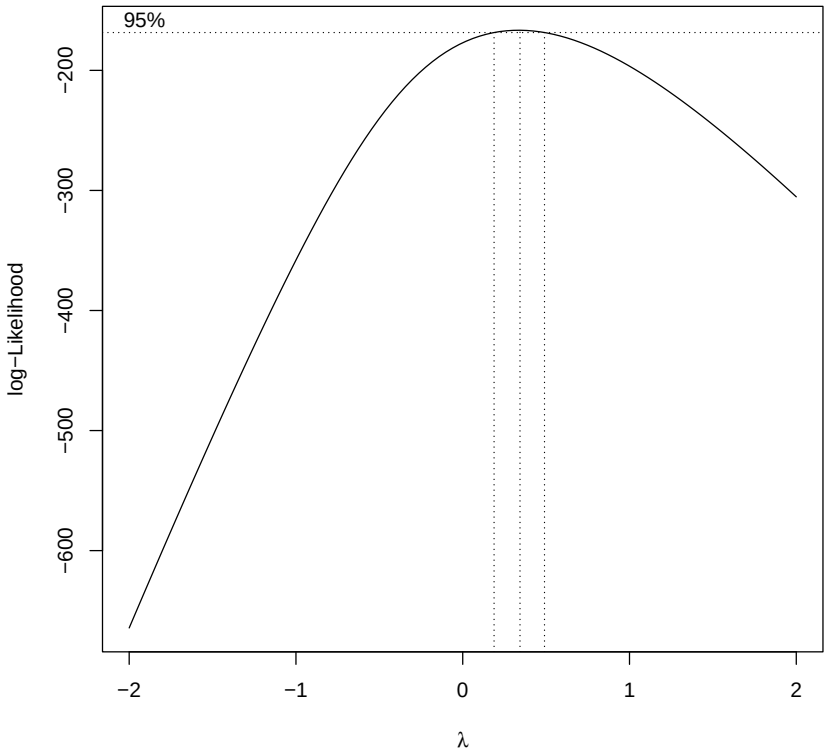
1. **Improving Normality**: It's often used when residuals from a linear model do not appear to be normally distributed. Transforming the response variable can lead to residuals that are more normally distributed.
2. **Stabilizing Variance**: It can be used when the variance of residuals is not constant across levels of an independent variable (heteroscedasticity). A Box-Cox transformation can help in stabilizing the variances.
3. **Enhancing Model Fit**: By transforming the response variable, it can improve the linearity of the relationship between independent and dependent variables in a regression model.

The optimal value of λ is usually found empirically. Choose the one that maximizes the log-likelihood.[4]

[4] The Box-Cox transformation is a powerful tool, particularly useful when dealing with non-normal data distributions and aiming to meet the assumptions of linear regression.

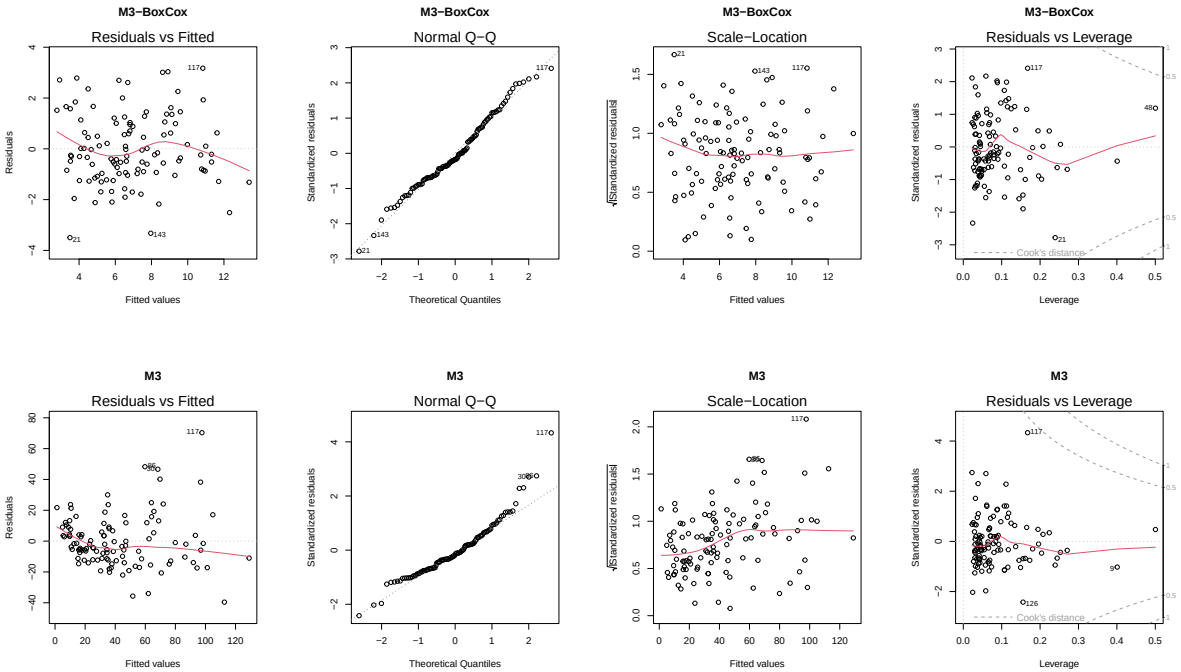
Box-Cox transformation in R

```
library(MASS)
set.seed(4)
data_ <- na.omit(data) #redef the data
data_ <- data_[sample(nrow(data_)), ]
boxcox.result <- boxcox(M3.ols)
boxcox(M3.ols)
```



Box-Cox transformation in R

```
lambda.best <- boxcox.result$x[which.max(boxcox.result$y)]
y_boxcox <- ((data_$Ozone^lambda.best) - 1)/lambda.best

M3_formula_ <- y_boxcox ~ Solar.R + Wind + Temp + I(Solar.R^2) +
  I(Wind^2) + I(Temp^2) + Solar.R:Wind +
  Solar.R:Temp + Wind:Temp
M3.ols_boxcox <- lm(M3_formula_, data = data_)
par(mfrow=c(2,4))
plot(M3.ols_boxcox, main="M3-BoxCox")
plot(M3.ols, main="M3")
```



Additional predictors

- Create 3 additional predictors Z1, Z2 and Z3. Z1 should be correlated with Solar.R with rho approximately equal to 0.8, Z2 with Wind and Z3 with Temp (same rho for all noise predictors). Include the 3 noise predictors in your model (model M4).

Additional predictors

- Create 3 additional predictors Z1, Z2 and Z3. Z1 should be correlated with Solar.R with rho approximately equal to 0.8, Z2 with Wind and Z3 with Temp (same rho for all noise predictors). Include the 3 noise predictors in your model (model M4).

Adjusted Approach for Creating Noisy Predictors:

1. To achieve a specific correlation, we can use a linear combination of the original variable and independent random noise.
2. The formula for creating a new variable Z with a desired correlation ρ with an original variable X is

$$Z = \rho \times X + \sqrt{1 - \rho^2} \times \texttt{Noise},$$

where `Noise` is randomly generated and independent of X.

Implementing the Revised Approach:

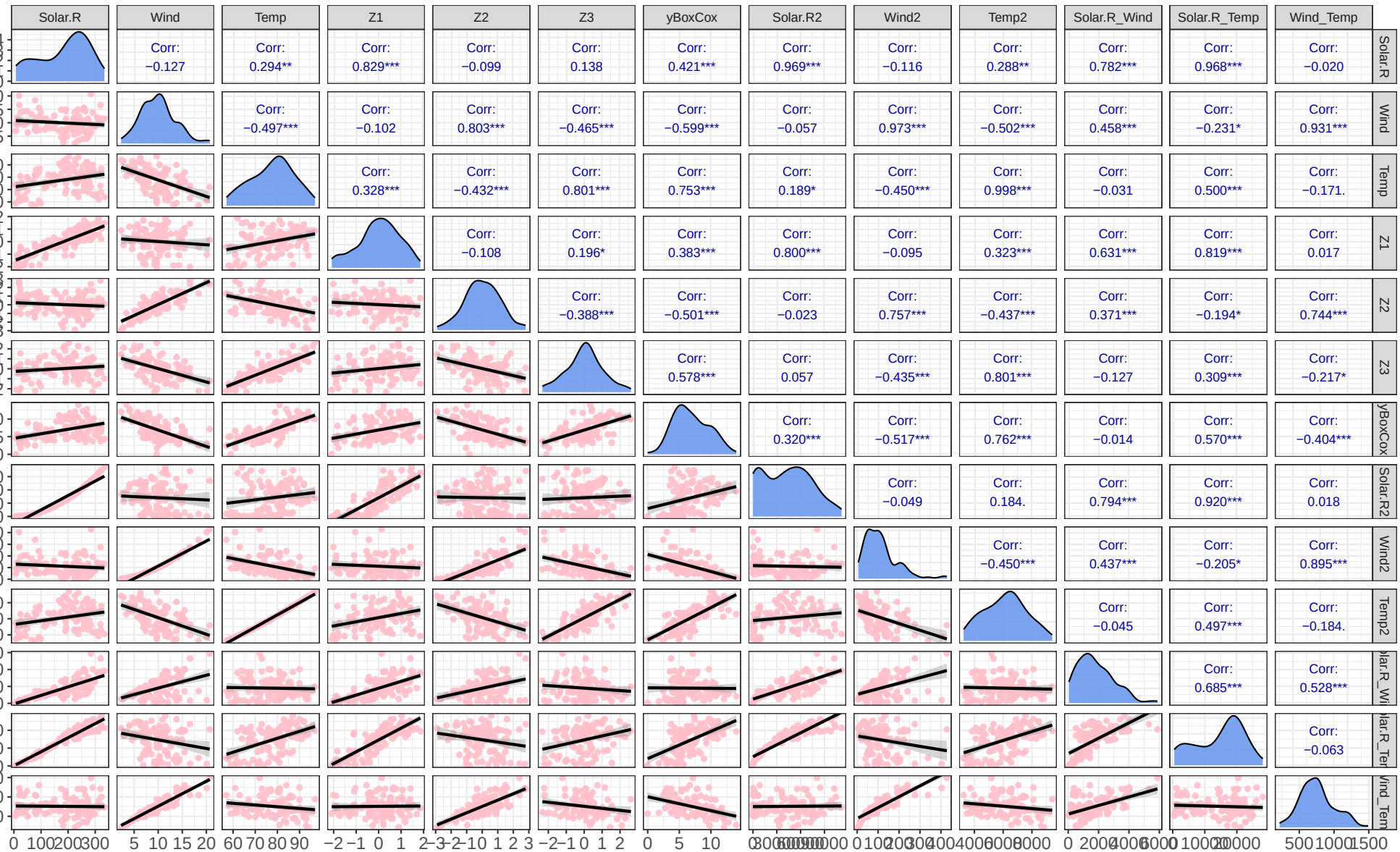
- Scale *Solar.R*, *Wind*, and *Temp*.
- Create Noise as a random normal variable.
- Construct $Z1$, $Z2$, and $Z3$ using the formula above.

Create Z1, Z2, Z3 predictors

```r
set.seed(3)
# Scaling func.
scale_var <- function(x) {
  return((x - mean(x)) / sd(x))
}
# Function to create noisy predictor with desired correlation
create_noisy_predictor <- function(variable, rho) {
  noise <- rnorm(length(variable))
  return(rho * variable + sqrt(1 - rho^2) * noise)
}
data_$Z1 <- create_noisy_predictor(scale_var(data_$Solar.R), 0.8)
data_$Z2 <- create_noisy_predictor(scale_var(data_$Wind), 0.8)
data_$Z3 <- create_noisy_predictor(scale_var(data_$Temp), 0.8)
# Checking the correlations
cor(data_$Z1, data_$Solar.R) # Should be approximately 0.8
cor(data_$Z2, data_$Wind)    # Should be approximately 0.8
cor(data_$Z3, data_$Temp)    # Should be approximately 0.8

>> 0.8293
>> 0.8026
>> 0.8014
```

FORTH

Pairplot with the additional predictors



Tofallis 2015 paper

- Transform the response in Model 4 so that your model complies with Relative Error Minimization as discussed in the attached paper by Tofallis (2015). Is the transformation similar or different to the one suggested by Box-Cox? Compare your new model (M5) against M4 and perform residual diagnostics on M5; is there an improvement relative to M3?

Tofallis 2015 paper

- Transform the response in Model 4 so that your model complies with Relative Error Minimization as discussed in the attached paper by Tofallis (2015). Is the transformation similar or different to the one suggested by Box-Cox? Compare your new model (M5) against M4 and perform residual diagnostics on M5; is there an improvement relative to M3?

```
M4_formula <- y_boxcox ~ 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

M4.ols <- lm(M4_formula, data = data_)
```

Tofallis 2015 paper

- Transform the response in Model 4 so that your model complies with Relative Error Minimization as discussed in the attached paper by Tofallis (2015). Is the transformation similar or different to the one suggested by Box-Cox? Compare your new model (M5) against M4 and perform residual diagnostics on M5; is there an improvement relative to M3?

```
y <- data_$Ozone
data_$y_boxcox <- y_boxcox
data_$log_y <- log(y) #We need that dtype  for later on
M5_formula <- log_y ~ 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

M5.ols <- lm(M5_formula, data = data_)
ols_test_breusch_pagan(M5.ols)
dwtest(M5.ols)
>>
```

FORTH

Tofallis 2015 paper - Relative error transformation

```
Breusch Pagan Test for Heteroskedasticity
 ---------------------------------------
 Ho: the variance is constant
 Ha: the variance is not constant

              Data
 ---------------------------------
 Response : log_y
 Variables: fitted values of log_y

         Test Summary
 ---------------------------------
 DF              =     1
 Chi2            =     22.83046
 Prob > Chi2     =     1.769392e-06

    Durbin-Watson test

data:  M5.ols
DW = 2.0301, p-value = 0.5549
alternative hypothesis: true autocorrelation is greater than 0
```

FORTH

Tofallis 2015 paper - Relative error transformation

```
summary(M5.ols)
>>
Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    6.161e+00  5.671e+00   1.086   0.2800
Solar.R        8.025e-03  6.183e-03   1.298   0.1974
Wind          -4.039e-01  2.790e-01  -1.448   0.1509
Temp          -7.193e-02  1.195e-01  -0.602   0.5487
I(Solar.R^2)  -1.741e-05  7.196e-06  -2.419   0.0174 *
I(Wind^2)      9.089e-03  4.219e-03   2.154   0.0337 *
I(Temp^2)      5.612e-04  6.662e-04   0.842   0.4016
Z1            -1.469e-03  9.672e-02  -0.015   0.9879
Z2             7.105e-02  7.305e-02   0.973   0.3331
Z3             2.386e-02  8.190e-02   0.291   0.7714
Solar.R:Wind  -1.388e-04  1.802e-04  -0.770   0.4429
Solar.R:Temp   2.621e-05  6.917e-05   0.379   0.7056
Wind:Temp      2.016e-03  2.723e-03   0.741   0.4607
---
Residual standard error: 0.495 on 98 degrees of freedom
Multiple R-squared:  0.7089,  Adjusted R-squared:  0.6732
F-statistic: 19.88 on 12 and 98 DF,  p-value: < 2.2e-16
```

