Exploring and Predicting Ozone Levels using the Airquality Dataset with the help of Multiple Imputation and Regularization Techniques

Course: 16:954:567:01 Statistical Modeling and Computing

Team Members :

Prashanth Aripirala

Anirudh Gaur

Rohit Macherla

Rishik Shekar Salver

Department of Statistics & Data Science

Rutgers University

New Brunswick, NJ

Supervised by

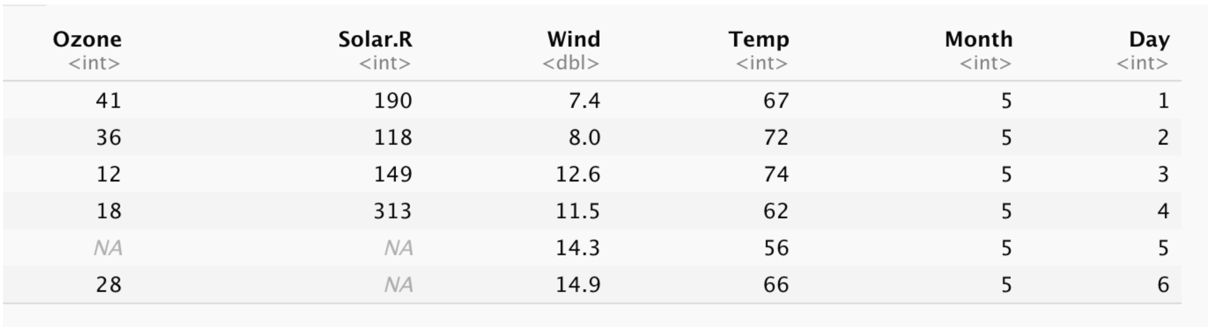
Prof. Jack Mardekian

ABSTRACT

Utilizing data on air quality, this project investigates and predicts ozone levels. Using a variety of imputation and regularization strategies, this study tries to build and compare models that estimate and predict ozone levels. To deal with missing data, the dataset underwent preliminary processing. With the help of the MICE package in R, multiple imputations were used to fill in the missing values. A total of four models were created: a linear regression model with missing values, an imputed linear regression model created using the mice package, an elastic net model, and a bootstrapped linear regression model. Based on their success in predicting ozone levels, the models were assessed. However, the imputed model also turned out to be a good fit. The results showed that the elastic net model had the least multicollinearity among the predictors.

INTRODUCTION

Air pollution is a significant problem worldwide, with both environmental and health impacts. One of the most harmful pollutants is ozone, which can cause respiratory problems and other health issues. It is crucial to monitor and understand the factors that contribute to ozone formation to develop effective strategies for reducing its levels. In this context, statistical models can be used to analyze the relationships between ozone levels and other air pollutants. The airquality dataset (Fig. 1) provides a valuable resource for studying the impacts of various factors on ozone levels in New York City. The dataset contains measurements of air pollutants such as ozone, solar radiation, wind, and temperature collected between May and September 1973 in New York City. This project aims to explore and predict ozone levels using this dataset and compare different modeling techniques for this purpose. Specifically, multiple imputation and regularization techniques are used to develop and evaluate different models based on their ability to predict ozone levels accurately. The results of this study can provide insights into the factors that influence ozone levels and help develop effective strategies for reducing their concentrations.



Ozone	Solar.R	Wind	Temp	Month	Day
<int>	<int>	<dbl>	<int>	<int>	<int>
41	190	7.4	67	5	1
36	118	8.0	72	5	2
12	149	12.6	74	5	3
18	313	11.5	62	5	4
NA	NA	14.3	56	5	5
28	NA	14.9	66	5	6

Fig. 1

MATERIALS AND METHODS

Data Cleaning

In the initial stage of the analysis, the Month and Day columns were removed from the airquality dataset as they were deemed redundant for the purpose. The modified dataset comprised four columns, namely Ozone, Solar Radiation, Wind, and Temperature. The dataset consisted of 153 observations, of which 44 contained missing values, primarily in the Ozone column.

Linear Regression without data imputation

Before proceeding with any data imputation, the dataset was divided into training and testing samples with a split of 70% and 30%, respectively. Subsequently, a linear regression model was fitted with Ozone as the response variable and the other columns as predictors to estimate and predict ozone levels. The quality of the model was evaluated by examining the residuals (Fig. 2.1), which were found to be skewed as observed from the Q-Q plot (Fig. 2.2). Furthermore, the Mean Squared Error (MSE) for the model was calculated, which was found to be 490 (Fig. 2.3).