FORTH

Tofallis 2015 paper - Relative error transformation

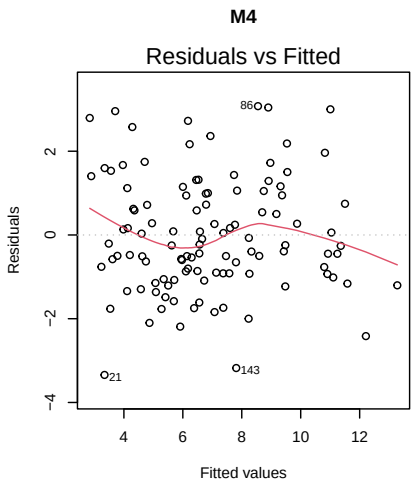
```
ols_test_breusch_pagan(M4.ols)$p
ols_test_breusch_pagan(M5.ols)$p
>>
0.650352818060884 #M4 deals with heteroscedasticity
1.76939234875026e-06
```

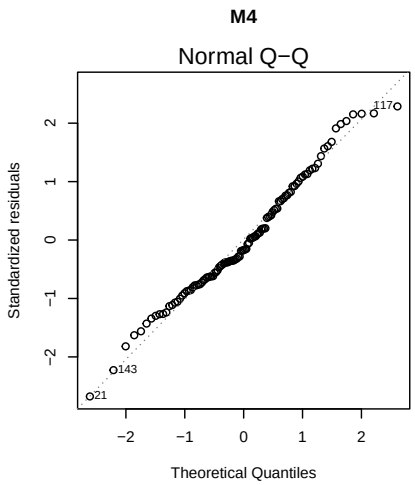
We can not compare models with different response via AICc/BIC/R^2

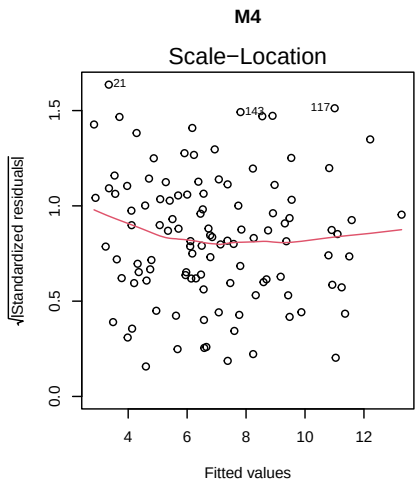
```
library(AICcmodavg)
AICc(M4.ols)
AICc(M5.ols)
BIC(M4.ols)
BIC(M5.ols)
summary(M4.ols)$r.squared
summary(M5.ols)$r.squared
>>
417.3133
177.4368
450.8717
210.9953
0.7528444
0.7088503
```

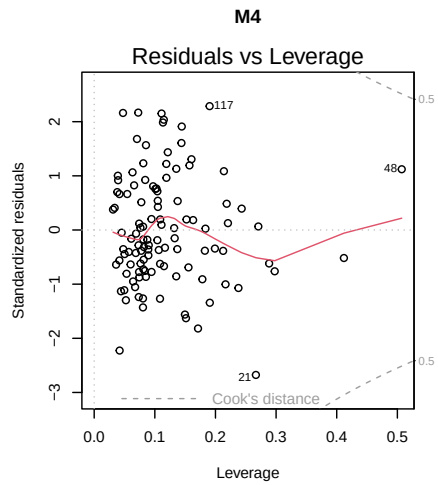
R-squared is the amount of variance in the response variable that is explained by the model. However, the R^2 of the first model describes the variance explained of $\mathbf{y}$, which is something different from that in the transformed model $\big(\log(\mathbf{y})\big)$. (**Source**)

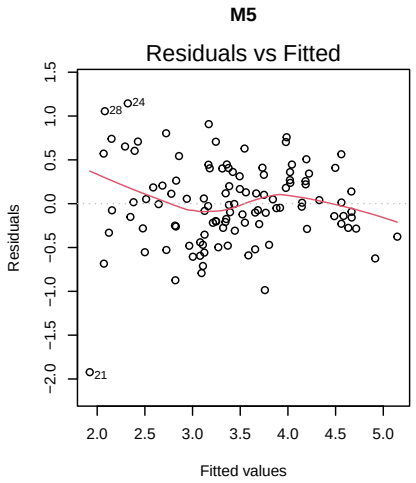
Tofallis 2015 paper - Relative error transformation

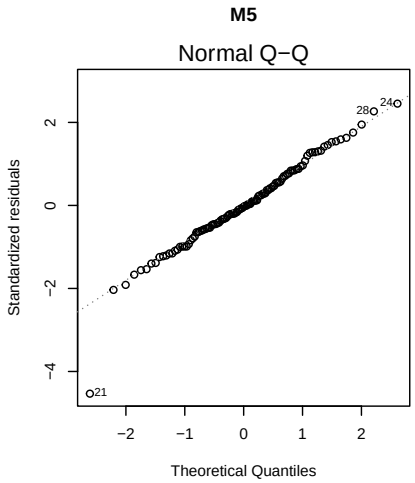


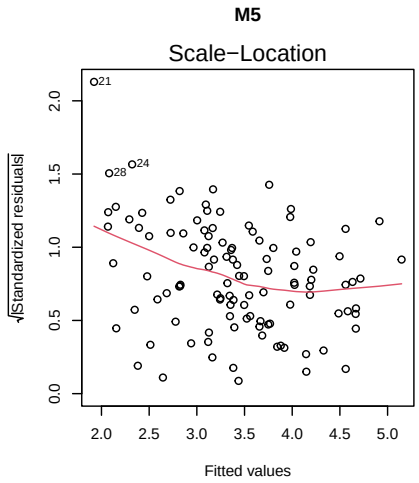


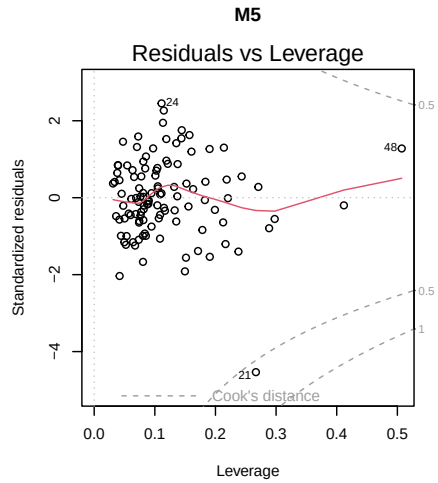










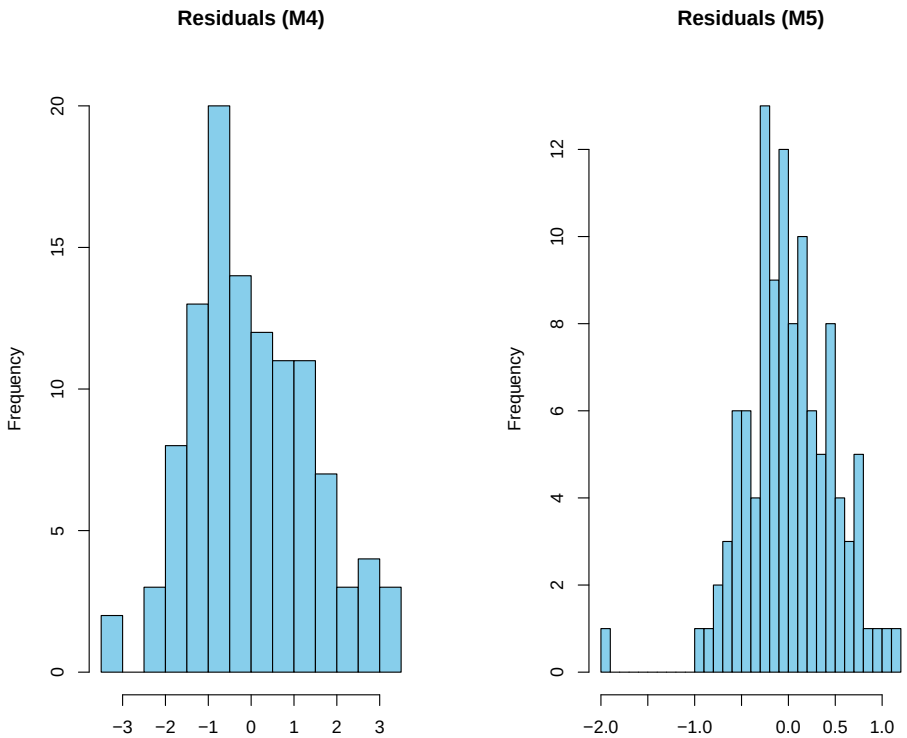


Tofallis 2015 paper - Relative error transformation

Box-Cox vs Tofalis log transformation:

- M4 deals with **heteroscedasticity**.
- M5 seems to have issues with **heteroscedasticity**.

```
par(mfrow = c(1, 2))
hist(residuals(M4.ols),  main = "Residuals (M4)",
     xlab="Residuals", col="skyblue", breaks=35)
hist(residuals(M5.ols),  main="Residuals (M5)",
    xlab="Residuals", col="skyblue", breaks=35)
```



Set up the metrics for cross-validation

```
LAR <- function(y, forcast) { #Log Acc. Ration
  n <- length(y)
  lar <- 0
  n_ <- 0 #counter
  for (i in 1:n) {
    if (y[i] != 0) {
      Q = as.numeric(forcast)[i]/y[i]
        if (Q > 0) {
          lar = lar + abs(log(Q))
          n_ = n_ + 1
        }}}
  lar <- lar/n_
  return(lar)
}
MAPE <- function(y, forcast) {
  n <- length(y)
  mape_sum <- 0
  n_ <- 0 #counter
  for (i in 1:n) {
    if (y[i] != 0) {
      mape_sum = mape_sum + abs((y[i]-as.numeric(forcast)[i])/y[i])
      n_ = n_ + 1
    }
  }
  mape <- (mape_sum/n_) * 100
  return(mape)
}
MAE <- function (actual, preds) mean(abs(actual-preds))
RMSE <- function (actual, preds) sqrt(mean((actual-preds)^2))
sMdAPE <- function(actual, preds) median(
                                    200*abs(actual-preds)/(actual+preds),
                                    na.rm = TRUE)
```

FORTH

Cross-validation (with transformation of the response)

```
library(foreach)
KfoldCVPerf <- function (K = 10, data, model_type = 'lm',
                         formula, name = 'model name',
                    folds.seed = 42, inv.trans= function (x) return(x) ) {
    ### Datatypes ###
    ## K: int #data: data.frame #model_type: 'lm' || 'lad'
    ## formula: as.formula() ## folds.seed: int: random folds
    #### inv.trans: inverse transformation of the response ####
    ### NOTE: The response y must be in the dataframe ###
    # Shuffle the data to get random folds:
    set.seed(folds.seed)
    data <- data[sample(nrow(data)), ]
    y <- all.vars(formula)[1] # get response name (str)
    rows_per_fold <- nrow(data)/K
    #Create a binning var
    binning_variable <- cut(seq(1, nrow(data)), breaks = K, labels = FALSE)
    folds <- split(data, binning_variable) # Split the folds
    folds_list <- list() # Create a list of K folds
    for (i in 1:K) {
        folds_list[[i]] = folds[[i]]
        }
    perf_df <- data.frame(
        Fold = integer(0), RMSE = numeric(0), MAE = numeric(0),
        sMdAPE = numeric(0), LAR = numeric(0), MAPE = numeric(0))
    #Loop through the folds
    foreach (i = 1:K) %do% {
        val_fold <- folds_list[[i]]
        y_val <- val_fold[[y]]
        train_folds <- do.call(rbind, folds_list[-i])
        y_train <- train_folds[[y]]
        # Change the format of the formula in order to convert response
        formula_ <- paste(all.vars(formula)[-1], collapse = ' + ')
```

Cross-validation (with transformation of the response)

```
        if (model_type == 'lm') {
            model <- lm(as.formula(paste("y_train~", formula_)),
                        data=train_folds)
        } else if (model_type == 'lad') {
            model <- lad(as.formula(paste("y_train~", formula_)),
                         data=train_folds)
        } else {
            message("Unsupported model type.")
            break}
        preds <- predict(model, newdata = val_fold)
        y_val <- inv.trans(y_val)
        preds <- inv.trans(preds)
        rmse_val <- RMSE(y_val, preds)
        mae_val <- MAE(y_val, preds)
        smdape_val <- sMdAPE(y_val, preds)
        MAPE_val <- MAPE(y_val, preds)
        LAR_val <- LAR(y_val, preds)
        perf_df <- rbind(perf_df, data.frame(
                                  Fold = i, RMSE = rmse_val,
                                  MAE = mae_val, sMdAPE = smdape_val,
                                  MAPE = MAPE_val, LAR = LAR_val))
    }
    mean_rmse <- mean(perf_df$RMSE)
    mean_mae <- mean(perf_df$MAE)
    mean_smdape <- mean(perf_df$sMdAPE)
    mean_MAPE <- mean(perf_df$MAPE)
    mean_LAR <- mean(perf_df$LAR)
    # Add the means to the df
    perf_df <- rbind(perf_df, c("Mean", mean_rmse, mean_mae, mean_smdape, mean_MAPE, mean_LAR))
    colnames(perf_df)[-1] <- paste0(name, ' (', colnames(perf_df)[-1], ')')
    return (perf_df)
}
```

Cross-validation (with transformation of the response)

Example

```
KfoldCVPerf(K=10, data=data_, formula=M4_formula, name='M4',
            inv.trans= function (y) (lambda.best*y + 1)^(1/lambda.best))
```

Fold	M4 (**RMSE**)	M4 (**MAE**)	M4 (**sMdAPE**)	M4 (**MAPE**)	M4 (**LAR**)
1	13.4844639394214	11.722952456799	37.3501032481086	44.4310852974232	0.42197374564178
2	16.4394103204079	12.9306891690037	42.999233192946	49.2779504040391	0.404678422436324
3	35.1132950868163	19.9278180375665	27.6364194117723	39.7538616913861	0.441930070869637
4	23.4537311108301	15.1247076248887	29.354174271746	30.5885051836696	0.316859449685931
5	15.0432980699381	13.2118868133534	33.3835683169767	48.3219687016842	0.503947671845283
6	17.6450654467307	12.3756864535048	25.5236236495682	40.0221035385367	0.38466353961046
7	24.2386967241797	17.622639684578	41.8945739314198	53.6521920721395	0.536175225467513
8	14.1999279686414	10.8636915903395	40.3606155800208	50.2385711635589	0.419861171663608
9	11.2999901354722	10.0769371570312	23.3068862055817	51.1617299299498	0.367456908732039
10	21.2692243135157	14.9242266585308	48.6564201229144	130.38199296247	0.666525243883205
Mean	**19.2187103115953**	**13.8781235645596**	**35.0465617931055**	**53.7829960944857**	**0.446407144983578**

Repeated cross-validation

```
RepeatedCV <- function (repeats=500, K=10, data, formula,
                        model_name, inv.trans = function(y) y) {
    require(foreach)
    mean_rows_list <- list() # store the row_means
    # Choose diff random.sds in order to get diff folds
    foreach (seed = 1:repeats) %do% {
        cv_results <- KfoldCVPerf(K = K, data=data, formula=formula,
                                  name=model_name, folds.seed = seed,
                                  inv.trans = inv.trans)
        mean_row <- cv_results[cv_results[,1] == "Mean", -1]
        mean_rows_list[[seed]] <- as.numeric(mean_row)
    }
    # Combine the mean rows into a df
    mean_rows_df <- do.call(rbind, mean_rows_list)
    # Average for each metric
    average_metrics <- colMeans(mean_rows_df, na.rm = TRUE)
    average_metrics_df <- as.data.frame(t(average_metrics))
    colnames(average_metrics_df) <- names(cv_results)[-1]
    rownames(average_metrics_df) <- "RCV Average"
    return(average_metrics_df)
}
```