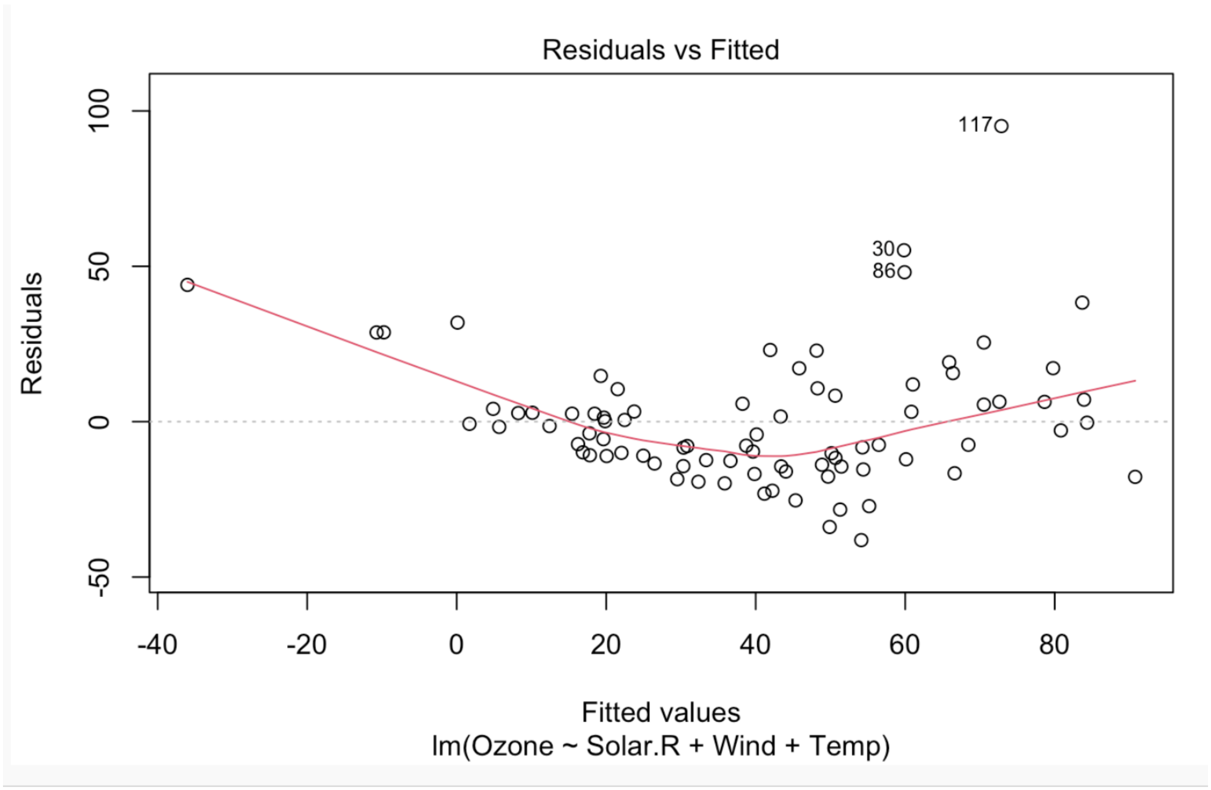

Residuals vs Fitted
Residuals
100
50
0
-50
117
30
86
-40
-20
0
20
40
60
80
Fitted values
lm(Ozone ~ Solar.R + Wind + Temp)

Fig. 2.1

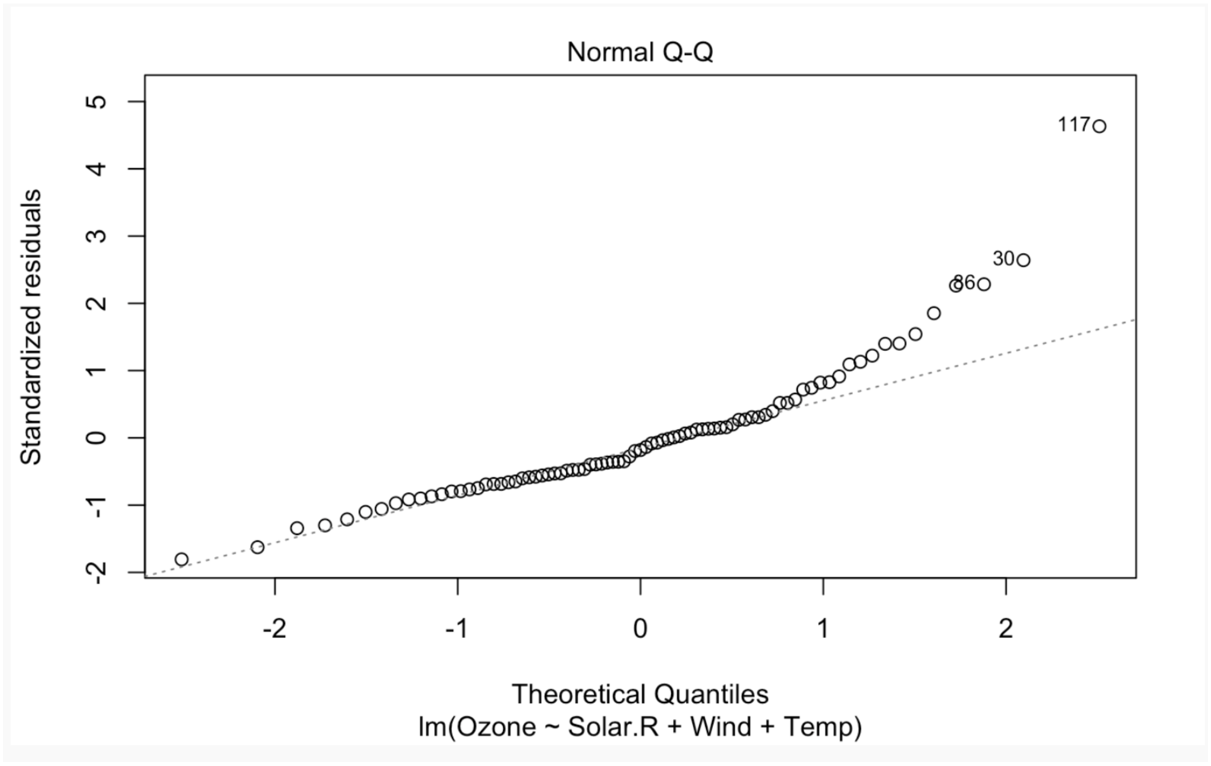
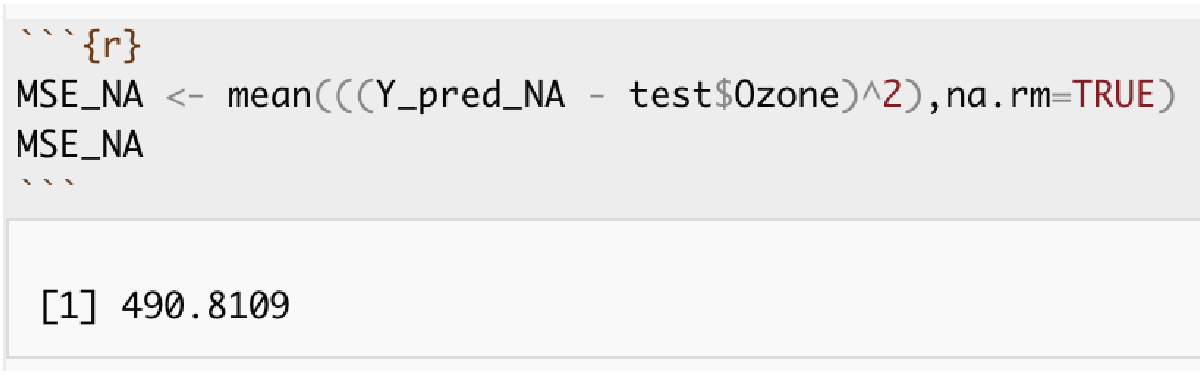

Normal Q-Q
Standardized residuals
5
4
3
2
1
0
-1
-2
117
30
86
-2
-1
0
1
2
Theoretical Quantiles
lm(Ozone ~ Solar.R + Wind + Temp)

Fig. 2.2



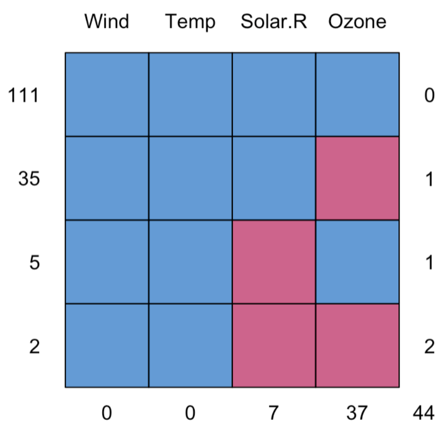
~~~
```{r}
MSE_NA <- mean(((Y_pred_NA - test$Ozone)^2),na.rm=TRUE)
MSE_NA
```
~~~

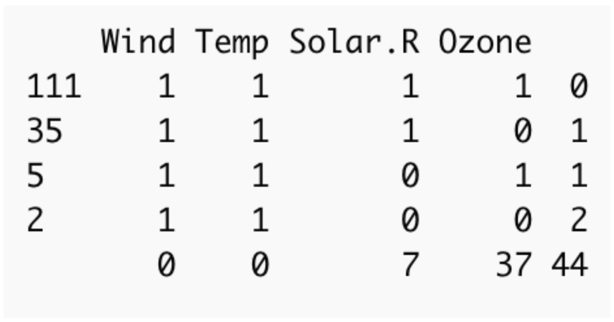
```
[1] 490.8109
```

Fig. 2.3

Finding Missing Values Type

After identifying the missing values in the dataset, the pattern of missing values was checked using the md.pattern() and sapply() functions. This allowed to identify which columns contained missing values and how many missing values were present in each column (Fig. 3). It was observed that the Ozone column had the highest number of missing values, followed by Solar Radiation and Wind columns.

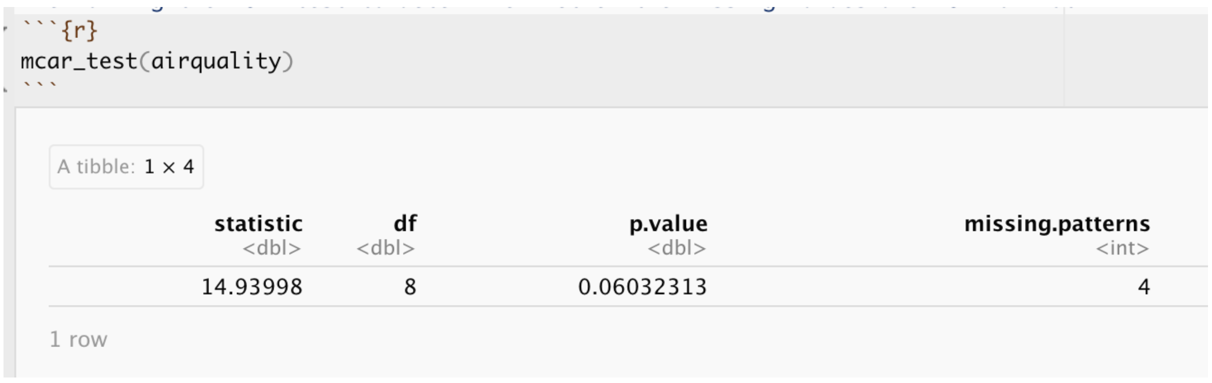




```
     Wind Temp Solar.R Ozone   
111     1    1       1     1  0
35      1    1       1     0  1
5       1    1       0     1  1
2       1    1       0     0  2
        0    0       7    37 44
```

Fig. 3

Next, a missing completely at random (MCAR) test was performed using the mcar_test() function. The null hypothesis of this test is that the missing data is MCAR, which means that the probability of missing data is the same for all observations and does not depend on any observed or unobserved variables. The alternative hypothesis is that the missing data is not MCAR. A p-value greater than 0.05 was obtained for the test (Fig. 4), which allowed to conclude that the missing data was MCAR.



```{r}
mcar_test(airquality)
```

```
A tibble: 1 × 4
   statistic      df      p.value   missing.patterns
       <dbl>   <dbl>        <dbl>              <int>
    14.93998       8   0.06032313                  4
1 row
```

Fig. 4

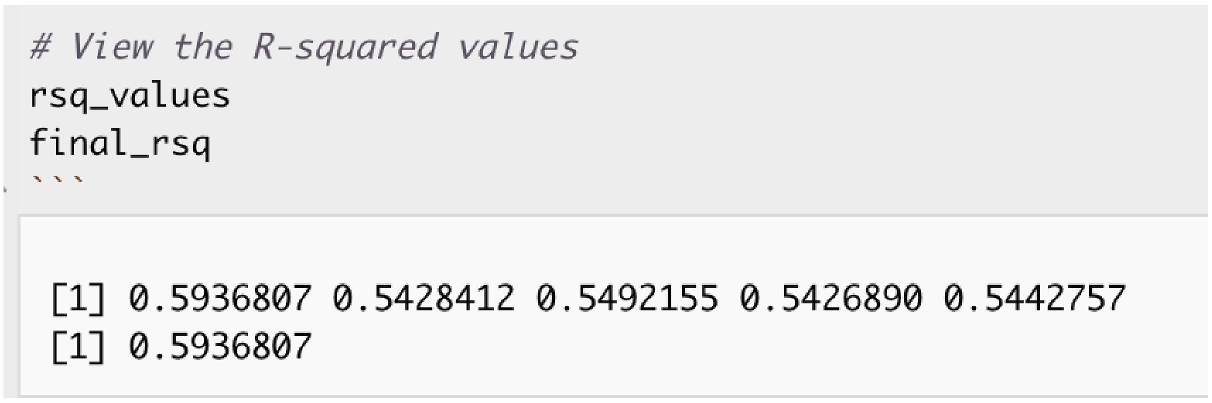
Data Imputation using MICE

After confirming that the missing data was MCAR, the multiple imputation by chained equations (MICE) method was used to impute the missing values. The mice() function was used with m=5 and maxit=50 to impute the missing values. The m parameter specifies the number of imputed datasets to generate, while the maxit parameter specifies the maximum number of iterations to run for each imputed dataset. The predictive mean matching (pmm) method was used for imputing the missing values. This method is based on matching the missing values to observed values that have similar values for the other variables.

After performing the imputation, 5 imputed datasets were obtained, each containing a different set of imputed values. The next step was to fit the same linear model to each of these imputed datasets using the lm() function.