FORTH

Repeated cross-validation

Example

```
M3.RCV <- RepeatedCV(
    repeats = 500, K = 10, data = data_, formula = M3_formula, model_name = 'M3')

M4.RCV <- RepeatedCV(
    repeats = 500, K = 10, data = data_,   formula = M4_formula, model_name = 'M4',
    inv.trans = function(y) (lambda.best * y + 1)^(1 / lambda.best))

M5.RCV <- RepeatedCV(
    repeats = 500, K = 10, data = data_, formula = M5_formula, model_name = 'M5',
    inv.trans = function(y) exp(y))
>>
```

M3.RCV	M3 (RMSE)	M3 (MAE)	M3 (sMdAPE)	M3 (MAPE)	M3 (LAR)
RCV Average	20.728841315252	16.1222941795992	33.8353230750363	67.7265253872971	0.438285472180775

M4.RCV	M4 (RMSE)	M4 (MAE)	M4 (sMdAPE)	M4 (MAPE)	M4 (LAR)
RCV Average	18.5803706748303	13.8479408576176	34.8100960781422	53.5017843693086	0.450559758646985

M5.RCV	M5 (RMSE)	M5 (MAE)	M5 (sMdAPE)	M5 (MAPE)	M5 (LAR)
RCV Average	18.731246022978	13.5511881738598	33.5722525817141	49.2004350320875	0.41236694504943

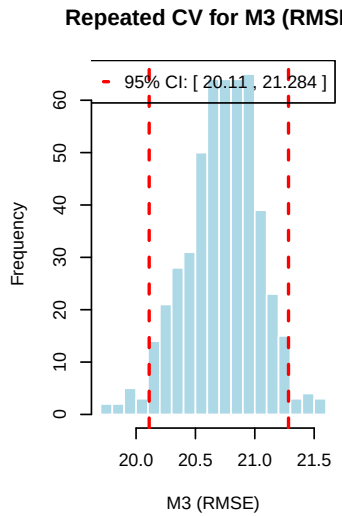
Repeated cross-validation: 95% CI

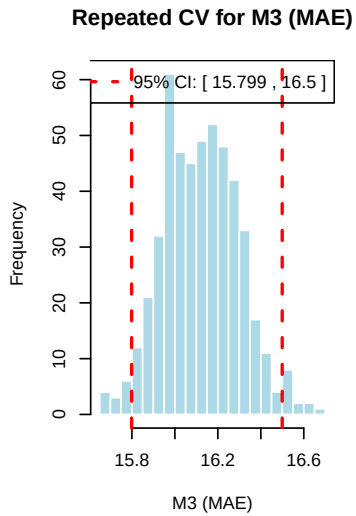
```
RCV.histograms <- function(repeats=100, K=10, data=data_,
                           formula, name, inv.trans=function(y) y) {
    require(foreach)
    metrics_values_list <- list()
    foreach (seed = 1:repeats) %do% {
        M5.cv <- KfoldCVPerf(K = K, data = data, formula = formula, name = name,
                             folds.seed = seed, inv.trans = inv.trans)
        # Extract the "Mean" row and store the metrics values
        mean_row <- M5.cv[M5.cv[,1] == "Mean", -1]
        for (metric in names(mean_row)) {
            metrics_values_list[[metric]] <- c(metrics_values_list[[metric]],
                                               as.numeric(mean_row[metric]))
        }
    }
    # plot the histograms for each metric and add the CI lines
    par(mfrow=c(2, 3))  # Set up the plotting area to display multiple plots
    for (metric in names(metrics_values_list)) {
        # Calculate the 95% CI for the metric
        metric_values <- unlist(metrics_values_list[metric])
        ci <- quantile(metric_values, probs=c(0.025, 0.975))
        hist(metric_values, main=paste("Repeated CV for", metric),
             xlab=metric, col='lightblue', border='white', breaks=15)
        abline(v=ci[1], col="red", lwd=2, lty=2) # left CI line
        abline(v=ci[2], col="red", lwd=2, lty=2) # right CI line
        legend("topright", legend=paste("95% CI: [", round(ci[1], 3),
                                        ',', round(ci[2], 3), "]"),
               col="red", lwd=2, lty=2)
    }
    par(mfrow=c(1, 1))
}
```

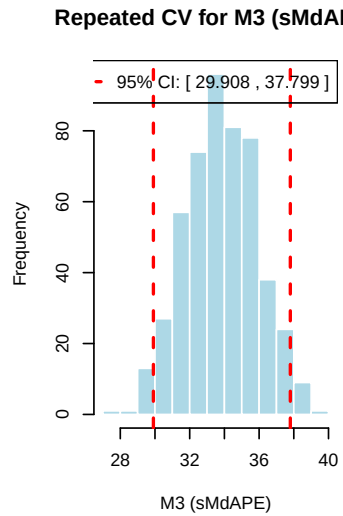
FORTH

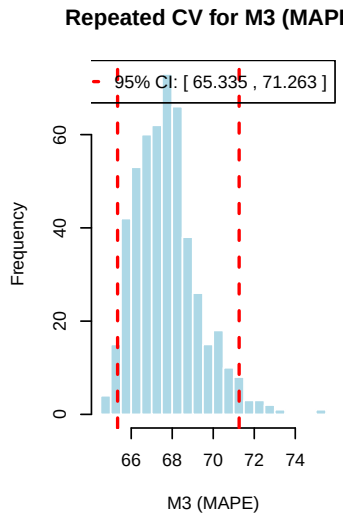
Repeated cross-validation - M3

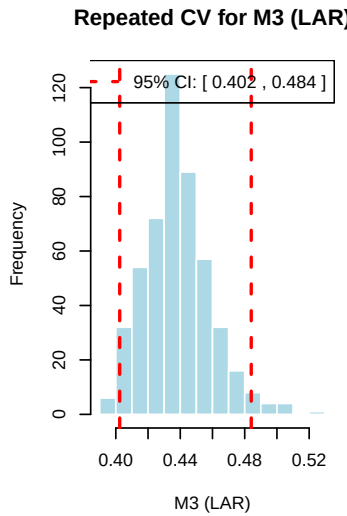
```
M3.RCVhists <- RCV.histograms(repeats=500, K=10, data=data_, formula=M3_formula, name='M3')
```





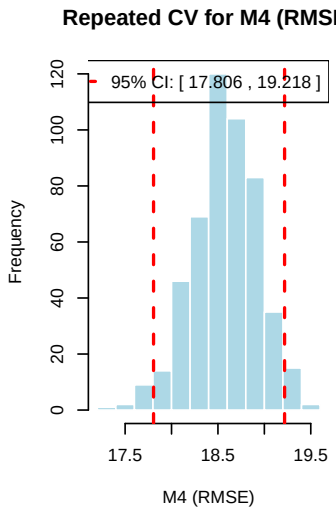


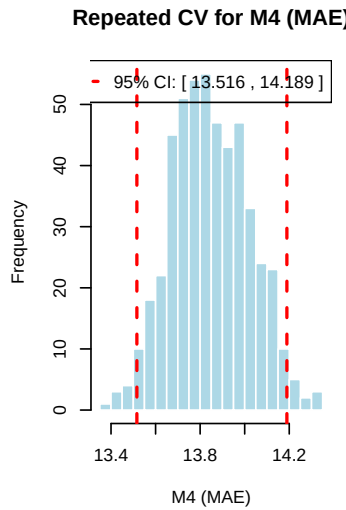


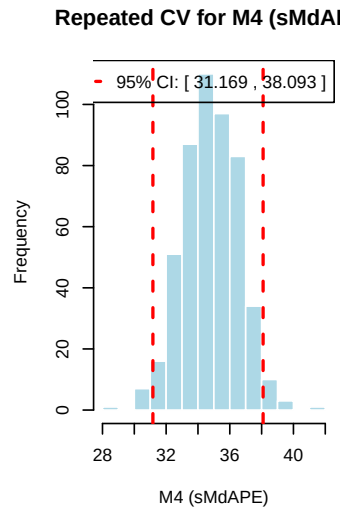


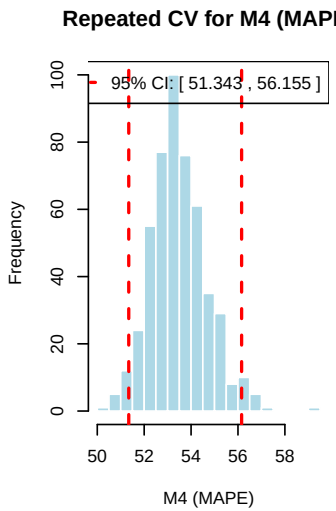
Repeated cross-validation - M4

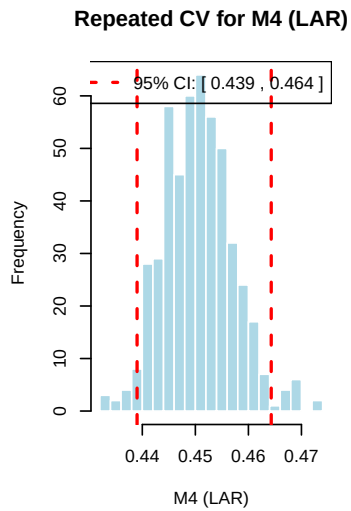
```
M4.RCVhists <- RCV.histograms(repeats=500, K=10, data=data_, formula=M4_formula, name='M4',
                              inv.trans= function (y) (lambda.best*y + 1)^(1/lambda.best))
```





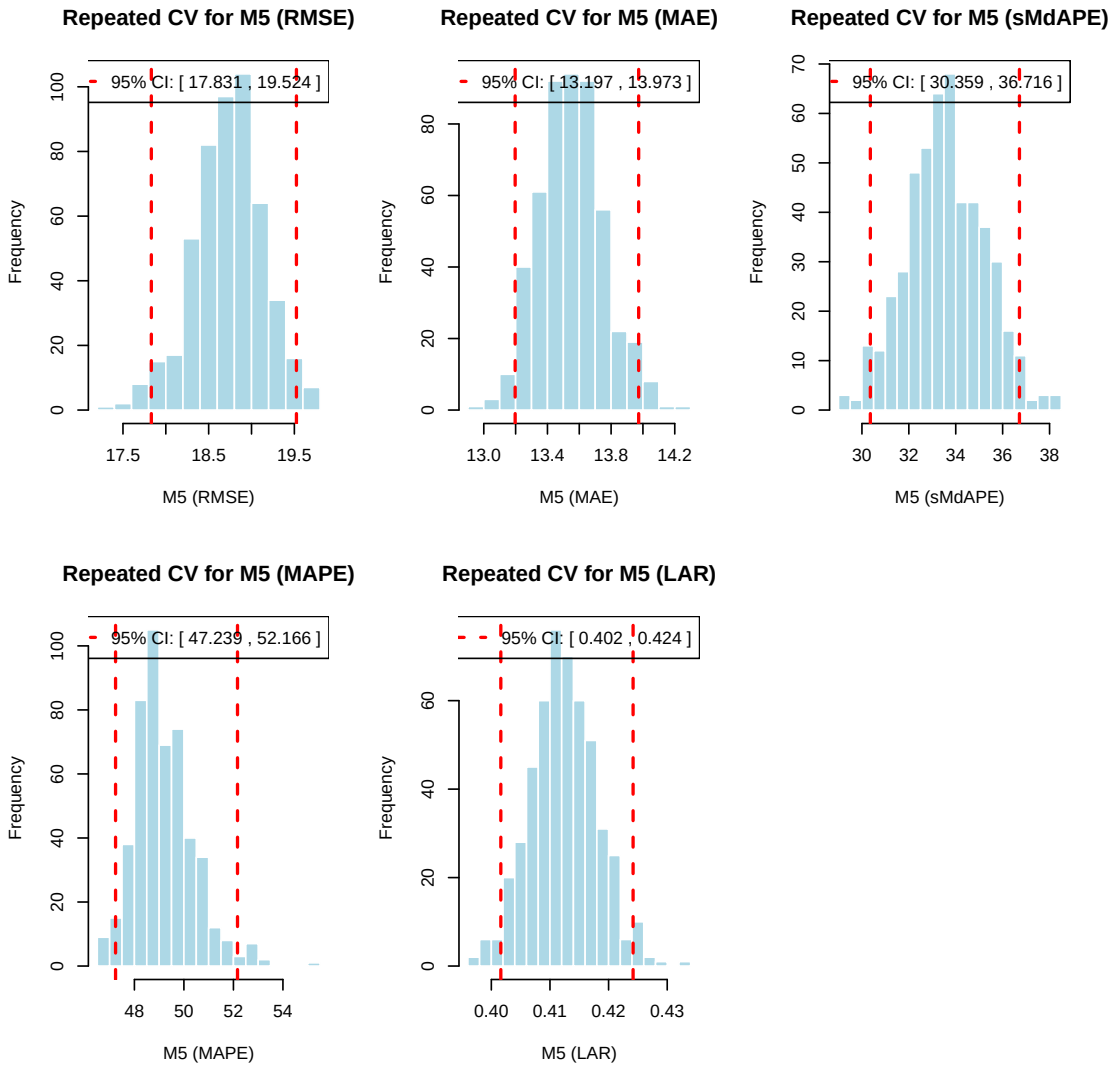






Repeated cross-validation - M5

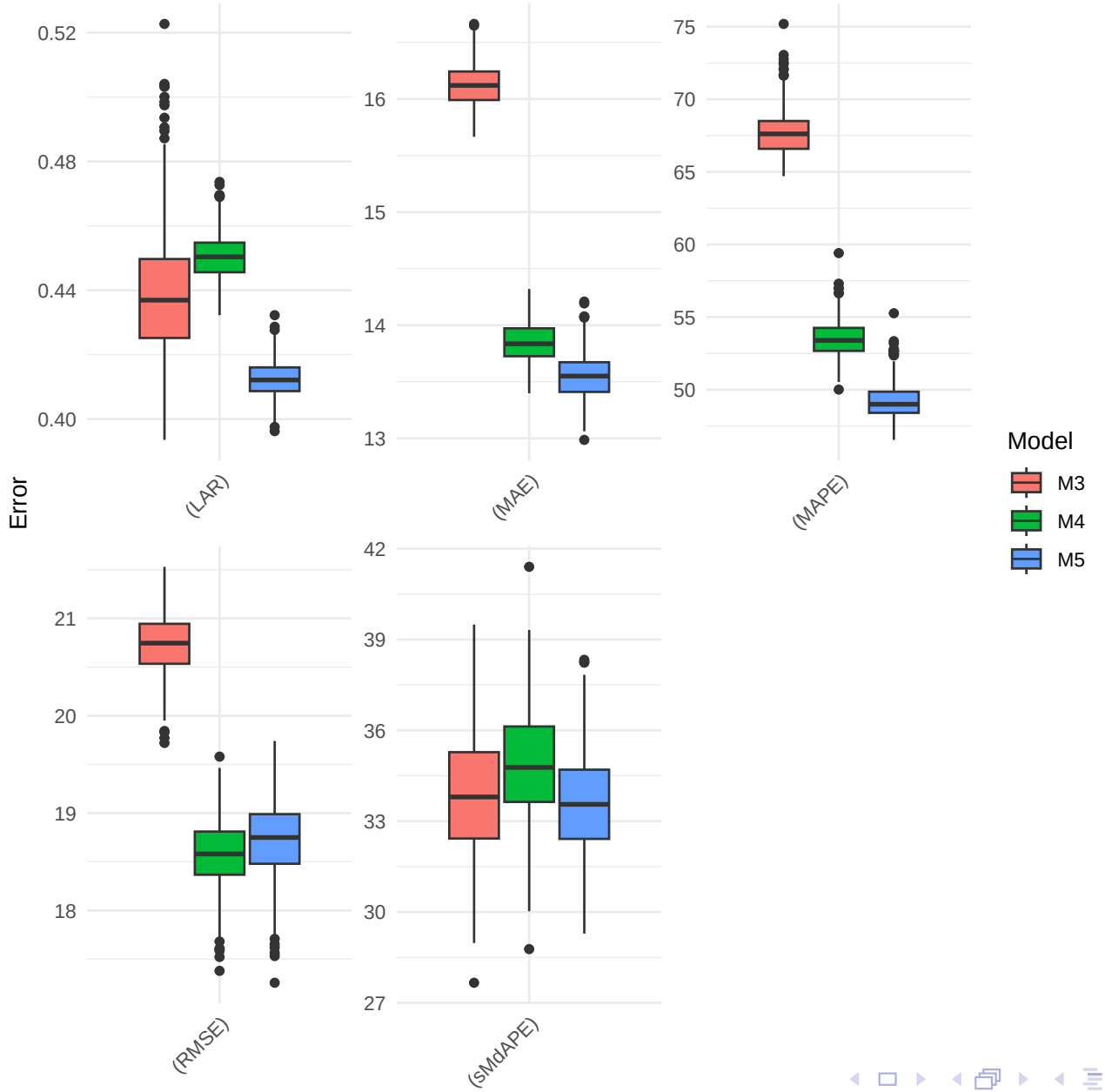
```
M5.RCVhists <- RCV.histograms(repeats=500, K=10, data=data_, formula=M5_formula, name='M5',
                              inv.trans= function (y) exp(y))
```