To assess the goodness of fit of each model, the adjusted R^2 was calculated for each imputed dataset. Adjusted R^2 is a measure of how well the model fits the data, taking into account the number of variables in the model. It provides a more accurate measure of the model's goodness of fit than the regular R^2, especially when dealing with multiple predictor variables.

After calculating the adjusted R^2 for each model, the imputed dataset that corresponded to the highest adjusted R^2 was selected for further analysis (Fig. 5). This approach of selecting the imputed dataset with the highest adjusted R^2 is commonly used in multiple imputation studies, as it helps to ensure that the best imputed dataset is used for further analysis. The imputed dataset was then converted into a dataframe using the complete() function in R for further analysis.



```
# View the R-squared values
rsq_values
final_rsq
```

```
[1] 0.5936807 0.5428412 0.5492155 0.5426890 0.5442757
[1] 0.5936807
```

Fig. 5

Linear Regression with data imputation

Next, the dataset was split again into a training set and a testing set using a 70-30 split ratio. A linear model was fitted using the training dataset, and the residuals were plotted. It was observed from the Q-Q plot that the residuals were less skewed than before imputation (Fig. 6.1). To further evaluate the performance of the imputed dataset, the Mean Squared Error (MSE) for this model was calculated, which was found to be 437 (Fig. 6.2). This result showed a significant improvement compared to the initial MSE of 490 obtained before imputation, indicating that the imputation process had reduced the errors in the dataset.

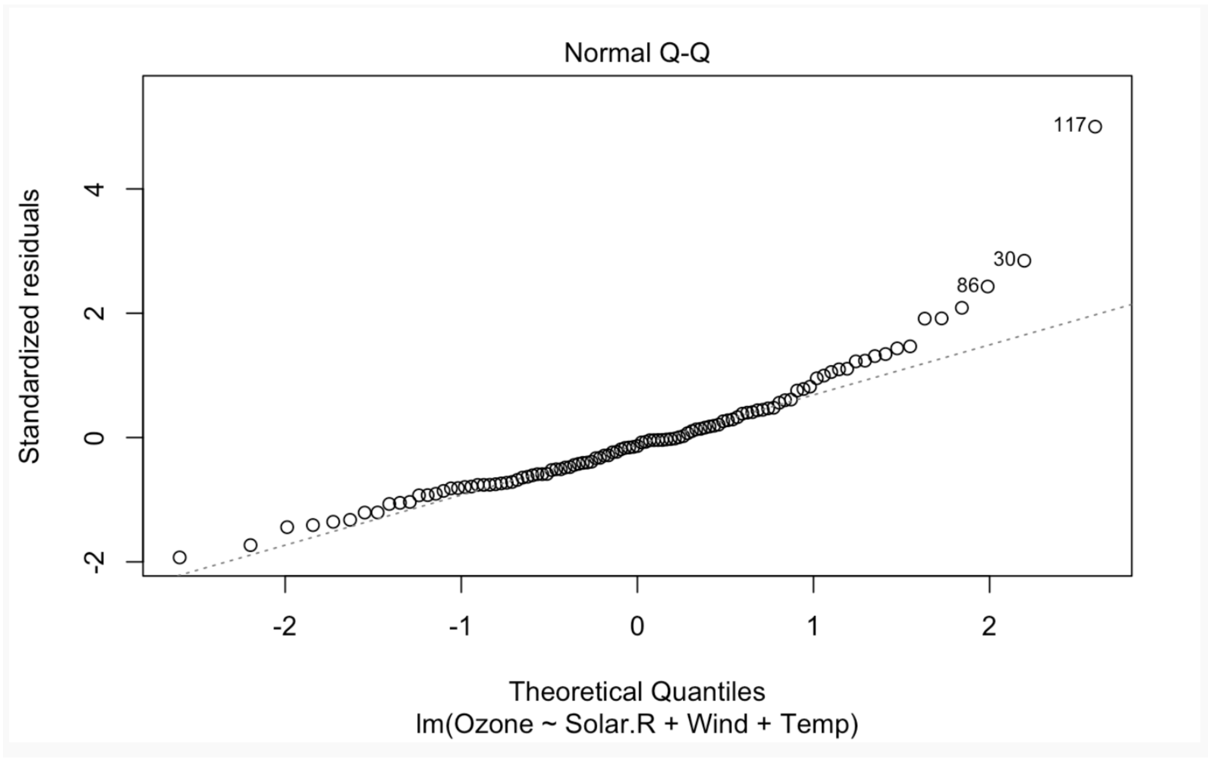
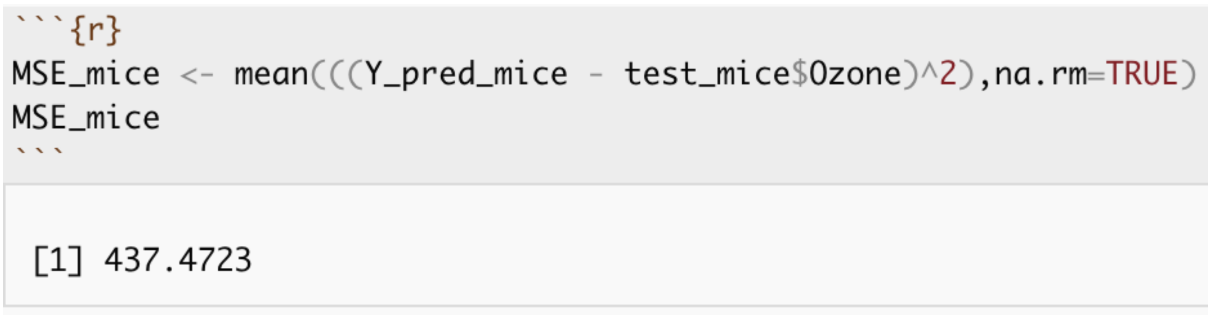


Fig. 6.1



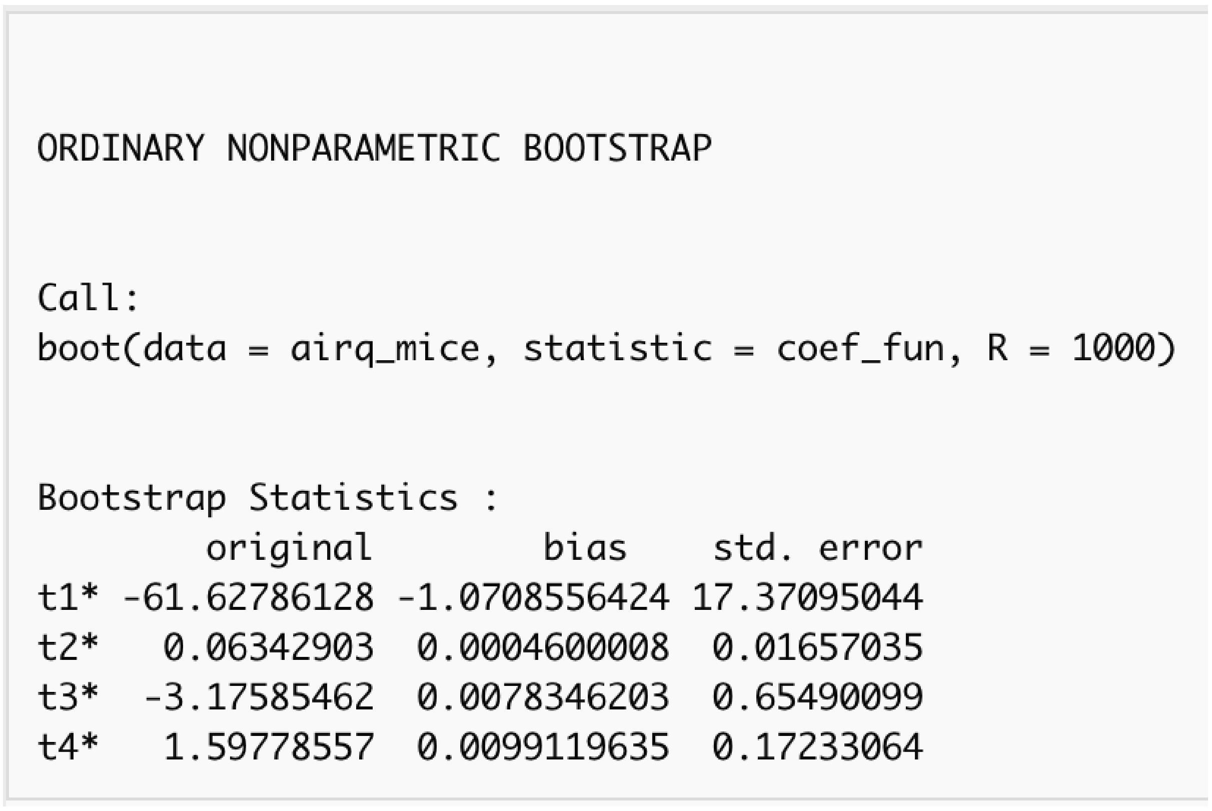
~~~
```{r}
MSE_mice <- mean(((Y_pred_mice - test_mice$Ozone)^2),na.rm=TRUE)
MSE_mice
```

[1] 437.4723
~~~

Fig. 6.2

Bootstrapping

After performing the imputation and fitting the linear model, bootstrapping was performed using the boot() function in R with 1000 replicates to further assess the accuracy of the dataset and to account for potential bias in the model.

Initially, some bias and standard error were observed in the intercept variable, whereas the estimates for the other predictors were reasonable. To obtain more accurate estimates, the bootstrap estimates were calculated using the colMeans() function. This function calculates the column means of a matrix or data frame, which allowed for obtaining the mean estimate of each of the predictors (Fig. 7).



```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = airq_mice, statistic = coef_fun, R = 1000)


Bootstrap Statistics :
        original          bias    std. error
t1* -61.62786128 -1.0708556424 17.37095044
t2*   0.06342903  0.0004600008  0.01657035
t3*  -3.17585462  0.0078346203  0.65490099
t4*   1.59778557  0.0099119635  0.17233064
```

Fig. 7

The bootstrap estimates were then used to calculate the 95% confidence intervals for each of the coefficients in the model. The confidence intervals were obtained using the boot.ci() function. The results showed that the confidence intervals for all the coefficients did not include zero, indicating that all the coefficients were statistically significant. Overall, the bootstrap analysis confirmed that the imputed dataset and linear model were accurate and reliable.

Next was to determine the confidence intervals for the estimates. 95% confidence interval was calculated for the intercept (t1) using the boot.ci() function. However, for the other predictors, the margin was very small, and the confidence interval did not make much sense to compute. Therefore, confidence interval was only obtained for the intercept.

After obtaining the confidence interval, bootstrap distributions was plotted for t1, t2, t3, and t4 using the plot.density() function (Fig. 8). The purpose of plotting these distributions was to visually represent the variability of the estimates and to show the spread of the data. 95% confidence interval line was also shown using the abline() function, which allowed us to see the range of the estimates that we were confident in.