Repeated cross-validation - Box-Plots

```
RCV.boxplots_combined <- function(repeats=500, K=10, data, M3_formula,
                                  M4_formula, M5_formula, inv.trans_M3,
                                  inv.trans_M4, inv.trans_M5) {
    require(ggplot2)
    combined_metrics <- data.frame(metric = character(),
                                   value = numeric(), model = factor(),
                                   stringsAsFactors = FALSE)
    # List of models and their respective formulas and inverse transformations
    models_list <- list(
        M3 = list(formula = M3_formula, trans = inv.trans_M3),
        M4 = list(formula = M4_formula, trans = inv.trans_M4),
        M5 = list(formula = M5_formula, trans = inv.trans_M5)
    )
    for (model_name in names(models_list)) {
        foreach (seed = 1:repeats) %do% {
            cv_results <- KfoldCVPerf(K, data, 'lm',
                                      models_list[[model_name]]$formula,
                                      model_name, seed, models_list[[model_name]]$trans)
            mean_row <- cv_results[cv_results[, 1] == "Mean", -1]
            # Append results to the combined metrics data frame
            combined_metrics <- rbind(combined_metrics, data.frame(
                metric = names(mean_row), value = as.numeric(mean_row), model = model_name))
        }
    }
    # Extract only the metric name (after the space)
    combined_metrics$metric <- sapply(strsplit(combined_metrics$metric, " "), function(x) x[2])
    p <- ggplot(combined_metrics, aes(x = metric, y = value, fill = model)) + geom_boxplot() +
        facet_wrap(~ metric, scales = 'free') + theme_minimal() +
        theme(axis.text.x = element_text(angle = 45, hjust = 1),
              strip.text.x = element_blank()) +
        labs(x = NULL, y = "Error", fill = "Model")
    print(p)
}
```

Repeated cross-validation - Box-Plots based on 500 repeats



Assignment 2 Part 2

Examine alternative tools for predictor selection on M5.

- First use regsubsets to perform BIC-based forward and backward selection (F-BIC, B-BIC); which is the most parsimonious specification?

Assignment 2 Part 2

Examine alternative tools for predictor selection on M5.

- First use regsubsets to perform BIC-based forward and backward selection (F-BIC, B-BIC); which is the most parsimonious specification?

```
M5.forward <- regsubsets(M5_formula, data=data_,
                         method="forward", nvmax=12)
M5.backward <- regsubsets(M5_formula, data=data_,
                         method="backward", nvmax=12)

M5.Fbic <- summary(M5.forward)$bic
M5.Bbic <- summary(M5.backward)$bic
M5.Fbic
M5.Bbic
>>
```

```
-81.616  -94.299  -105.724  -105.455  -103.401  -100.827  -97.223  -93.641  -89.421  -85.064
-80.45  -75.741

-81.616  -94.137  -103.77  -105.762  -106.207  -102.545  -98.415  -93.857  -89.612  -85.064
-80.45  -75.741
```

FORTH

F-BIC

Examine alternative tools for predictor selection on M5.

```
summary(M5.forward)$outmat
```

summary	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)	*	*		*	*	*		*			*	*
9 (1)	*	*		*	*	*		*		*	*	*
10 (1)	*	*	*	*	*	*		*		*	*	*
11 (1)	*	*	*	*	*	*		*	*	*	*	*
12 (1)	*	*	*	*	*	*	*	*	*	*	*	*

B-BIC

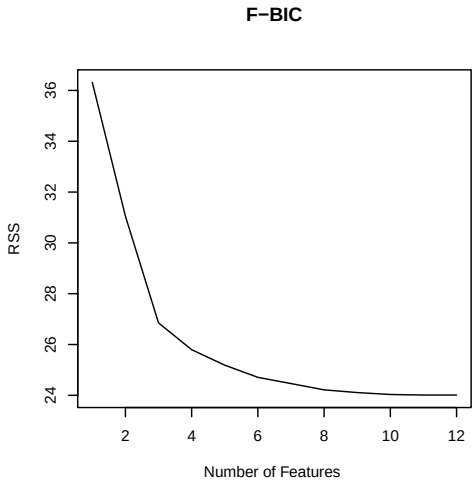
Examine alternative tools for predictor selection on M5.

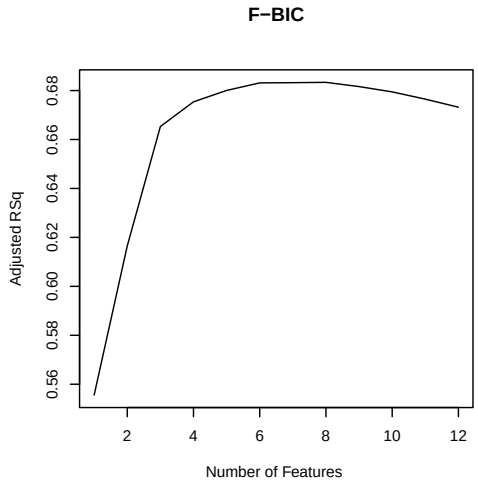
```
summary(M5.backward)$outmat
```

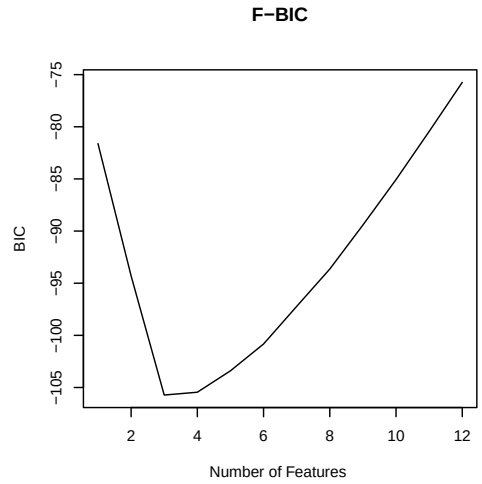
summary	Solar.R	Wind	Temp	I(Solar.R^2)	I(Wind^2)	I(Temp^2)	Z1	Z2	Z3	Solar.R:Wind	Solar.R:Temp	Wind:Temp
1 (1)						*						
2 (1)	*					*						
3 (1)	*	*				*						
4 (1)	*	*			*	*						
5 (1)	*	*		*	*	*						
6 (1)	*	*		*	*	*		*				
7 (1)	*	*		*	*	*		*		*		
8 (1)	*	*		*	*	*		*		*		*
9 (1)	*	*	*	*	*	*		*		*		*
10 (1)	*	*	*	*	*	*		*		*	*	*
11 (1)	*	*	*	*	*	*		*	*	*	*	*
12 (1)	*	*	*	*	*	*	*	*	*	*	*	*

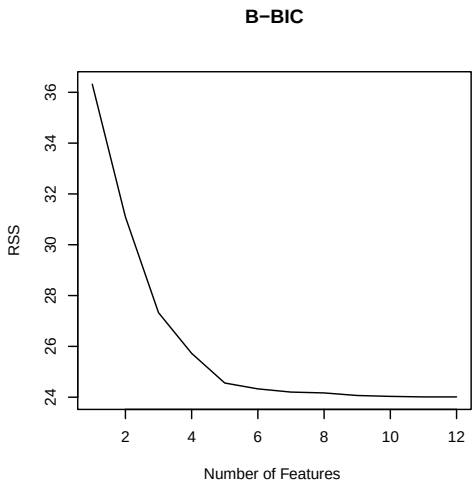
F-BIC; B-BIC plots

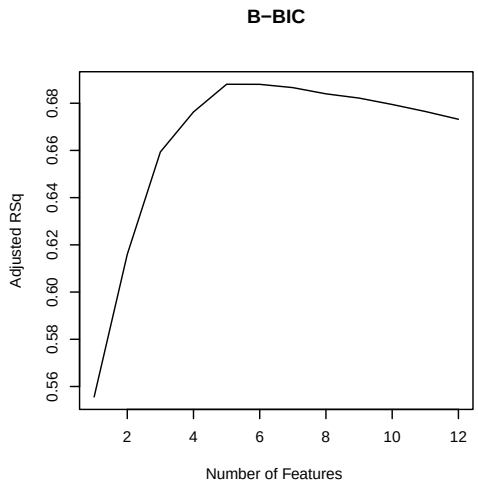
Examine alternative tools for predictor selection on M5.

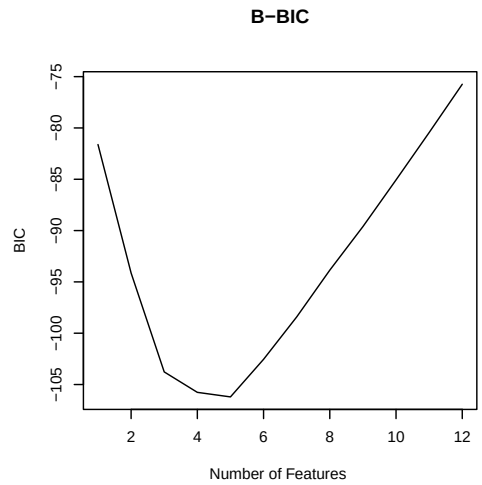






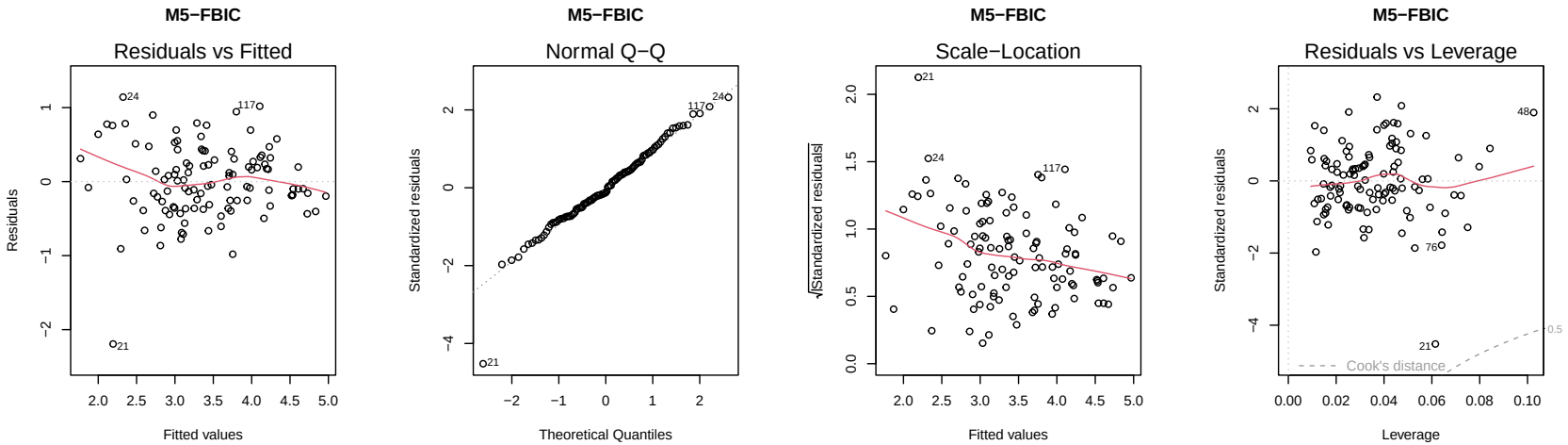


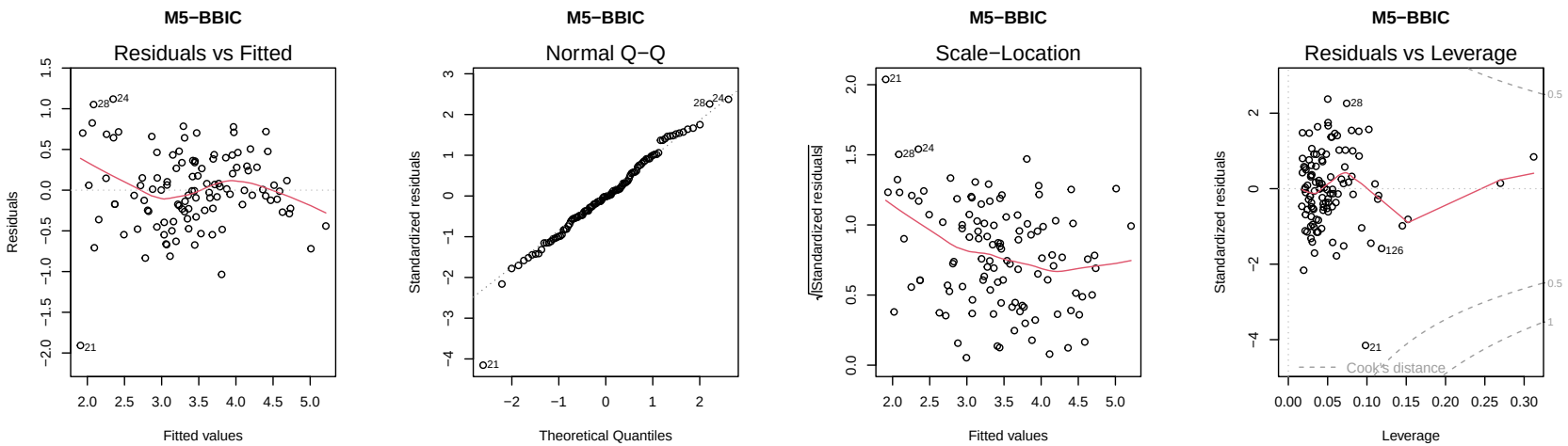




F-BIC; B-BIC residuals

```
par(mfrow=c(2,4))
plot(M5.FBIC, main="M5-FBIC")
plot(M5.BBIC, main="M5-BBIC")
```





F-BIC; B-BIC est.

Examine alternative tools for predictor selection on M5.

```
coef(M5.forward, which.min(M5.Fbic))
coef(M5.backward, which.min(M5.Bbic))
>>
(Intercept): 1.494   I(Temp^2): 0.00033   Solar:R: 3.50e-05   Wind:Temp: -0.000807

(Intercept): 2.279   Solar:R: 0.0075   Wind: -0.211   I(Solar.R^2): -1.49e-05   I(Wind^2):
     0.007 I(Temp^2) 0.00027


# Compare and choose the most parsimonious one
M5F_lowest.bic <- M5.Fbic[which.min(M5.Fbic)]
M5B_lowest.bic <- M5.Bbic[which.min(M5.Bbic)]
M5F.len <- length(coef(M5.forward, which.min(M5.Fbic))[-1])
M5B.len <- length(coef(M5.backward, which.min(M5.Bbic))[-1])

# Parsimonious: if BICs are close choose the one with the < features
if (abs(M5F_lowest.bic-M5B_lowest.bic) < 3 && M5F.len < M5B.len) {
  cat("Most parsimonious model is from Forward Selection,",
      "BIC:", M5.Fbic[which.min(M5.Fbic)], ", features:", M5F.len)
} else {
  cat("Most parsimonious model is from Backward Selection,",
      "BIC:", M5.Bbic[which.min(M5.Fbic)], ", features:", M5B.len)
}
>>
Most parsimonious model is from Forward Selection, BIC: -105.7242 , features: 3
```

F-BIC; B-BIC vs F-BIC; B-AIC

- Compare F-BIC, B-BIC to F-AIC (use `ols_step_forward_aic` from `olsrr`) and B-AIC (`ols_step_backward_aic`).

F-BIC; B-BIC vs F-BIC; B-AIC

- Compare F-BIC, B-BIC to F-AIC (use `ols_step_forward_aic` from `olsrr`) and B-AIC (`ols_step_backward_aic`).

```
M5.FAIC <- ols_step_forward_aic(M5.ols)
M5.BAIC <- ols_step_backward_aic(M5.ols)

M5.FAIC
M5.BAIC
>>
```

F-BIC; B-BIC vs F-BIC; B-AIC

>>

Selection Summary

Variable	AIC	Sum Sq	RSS	R-Sq	Adj. R-Sq
I(Temp^2)	196.992	46.152	36.317	0.55963	0.55559
Solar.R:Temp	181.599	51.420	31.050	0.62349	0.61652
Wind:Temp	167.464	55.620	26.850	0.67443	0.66530
I(Solar.R^2)	165.024	56.673	25.797	0.68720	0.67539
I(Wind^2)	164.369	57.283	25.187	0.69459	0.68005
Wind	164.233	57.763	24.707	0.70041	0.68313

>>

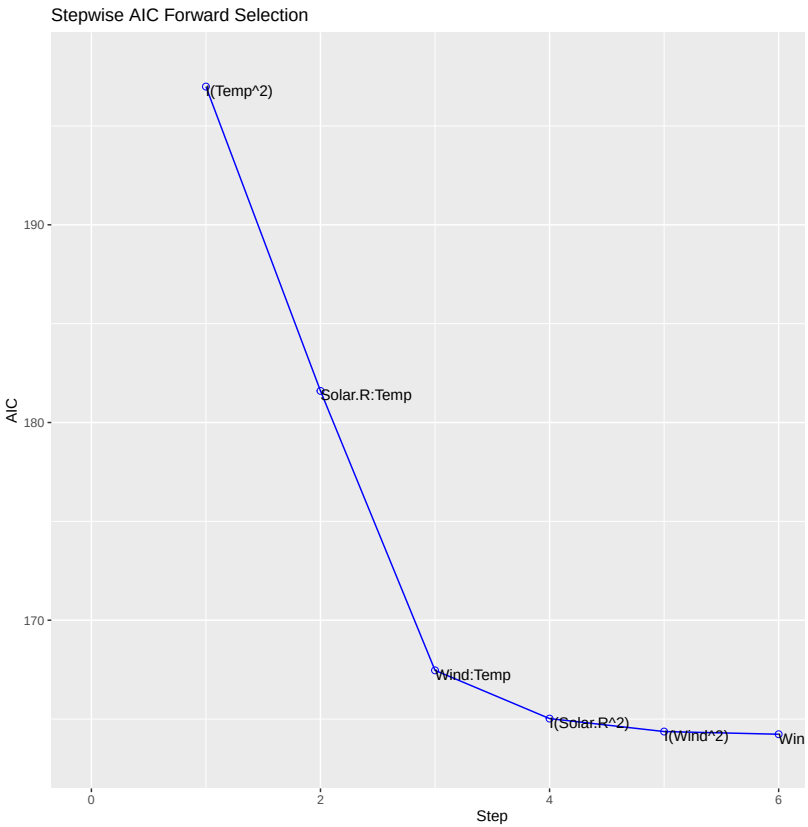
Backward Elimination Summary

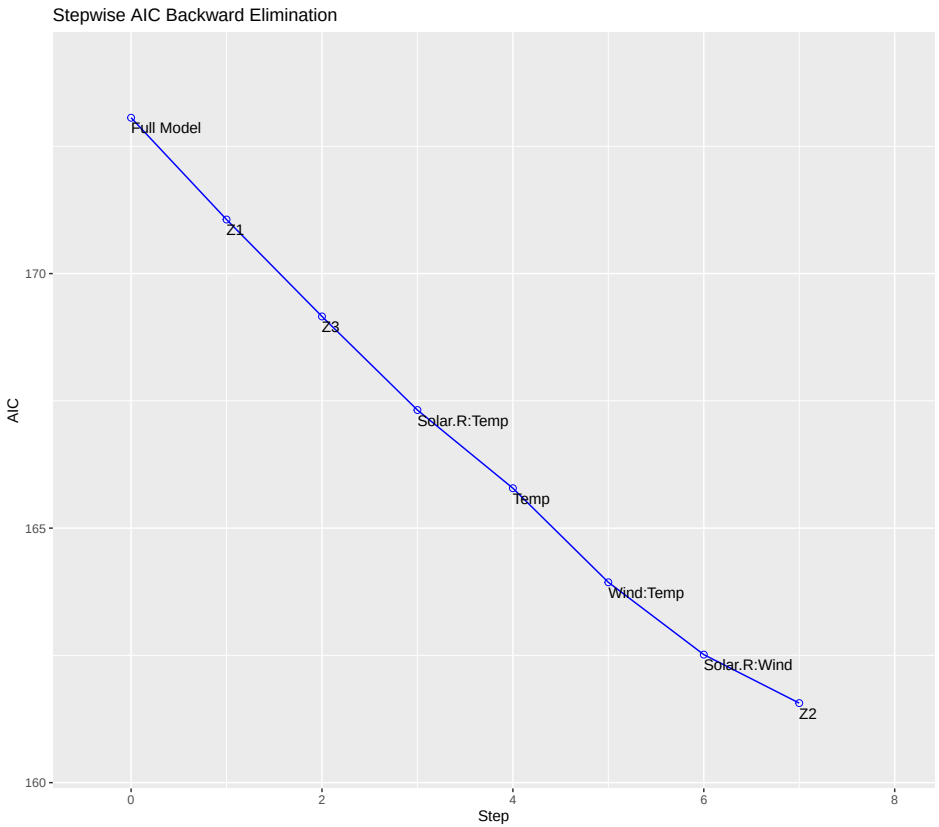
Variable	AIC	RSS	Sum Sq	R-Sq	Adj. R-Sq
Full Model	173.062	24.011	58.459	0.70885	0.67320
Z1	171.062	24.011	58.459	0.70885	0.67650
Z3	169.158	24.032	58.438	0.70860	0.67946
Solar.R:Temp	167.320	24.067	58.403	0.70817	0.68217
Temp	165.784	24.168	58.302	0.70695	0.68397
Wind:Temp	163.936	24.201	58.269	0.70655	0.68661
Solar.R:Wind	162.515	24.328	58.142	0.70501	0.68799
Z2	161.562	24.558	57.912	0.70222	0.68804

FORTH

F-AIC; B-AIC plots

```
plot(M5.FAIC)
plot(M5.BAIC)
>>
```





Compare F-AIC; B-AIC with F-BIC; B-BIC

```
AIC_variables <- c(AIC(M5.FAIC$model), AIC(M5.BAIC$model), AIC(M5.FBIC), AIC(M5.FBIC))
AICc_variables <- c(AICc(M5.FAIC$model), AICc(M5.BAIC$model), AICc(M5.FBIC), AICc(M5.FBIC))
BIC_variables <- c(BIC(M5.FAIC$model), BIC(M5.BAIC$model), BIC(M5.FBIC), BIC(M5.FBIC))
metrics_matrix <- rbind(AIC = AIC_variables, AICc = AICc_variables, BIC = BIC_variables)
IC_df <- as.data.frame(metrics_matrix)
colnames(IC_df) <- c("F-AIC", "B-AIC", "F-BIC", "B-BIC")
IC_df
```

Criterion	F-AIC	B-AIC	F-BIC	B-BIC
AIC	164.232623164374	161.562457576751	167.464403165684	161.562457576751
AICc	165.644387870256	162.649836217528	168.035831737113	162.649836217528
BIC	185.908864774872	180.529168985937	181.012054172246	180.529168985937

Cross-validation for stepwise models

We can modify the `KfoldCVPerf.stepwise()` function:

```
KfoldCVPerf.stepwise <- function (K=10, data, formula, name='model name',
                                  folds.seed = 42, inv.trans=function(y) return(y),
                                  method="forward", criterion="AIC", nvmax=12) {
 ### Same code as KfoldCVPerf() ###
 foreach (i = 1:K) %do% {
        #Keep the i-th fold as it is (testing)
        val_fold <- folds_list[[i]]
        y_val <- val_fold[[y]]

        #Combine the remaining folds (training)
        train_folds <- do.call(rbind, folds_list[-i])
        y_train <- train_folds[[y]]

        # Temporarily assign train_folds to a global variable
        .GlobalEnv$train_folds_temp <- train_folds #Important for F-BAIC

        #Choose model
        if (method == "forward" && criterion == "AIC") {
            # Use the global temporary variable for model fitting
            LM <- lm(formula, data = train_folds_temp)
            model.FAIC <- ols_step_forward_aic(LM)
            model <- model.FAIC$model
        } else if (method == "forward" && criterion == "BIC") {
            model.forward <- regsubsets(formula, data=train_folds,
                                        method="forward", nvmax=nvmax)
            model.Fbic <- summary(model.forward)$bic
            FBIC_vars <- names(coef(model.forward, which.min(model.Fbic)))
            FBIC_formula <- as.formula(paste("y_train ~",
                                  paste(FBIC_vars[-1], collapse=' + ')))
            model <- lm(FBIC_formula, data=train_folds)
```

Cross-validation for stepwise models

```
    } else if (method == "backward" && criterion == "AIC") {
        # Use the global temporary variable for model fitting
        LM <- lm(formula, data = train_folds_temp)
        model.BAIC <- ols_step_backward_aic(LM)
        model <- model.BAIC$model
    } else if (method == "backward" && criterion == "BIC") {
        model.backward <- regsubsets(formula, data=train_folds,
                                     method="backward", nvmax=nvmax)
        model.Bbic <- summary(model.backward)$bic
        BBIC_vars <- names(coef(model.backward, which.min(model.Bbic)))
        BBIC_formula <- as.formula(paste("y_train ~",
                                paste(BBIC_vars[-1], collapse=' + ')))

        model <- lm(BBIC_formula, data=train_folds)
    } else {
        stop("Unsupported method or criterion.")
    }
    rm(train_folds_temp, envir = .GlobalEnv) #remove globenv

    preds <- predict(model, newdata = val_fold)
    y_val <- inv.trans(y_val)
    preds <- inv.trans(preds)

    ### Same code as before ###

    return (perf_df)
}
```

FORTH

Repeated cross-validation for stepwise models

```
RepeatedCV.stepwise <- function (repeats=50, K=10, data, method, formula,
                        model_name, criterion, inv.trans = function(y) y) {
    require(foreach)
    mean_rows_list <- list() # store the row_means
    foreach (seed = 1:repeats) %do% {
        cv_results <- KfoldCVPerf.stepwise(K=K, data=data, method=method,
                                criterion=criterion, formula=formula,
                                name=model_name, folds.seed = seed,
                                inv.trans=inv.trans)
        mean_row <- cv_results[cv_results[,1]=="Mean", -1]
        mean_rows_list[[seed]] <- as.numeric(mean_row)
    }
    mean_rows_df <- do.call(rbind, mean_rows_list)
    # Average for each metric
    average_metrics <- colMeans(mean_rows_df, na.rm = TRUE)
    average_metrics_df <- as.data.frame(t(average_metrics))
    colnames(average_metrics_df) <- names(cv_results)[-1]
    rownames(average_metrics_df) <- "RCV Average"
    return(average_metrics_df)
}
M5_FAIC.RCV <- RepeatedCV.stepwise(
    repeats = 50,
    K = 10,
    data = data_,
    method='forward',
    criterion='AIC',
    formula = M5_formula,
    model_name = 'M5-FAIC',
    inv.trans = function(y) exp(y))
### ... define BAIC, FBIC, BBIC ...        ###
```

FORTH

Repeated cross-validation - based on 50 repeats

```
M5_FAIC.RCV
M5_FBIC.RCV
M5_BAIC.RCV
M5_BBIC.RCV
>>
```

M5	M5-FAIC (RMSE)	M5-FAIC (MAE)	M5-FAIC (sMdAPE)	M5-FAIC (MAPE)	M5-FAIC (LAR)
RCV Average	20.7875055645847	14.5760054786763	33.7482568899758	50.21819388008	0.414177365614387