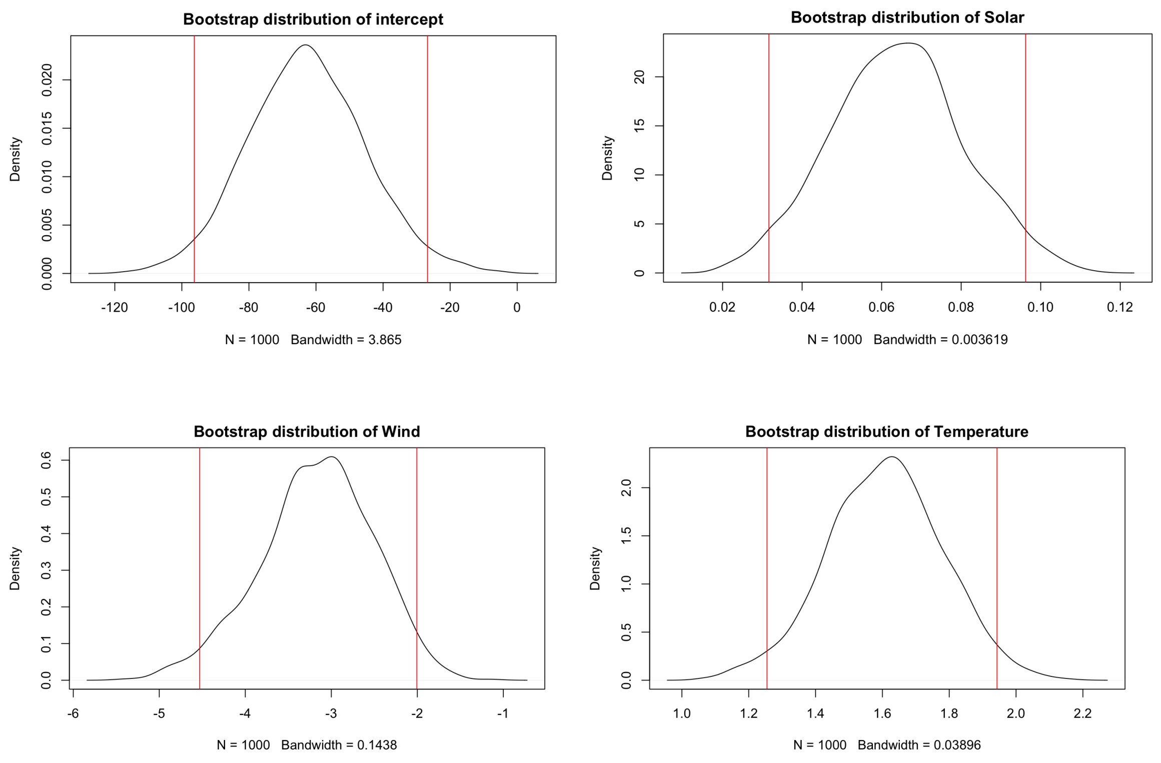


Fig. 8

Overall, the bootstrapping analysis was done to reduce bias and standard error and obtain more accurate estimates of the linear model. The visualization of the bootstrap distributions also allowed for seeing the variability of the estimates and determining the range of values that were confident in. After completing the bootstrapping analysis, elastic net regression was done to further investigate the relationship between ozone and other variables. Elastic net regression helps to identify the most important variables for predicting ozone levels while also addressing the issue of collinearity (Fig. 9) that often arises when multiple predictors are used in a linear model.

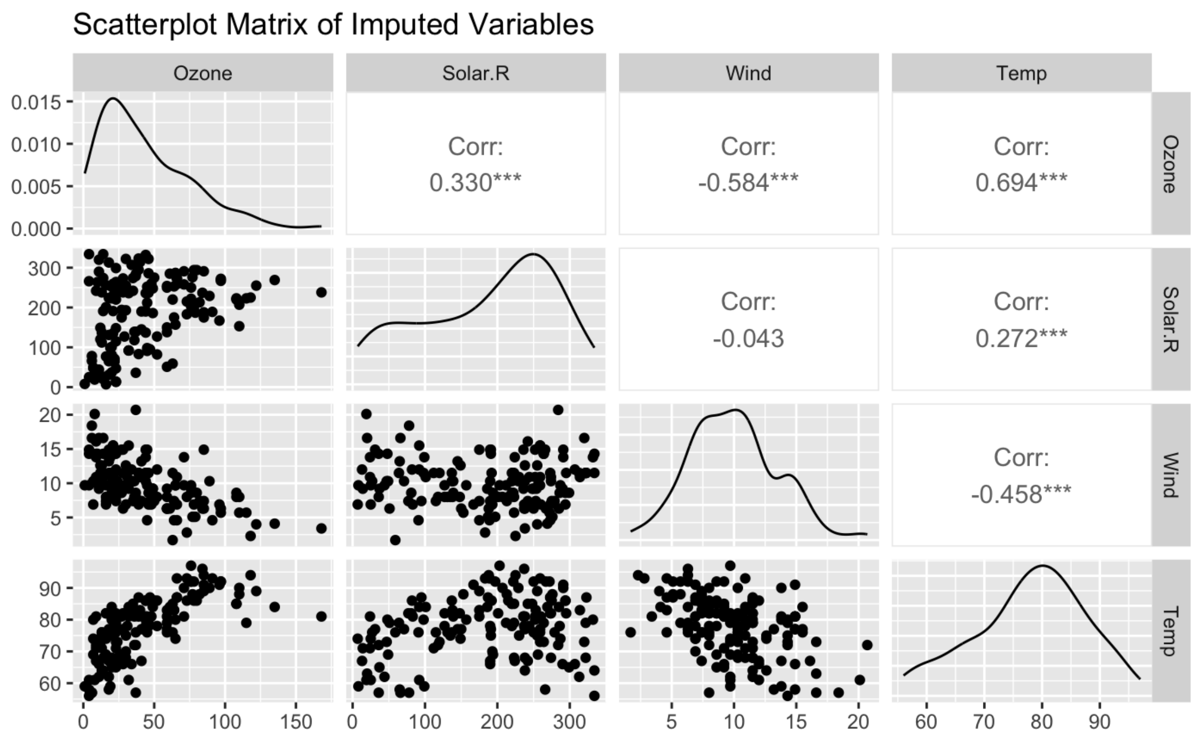


Fig. 9

Elastic Net Regression with Imputed Data

To perform elastic net regression, the predictor and response variables were created from the imputed dataset using the model.matrix() function in R. The data was then split into training and testing sets using a 70-30 split.

Next, a list of models was created using the cv.glmnet() function with different values of alpha ranging from 0 to 1 in increments of 0.05. Alpha is a tuning parameter that controls the balance between the L1 (Lasso) and L2 (Ridge) regularization methods used in elastic net regression. 10-fold cross-validation was used to select the optimal value of alpha that minimized the mean squared error (MSE) of the model.

The MSE and alpha values for each model were stored in a data frame and the relationship between alpha and MSE was plotted using ggplot (Fig. 10). The optimal value of alpha for the elastic net model was selected as the value that produced the lowest MSE, which was found to be 0.7.