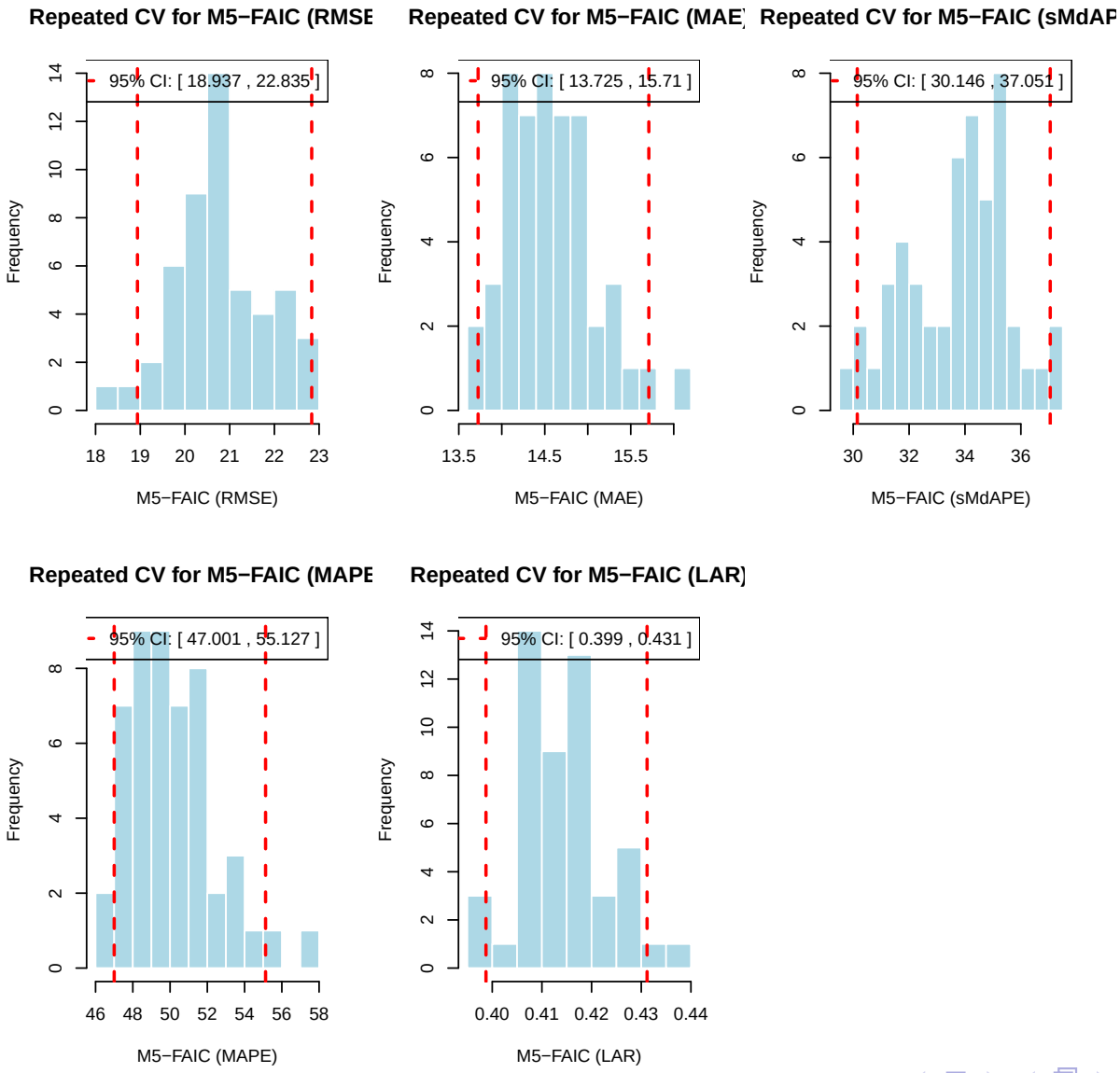
M5	M5-FBIC (RMSE)	M5-FBIC (MAE)	M5-FBIC (sMdAPE)	M5-FBIC (MAPE)	M5-FBIC (LAR)
RCV Average	20.1683579404873	14.4788907089285	33.713714195958	49.1706383379745	0.415243308699007

M5	M5-BAIC (RMSE)	M5-BAIC (MAE)	M5-BAIC (sMdAPE)	M5-BAIC (MAPE)	M5-BAIC (LAR)
RCV Average	20.5759123372757	14.2113507578813	32.8681337086692	50.023554489658	0.402491996935441

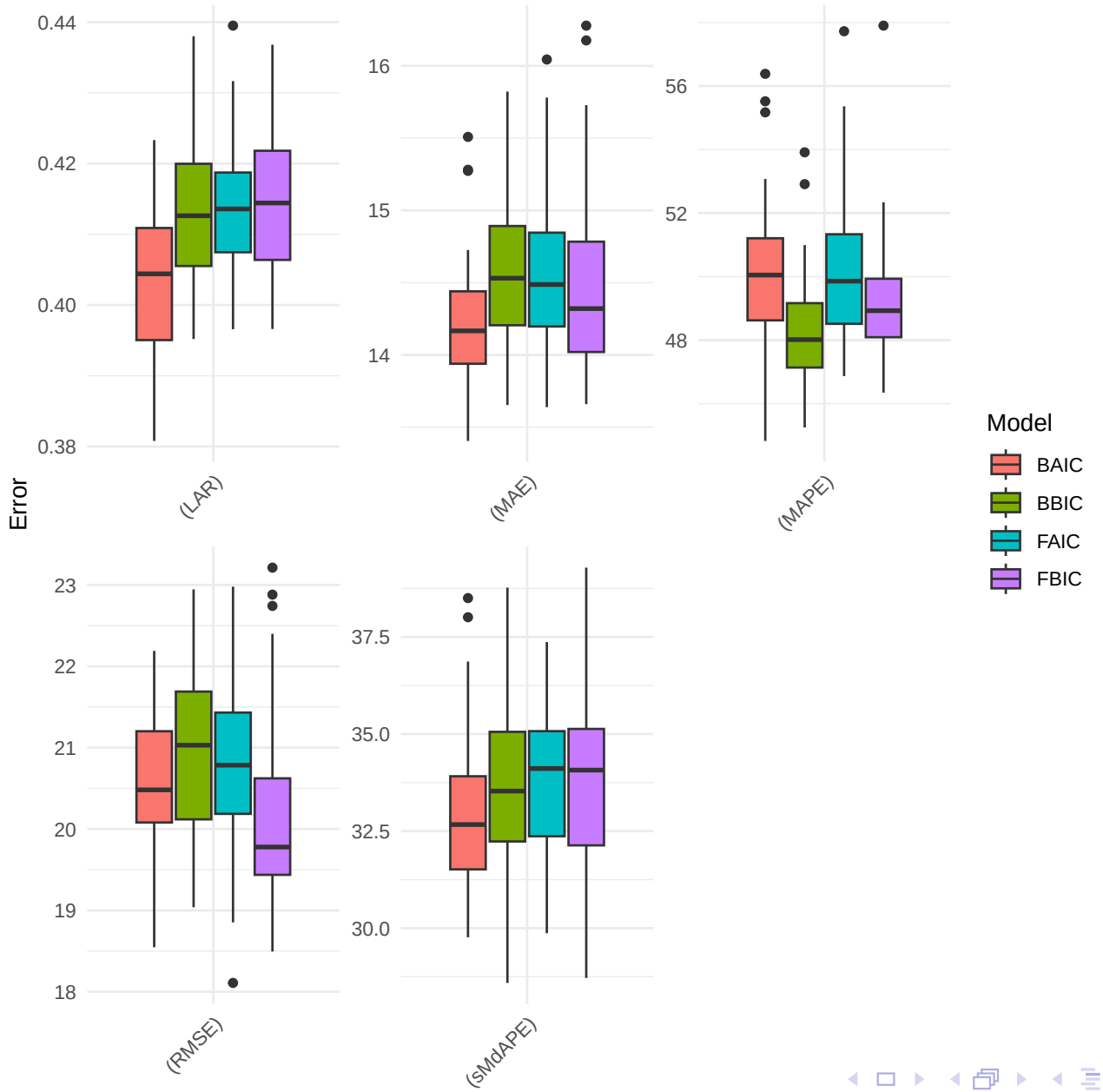
M5	M5-BBIC (RMSE)	M5-BBIC (MAE)	M5-BBIC (sMdAPE)	M5-BBIC (MAPE)	M5-BBIC (LAR)
RCV Average	21.008066759466	14.5744690252833	33.8128440882392	48.2419010016839	0.412924164357413

Repeated cross-validation - 95% CI for forward-AIC

based on 50 repeats



Repeated cross-validation - Box-Plots based on 50 repeats



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

- Regarding **LAR**, **FBIC** exhibits a lower median error compared to **FAIC**, while **BAIC** shows a lower median error than **BBIC**, indicating a better performance of the AIC criterion in the backward selection approach for this metric.

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

- Regarding **LAR**, **FBIC** exhibits a lower median error compared to **FAIC**, while **BAIC** shows a lower median error than **BBIC**, indicating a better performance of the AIC criterion in the backward selection approach for this metric.
- In terms of **MAE**, **FBIC** has a lower median error compared to **FAIC**, indicating better performance with forward selection using BIC. Conversely, **BAIC** has a lower median error than **BBIC**, suggesting better performance with backward selection using AIC.

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

- Regarding **LAR**, **FBIC** exhibits a lower median error compared to **FAIC**, while **BAIC** shows a lower median error than **BBIC**, indicating a better performance of the AIC criterion in the backward selection approach for this metric.
- In terms of **MAE**, **FBIC** has a lower median error compared to **FAIC**, indicating better performance with forward selection using BIC. Conversely, **BAIC** has a lower median error than **BBIC**, suggesting better performance with backward selection using AIC.
- With respect to **MAPE**, both **FBIC** and **BBIC** present lower median errors than **FAIC** and **BAIC**, respectively, which could imply that the BIC criterion is more effective in both forward and backward selection methods for minimizing MAPE.

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

- Regarding **LAR**, **FBIC** exhibits a lower median error compared to **FAIC**, while **BAIC** shows a lower median error than **BBIC**, indicating a better performance of the AIC criterion in the backward selection approach for this metric.
- In terms of **MAE**, **FBIC** has a lower median error compared to **FAIC**, indicating better performance with forward selection using BIC. Conversely, **BAIC** has a lower median error than **BBIC**, suggesting better performance with backward selection using AIC.
- With respect to **MAPE**, both **FBIC** and **BBIC** present lower median errors than **FAIC** and **BAIC**, respectively, which could imply that the BIC criterion is more effective in both forward and backward selection methods for minimizing MAPE.
- For **RMSE**, **FBIC** shows a lower median error than **FAIC**, while **BAIC** shows a lower median error than **BBIC**.

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

- Compare the previous models to F-P and B-P (use the p-value threshold that was discussed in class).

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

- Compare the previous models to F-P and B-P (use the p-value threshold that was discussed in class).

We can just add the following code to the `KfoldCVPerf.stepwise()`:

```
### ... ###
} else if (method == "forward" && criterion == "p_val") {
    # Use the global temporary variable for model fitting
    LM <- lm(formula, data = train_folds_temp)
    model.Fp <- ols_step_forward_p(LM, p_val=p) #def p as 0.005 (input)
    model <- model.Fp$model
} else if (method == "backward" && criterion == "p_val") {
    # Use the global temporary variable for model fitting
    LM <- lm(formula, data = train_folds_temp)
    model.Bp <- ols_step_backward_p(LM, p_val=p) #0.005
    model <- model.Bp$model
} ### ... same code ... ###
```

Repeated cross-validation - based on 50 repeats (Fp; Bp)

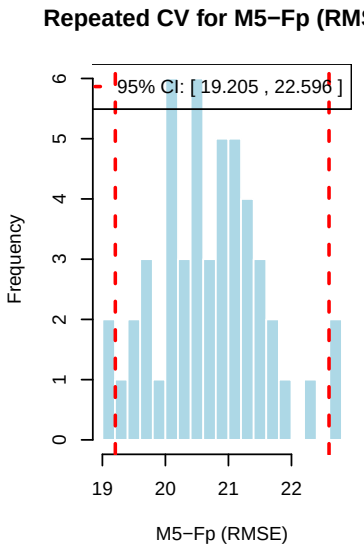
```
M5_Fp.RCV
M5_Bp.RCV
>>
```

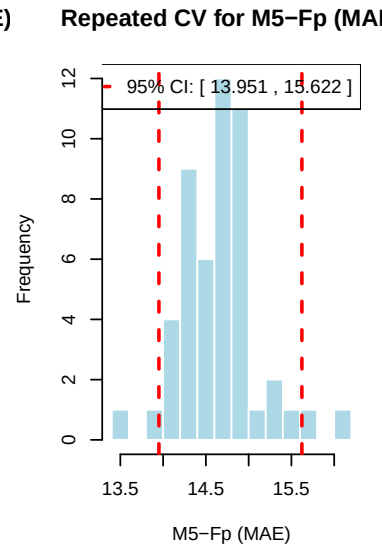
M5	M5-Fp (RMSE)	M5-Fp (MAE)	M5-Fp (sMdAPE)	M5-Fp (MAPE)	M5-Fp (LAR)
RCV Average	20.7095116438382	14.6475772954767	34.5658151361359	52.4466762438904	0.416701296907865

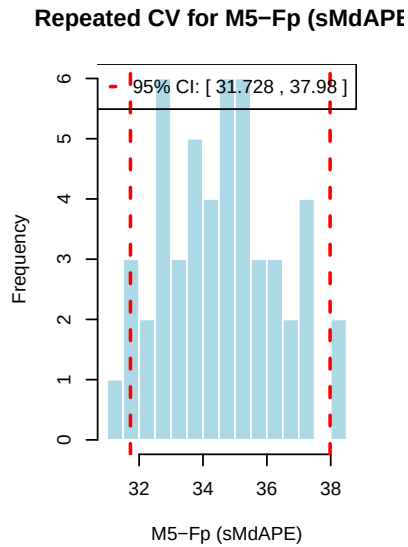
M5	M5-Bp (RMSE)	M5-Bp (MAE)	M5-Bp (sMdAPE)	M5-Bp (MAPE)	M5-Bp (LAR)
RCV Average	20.605690303201	14.3281856453847	33.2056685005456	50.2603883008135	0.406934985132059

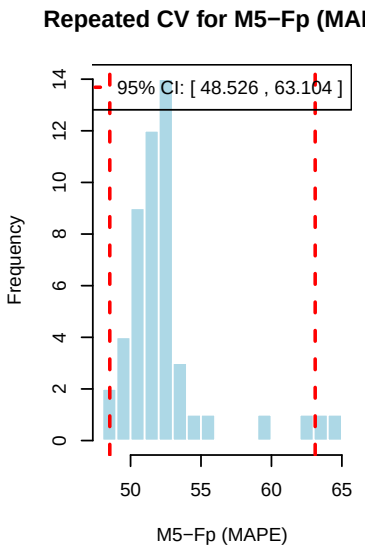
Repeated cross-validation - 95% CI for forward-p-value

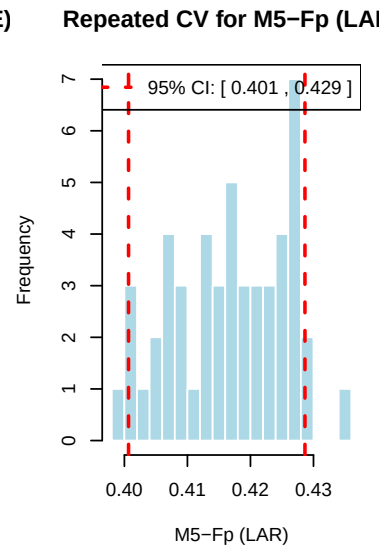
based on 50 repeats





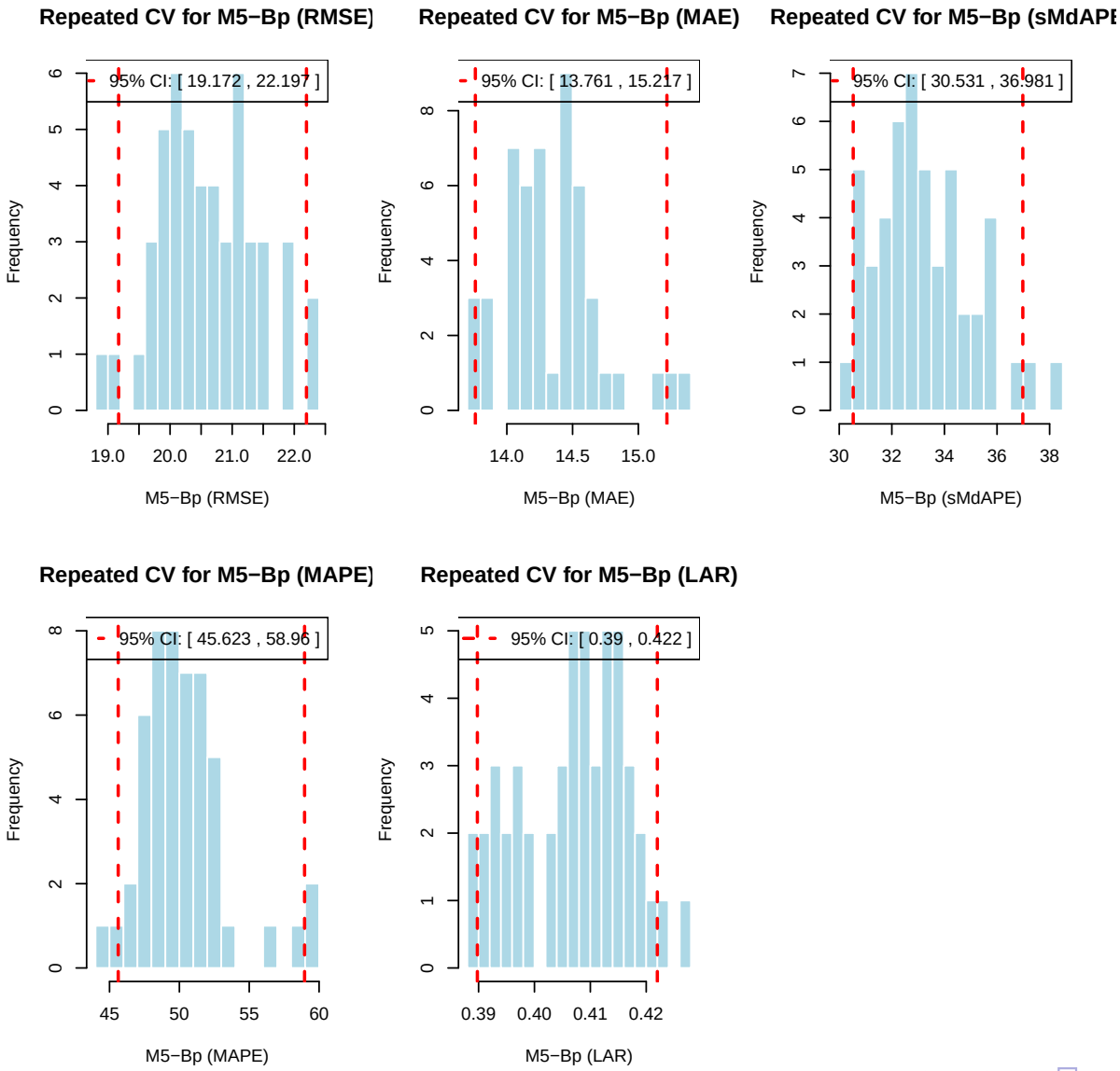




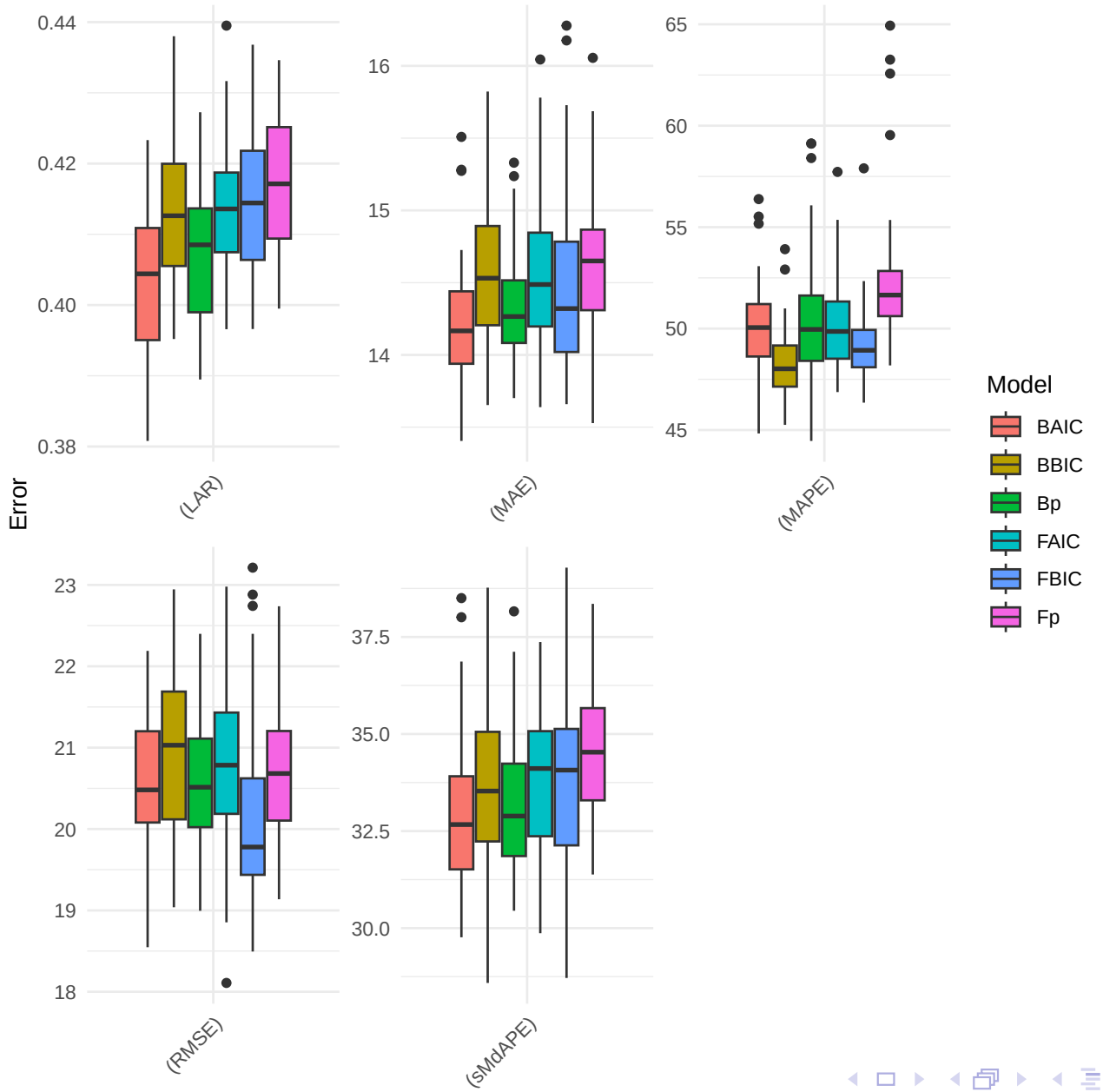


Repeated cross-validation - 95% CI for backward-p-value

based on 50 repeats



Repeated cross-validation - Box-Plots based on 50 repeats



LASSO and Ridge regression

- Finally estimate M5 using LASSO and ridge with lambda tuned via cross-validation. Compare against the previous models.

LASSO and Ridge regression

- Finally estimate M5 using LASSO and ridge with lambda tuned via cross-validation. Compare against the previous models.

Again, we can modify the `KfoldCVPerf.stepwise()` function in order to get the performance for penalized models like Ridge and LASSO:

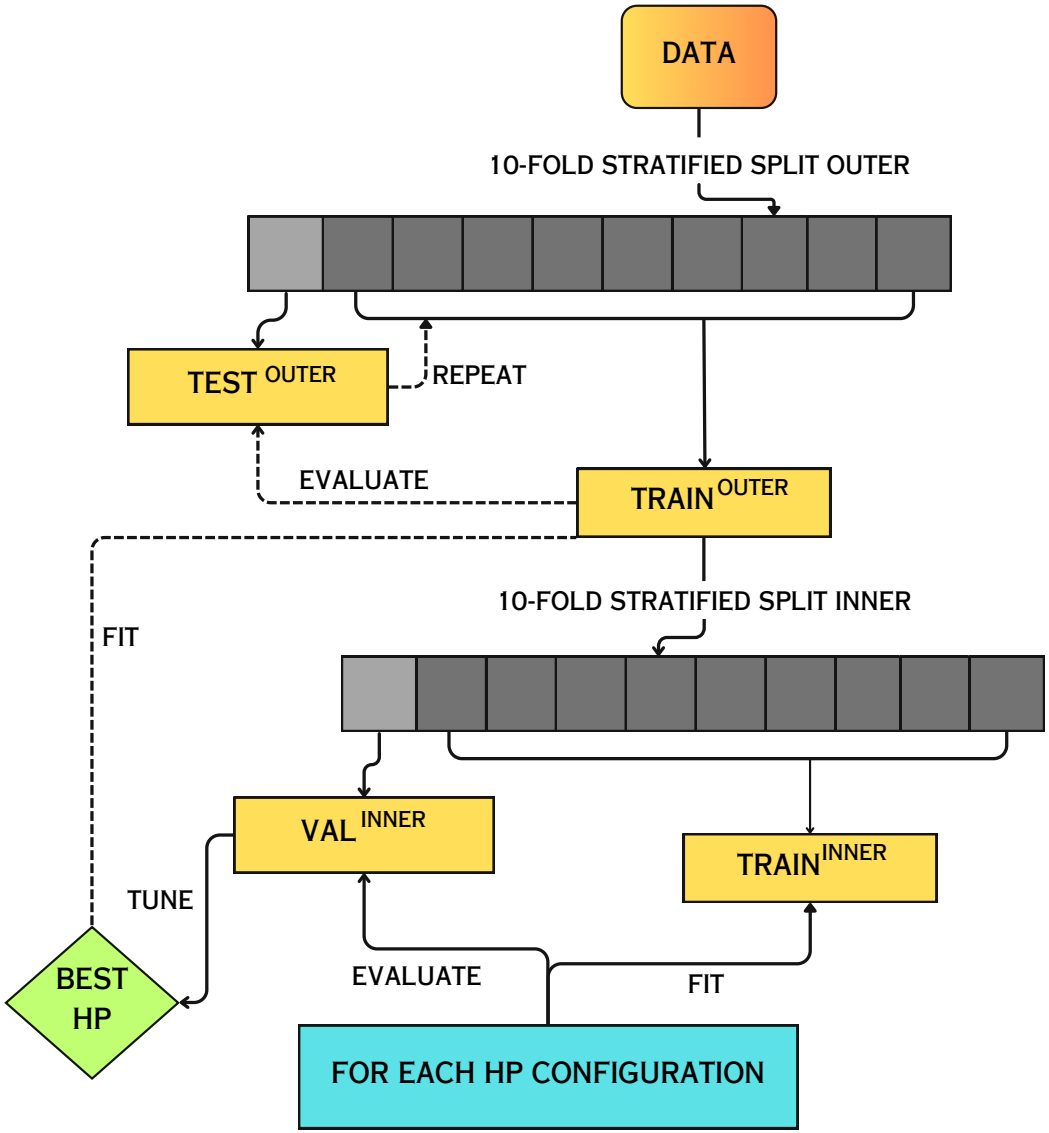
```
### ... same code ... ###
} else if (method == "ridge") {
    require(doMC)
    set.seed(42)
    registerDoMC(cores = 2) #parallel backend
    system.time(
      model <- cv.glmnet(
        x=model.matrix(formula, data),
        y=as.vector(data[y])[[1]],
        type.measure="mse", alpha=0, nfolds=10, parallel = TRUE))
} else if (method == "lasso") {
    require(doMC)
    set.seed(42)
    registerDoMC(cores = 2) #parallel backend
    system.time(
      model <- cv.glmnet(
        x=model.matrix(formula, data),
        y=as.vector(data[y])[[1]],
        type.measure="mse", alpha=1, nfolds=10, parallel = TRUE))
} else {
    stop("Unsupported method or criterion.")
    }
```


LASSO and Ridge regression

```
        rm(train_folds_temp, envir = .GlobalEnv) #remove globenv
        # Check for nestedcv (ridge; lasso)
        if (method == "ridge" || method == "lasso") {
            preds <- as.vector(predict(model,
                               newx = model.matrix(formula, val_fold),
                               s = "lambda.min"))
        } else {
            preds <- predict(model, newdata = val_fold)
        }
        y_val <- inv.trans(y_val)
        preds <- inv.trans(preds)
        ### ... same code ... ###

# Repeated CV for Ridge and LASSO
M5_ridge.RCV <- RepeatedCV.stepwise(
    repeats = 50,
    K = 10,
    data = data_,
    method='ridge',
    formula = M5_formula,
    model_name = 'M5-Ridge',
    inv.trans = function(y) exp(y))
M5_lasso.RCV <- RepeatedCV.stepwise(
    repeats = 50,
    K = 10,
    data = data_,
    method='lasso',
    formula = M5_formula,
    model_name = 'M5-LASSO',
    inv.trans = function(y) exp(y))
```


Nested cross-validation; tune `lambda` for Ridge, LASSO

$\min_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} \frac{1}{2N}\sum_{i=1}^{N}(y_i-\beta_0-x_i^T\beta)^2+\lambda\left[(1-a)\|\beta\|_2^2/2+a\|\beta\|_1\right], a=0,1.$



Repeated cross-validation for LASSO and Ridge

```
M5_ridge.RCV
M5_lasso.RCV
>>
```

M5	M5-Ridge (RMSE)	M5-Ridge (MAE)	M5-Ridge (sMdAPE)	M5-Ridge (MAPE)	M5-Ridge (LAR)
RCV Average	18.2001363127757	13.0288544659089	30.9505759289211	42.9486625659833	0.374310211087681