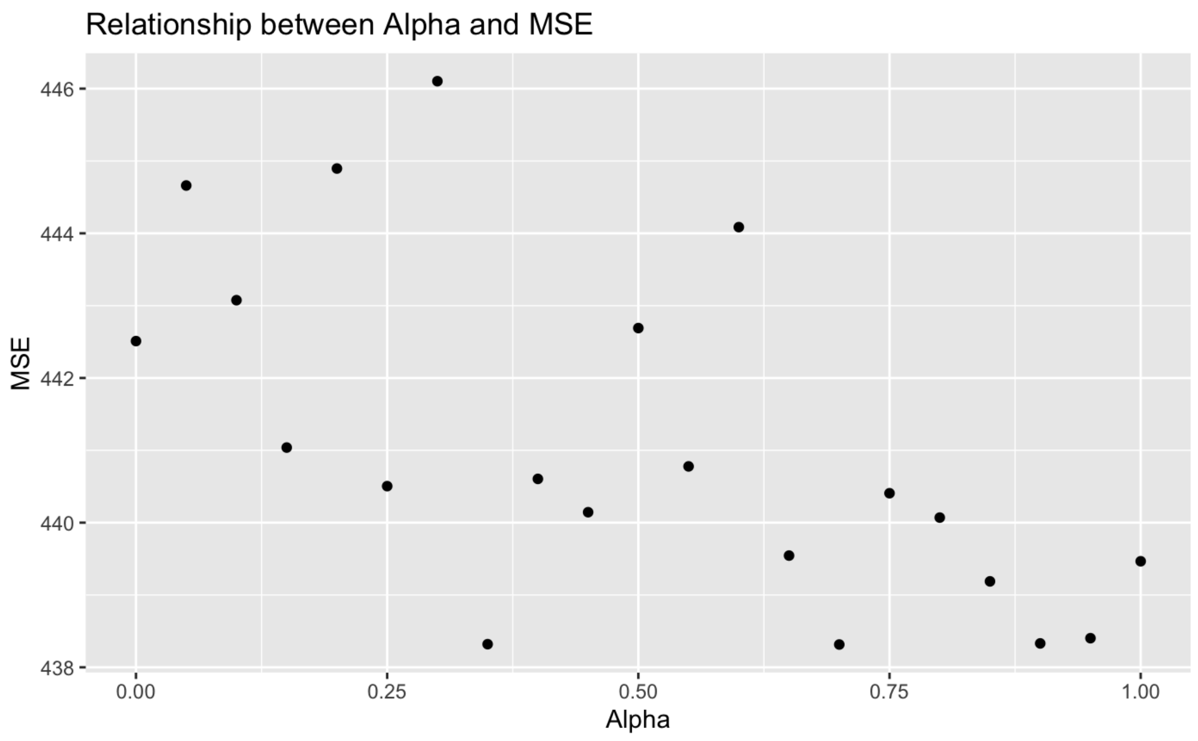


Fig. 10

After selecting the optimal alpha value through cross-validation, the elastic net model was fit to the training data using the glmnet() function. Next, the predict() function was used to obtain predicted ozone levels for the test data, and the mean squared error between the predicted and actual ozone levels was calculated to evaluate the model's performance.

Finally, the coef() function was used to obtain the coefficients for the elastic net model at the optimal alpha value. These coefficients represent the strength and direction of the relationship between each predictor variable and the ozone levels.

Using the elastic net regression, the problem of overfitting and multicollinearity was removed reduced yielding a good model.