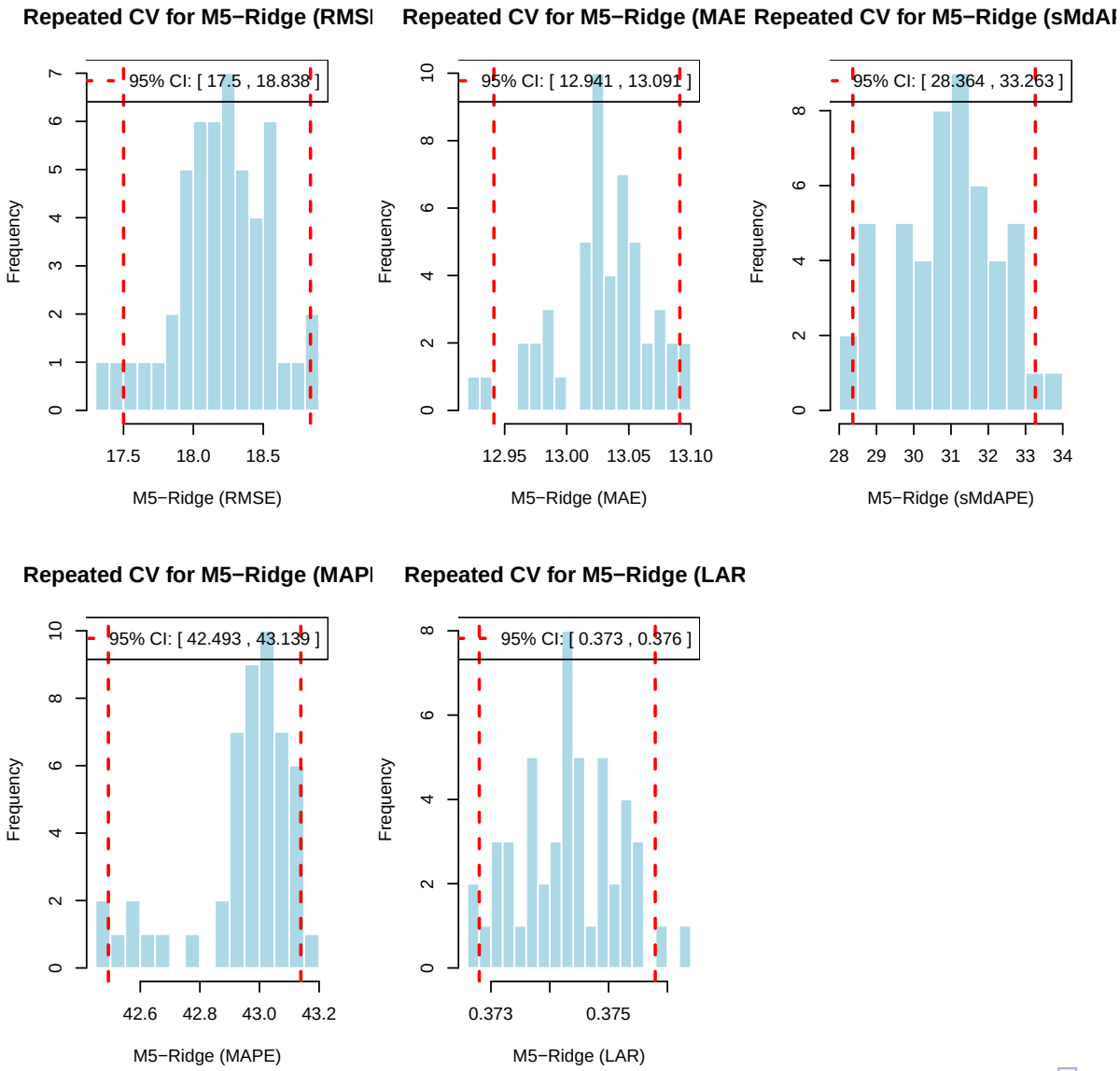
M5	M5-LASSO (RMSE)	M5-LASSO (MAE)	M5-LASSO (sMdAPE)	M5-LASSO (MAPE)	M5-LASSO (LAR)
RCV Average	18.5161292116031	13.2162243476291	31.2672848724556	44.2980742675429	0.378301429968909

- **Ridge Regression** demonstrates superior performance compared to all previously evaluated models.
- **Lasso Regression** surpasses the earlier models as well, yet **Ridge** is preferred for its slightly lower errors.

FORTH

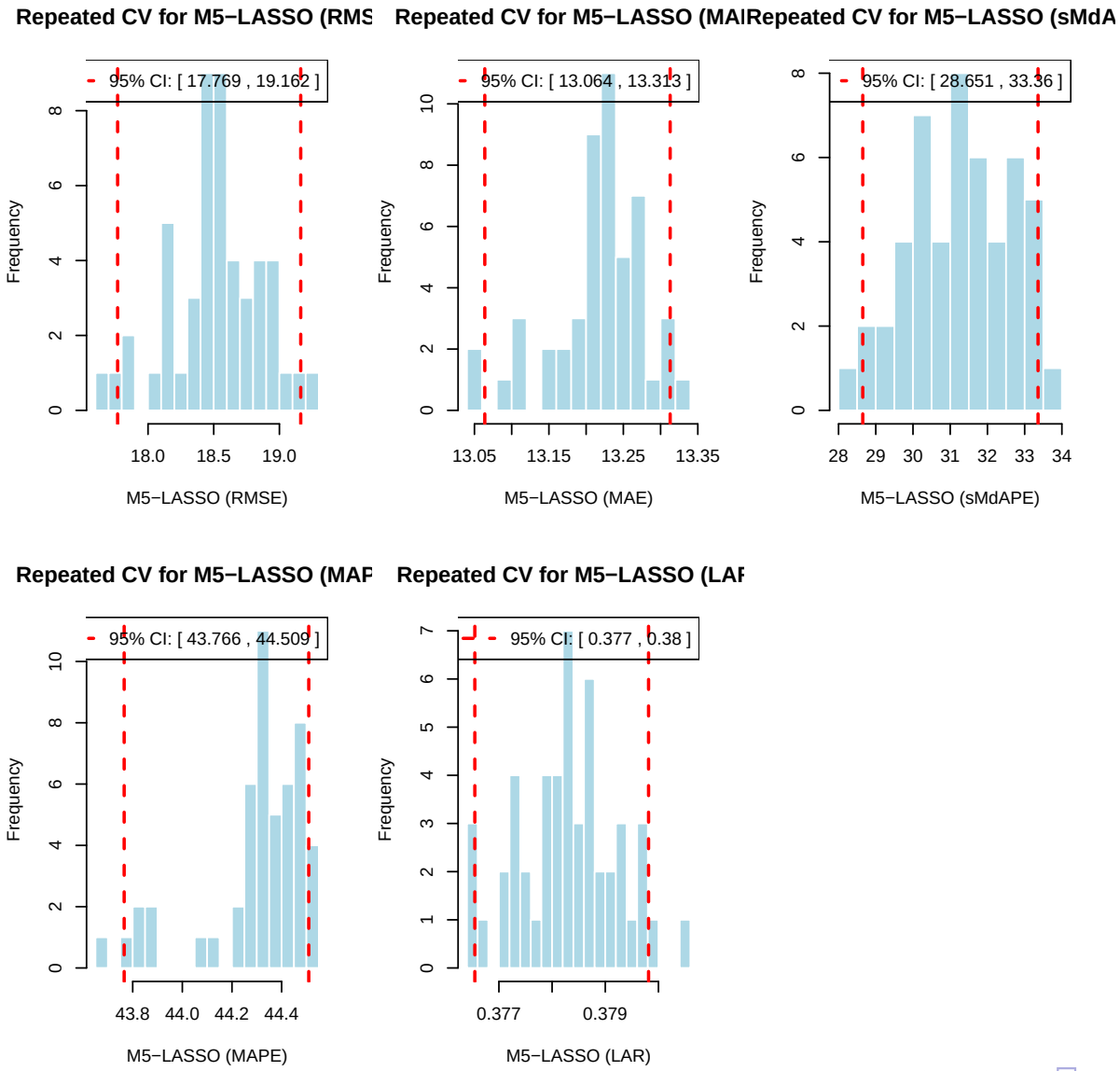
Repeated cross-validation - 95% CI for Ridge

based on 50 repeats

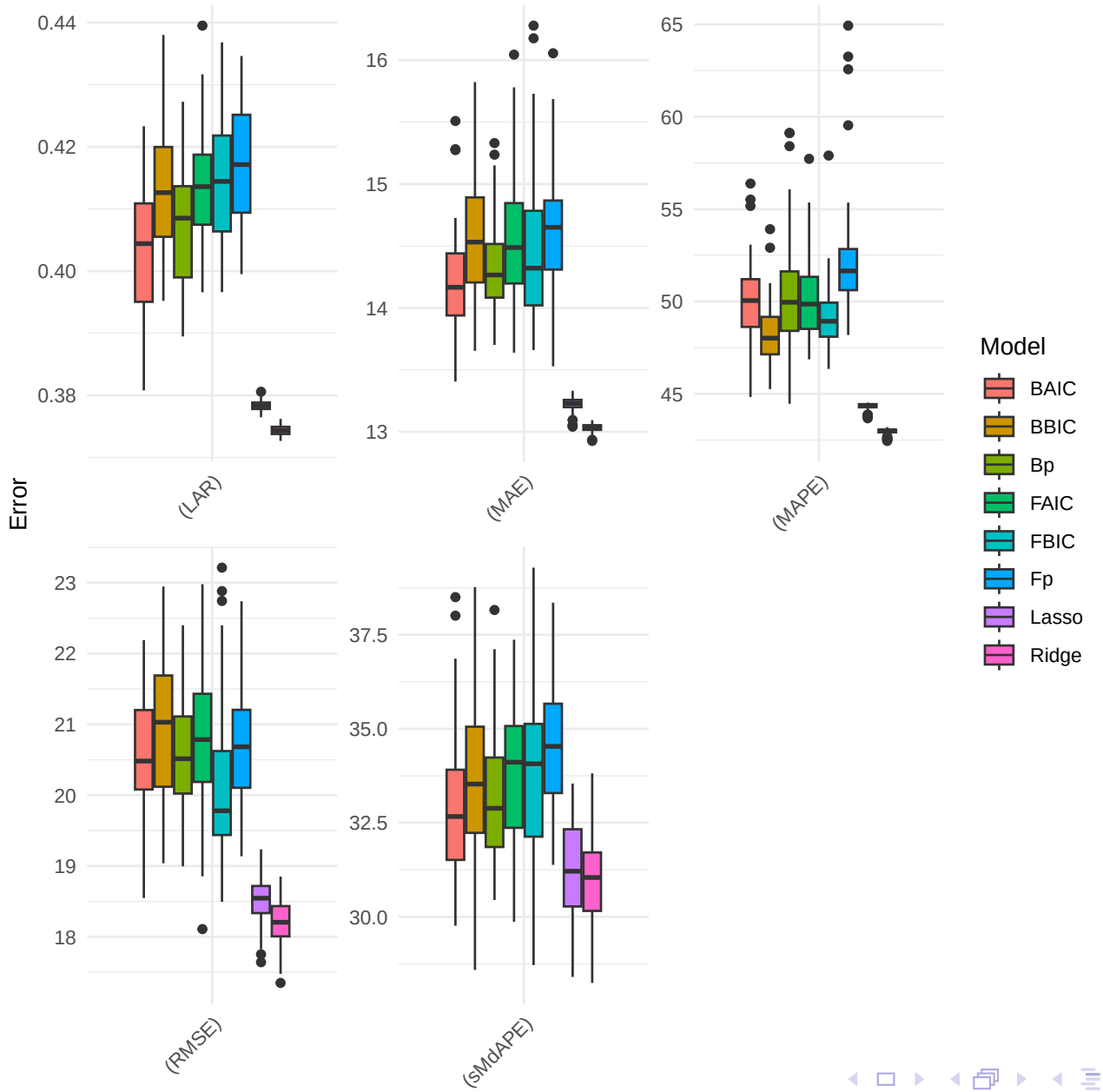


Repeated cross-validation - 95% CI for LASSO

based on 50 repeats



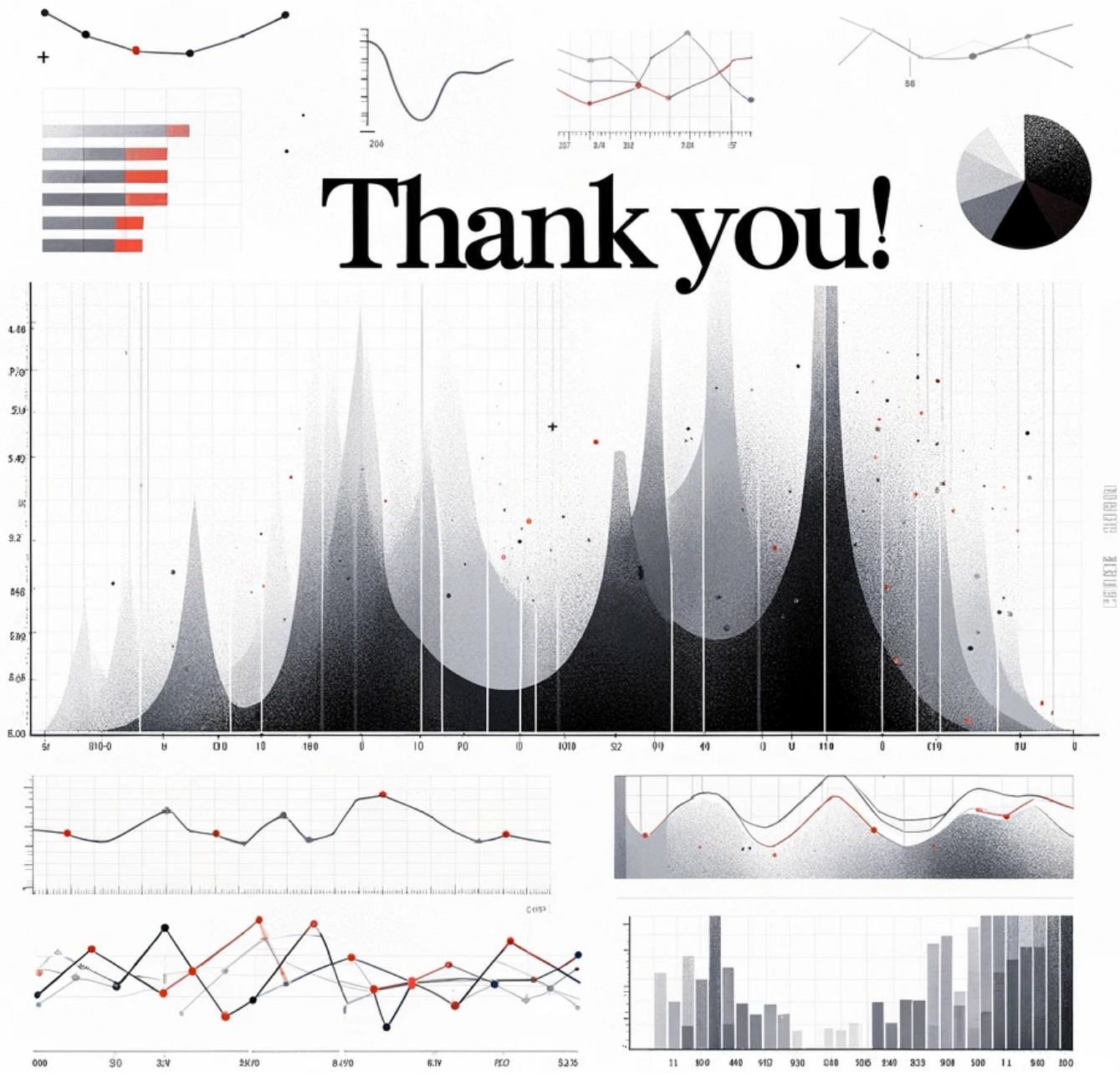
Repeated cross-validation - Box-Plots based on 50 repeats



Overall results based on repeated cross-validation

Model	RMSE	MAE	sMdAPE	MAPE	LAR
M3 RCV Average	20.7288	16.1223	33.8353	67.7265	0.4383
M4 RCV Average	18.5804	13.8479	34.8101	53.5018	0.4506
M5 RCV Average	18.7312	13.5512	33.5723	49.2004	0.4124
M5-FAIC RCV Average	20.7875	14.5760	33.7483	50.2182	0.4142
M5-FBIC RCV Average	20.1684	14.4789	33.7137	49.1706	0.4152
M5-BAIC RCV Average	20.5759	14.2114	32.8681	50.0236	0.4025
M5-BBIC RCV Average	21.0081	14.5745	33.8128	48.2419	0.4129
M5-Fp RCV Average	20.7095	14.6476	34.5658	52.4467	0.4167
M5-Bp RCV Average	20.6057	14.3282	33.2057	50.2604	0.4069
M5-Ridge RCV Average	18.2001	13.0289	30.9506	42.9487	0.3743
M5-LASSO RCV Average	18.5161	13.2162	31.2673	44.2981	0.3783

Table: Repeated Cross-validation average performance for different models


Thank you!