CODE

```
#importing the necessary libraries
library(missForest)
library(Hmisc)
library(mice)
library(VIM)
library(rms)
library(naniar)
library(pool)
library(boot)
library(GGally)
library(glmnet)
library(caret)
library(dplyr)
#Import Data
data(airquality)
head(airquality)
#Removing the Month and Day columns as they are not useful to our analysis
airquality <- select(airquality, -Month, -Day)
head(airquality)
# split the data into train and test
set.seed(123)
sample <- sample(nrow(airquality), nrow(airquality)*0.7)
train <- airquality[sample,]
test <- airquality[-sample,]
#Plotting a linear model to this dataset with Ozone as a response and all the other variables as
predictors
airq_NA <- lm(Ozone ~ Solar.R + Wind + Temp, data=train)
summary(airq_NA)
residuals_NA <- as.data.frame(test$Ozone - predict(airq_NA,newdata = test))
Y_pred_NA <- predict(airq_NA,newdata = test)
Y_pred_NA
MSE_NA <- mean(((Y_pred_NA - test$Ozone)^2),na.rm=TRUE)
MSE_NA
plot(airq_NA)
#Checking the pattern of the missing values
md.pattern(airquality)
#Counting the mssing values in each column
sapply(airquality, function(x)sum(is.na(x)))
```

```r
#Performing the MCAR test to determine whether the missing values are MCAR or not
mcar_test(airquality)
#As the p-value is greater than 0.05, we can say that the missing values are MCAR(Missing
completely at Random)
#Imputing the missing values using MICE with m=5 and using PMM
air_mice <- mice(airquality, m = 5, maxit = 50, meth = "pmm")
models <- with(air_mice, lm(Ozone ~ Solar.R + Wind + Temp))
results <- pool(models)
summary(results)
#Finding the estimates of all the 5 imputed models
# Assuming that your response variable is named 'y' and your predictors are named 'x1', 'x2', etc.
# Also assuming that the MICE object is named 'mice_imputed'
models <- list() # Initialize an empty list to store the models
# Iterate through each completed dataset and fit a linear model
for(i in 1:5) {
  completed_data <- complete(air_mice, action = i)
  model <- lm(Ozone ~ Solar.R + Wind + Temp, data = completed_data)
  models[[i]] <- model # Store the model in the list
}
# View the results
#models
#Finding the adjusted R^2 for the 5 models
# Initialize an empty vector to store the R-squared values
rsq_values <- c()
final_rsq <- 0
#final_model
# Iterate through each model and extract the R-squared value
for(i in 1:5) {
  summary_model <- summary(models[[i]])
  rsq_values[i] <- summary_model$adj.r.squared
  if(rsq_values[i]>final_rsq){
    final_rsq<-rsq_values[i]
    final_model <- i
  }
}
```

```
# View the R-squared values
rsq_values
Final_rsq
#The model with highest Adjusted R square was picked was further analysis.
airq_mice <- complete(air_mice, action = final_model)
airq_mice <- as.data.frame(airq_mice)
airq_mice
#train test split of mice data
train_mice <- airq_mice[sample,]
test_mice <- airq_mice[-sample,]
#Plotting a linear model to this dataset with Ozone as a response and all the other variables as
predictors
airq_mice_lm <- lm(Ozone ~ Solar.R + Wind + Temp, data=train_mice)
summary(airq_mice_lm)
residuals_mice <- as.data.frame(test_mice$Ozone - predict(airq_mice_lm,newdata = test_mice))
Y_pred_mice <- predict(airq_mice_lm,newdata = test_mice)
Y_pred_mice
MSE_mice <- mean(((Y_pred_mice - test_mice$Ozone)^2),na.rm=TRUE)
MSE_mice
plot(airq_mice_lm)
#Finding correlation
ggpairs(airq_mice, columns = c("Ozone", "Solar.R", "Wind", "Temp"),
        title = "Scatterplot Matrix of Imputed Variables")
#Performing bootstrapping to estimsate bias and standard error
coef_fun <- function(data, indices) {
  fit <- lm(Ozone ~ Solar.R + Wind + Temp, data = data[indices,])
  coef(fit)
}
# Perform bootstrapping on the pooled results
set.seed(123)
boot_results <- boot(airq_mice, coef_fun, R = 1000)
# View the bootstrap results
boot_results
#Finding the estimates after bootstrapping
boot_coefs <- boot_results$t
```

```
boot.coef <- colMeans(boot_coefs)
boot.coef
#Finding the confidence intervals for t1,t2,t3,t4
set.seed(123)

# Generate bootstrap estimates
boot_coefs <- boot_results$t
boot.coef <- colMeans(boot_coefs)

# Calculate 95% confidence intervals
boot.ci(boot_results, type = "bca", index = c(1,2,3,4))
#We only see the CI for t1 here because for t2,t3,t4 there is very less variability so bootstrapping
couldn't provide meaningful confidence intervals.
# Calculate the bias-corrected and accelerated (BCa) confidence intervals
bc_ci <- boot::boot.ci(boot_results, type = "bca")

# Plot the bootstrap distribution of the intercept
plot(density(boot_coefs[,1]), main = "Bootstrap distribution of intercept")

# Add vertical lines at the 2.5th and 97.5th percentiles of the distribution
abline(v = quantile(boot_coefs[,1], c(0.025, 0.975)), col = "red")
# Calculate the bias-corrected and accelerated (BCa) confidence intervals
bc_ci <- boot::boot.ci(boot_results, type = "bca")

# Plot the bootstrap distribution of the intercept
plot(density(boot_coefs[,2]), main = "Bootstrap distribution of Solar")

# Add vertical lines at the 2.5th and 97.5th percentiles of the distribution
abline(v = quantile(boot_coefs[,2], c(0.025, 0.975)), col = "red")
# Calculate the bias-corrected and accelerated (BCa) confidence intervals
bc_ci <- boot::boot.ci(boot_results, type = "bca")

# Plot the bootstrap distribution of the intercept
plot(density(boot_coefs[,3]), main = "Bootstrap distribution of Wind")
```

```
# Add vertical lines at the 2.5th and 97.5th percentiles of the distribution
abline(v = quantile(boot_coefs[,3], c(0.025, 0.975)), col = "red")

# Calculate the bias-corrected and accelerated (BCa) confidence intervals
bc_ci <- boot::boot.ci(boot_results, type = "bca")

# Plot the bootstrap distribution of the intercept
plot(density(boot_coefs[,4]), main = "Bootstrap distribution of Temperature")

# Add vertical lines at the 2.5th and 97.5th percentiles of the distribution
abline(v = quantile(boot_coefs[,4], c(0.025, 0.975)), col = "red")

#Again only intercept CI is given as others dont have much variability.

# Generate predicted values for new data using bootstrap estimates
boot_preds <- apply(boot.coefs, 1, function(x) x[1] + x[2]*airq_mice$Solar.R + x[3]*airq_mice$Wind +
x[4]*airq_mice$Temp)
#boot_preds

# Calculate the 95% prediction interval
boot_pi <-quantile(boot_preds, c(0.025, 0.975), na.rm = TRUE)

# Print the prediction interval
boot_pi

library(ggplot2)
library(tidyr)

# Convert the boot_coefs matrix into a dataframe
df_boot_coefs <- as.data.frame(boot_coefs)

# Add a column for the row number (coefficient number)
df_boot_coefs$row <- rownames(df_boot_coefs)

# Convert the dataframe from wide to long format
```

```
df_boot_coefs_long <- pivot_longer(df_boot_coefs, -row, names_to = "bootstrap", values_to = "value")

# Plot the histogram with different colors for different coefficients
ggplot(df_boot_coefs_long, aes(x = value, fill = bootstrap)) +
  geom_histogram(bins = 20, color = "white", alpha = 0.7, position = "identity") +
  scale_fill_discrete(labels = c("Intercept", "Solar.R", "Wind", "Temp"),name = "Coefficient") +
  labs(x = "Bootstrap Coefficient Estimates", y = "Frequency",
       title = "Histogram of Bootstrap Distribution of Coefficients")
#The resulting plot shows the distribution of the bootstrap coefficient estimates for "t1", which can
provide information about the variability and uncertainty in the coefficient estimate.

#Performing elastic net regression to reduce the multicollinearity between the predictors.
X <- model.matrix(Ozone ~ ., data = airq_mice)[,-1]
Y <- airq_mice$Ozone
X_train_elnet <- X[sample, ]
Y_train_elnet <- Y[sample]
X_test_elnet <- X[-sample, ]
Y_test_elnet <- Y[-sample]
models <- list()
for (i in 0:20) {
  name <- paste0("alpha", i/20)
  models[[name]] <- cv.glmnet(X_train_elnet, Y_train_elnet, type.measure="mse",
alpha=i/20,family="gaussian")
}
results <- data.frame()
for (i in 0:20) {
name <- paste0("alpha", i/20)
## Use each model to predict 'y' given the Testing dataset
Y_pred_CV <- predict(models[[name]],
s=models[[name]]$lambda.min, newx=X_test_elnet)
## Calculate the Mean Squared Error...
mse <- mean((Y_pred_CV - Y_test_elnet)^2)
## Store the results
temp <- data.frame(alpha=i/20, mse=mse, name=name)
results <- rbind(results, temp)
}

print(results)
ggplot(results, aes(x = alpha, y = mse)) +
  geom_point() +
  labs(x = "Alpha", y = "MSE", title = "Relationship between Alpha and MSE")
min_mse_alpha <- results$alpha[which.min(results$mse)]
min_mse_alpha
#There is no difference in the MSE no matter the aplha value, hence we chose aplha=0.5 and
performed elastic net regression on it.
en_mod <- glmnet(x = X_train_elnet, y = Y_train_elnet, alpha = min_mse_alpha)
Y_pred_elnet <- predict(en_mod, newx = X_test_elnet)
mean((Y_pred_elnet - Y_test_elnet)^2)
elnet_coef <-coef(en_mod, s=models[[name]]$lambda.min)
Elnet_coef
#All the estimates
lm_estimates <- coef(airq_NA)
imp_estimates <- coef(airq_mice_lm)
boot_estimates <- boot.coef
elnet_coef_mat <- as.matrix(elnet_coef)
enet_estimates <- elnet_coef_mat[, "s1"]

lm_estimates
imp_estimates
boot_estimates
enet_estimates
# Create a data frame with the MSE values
df_mse <- data.frame(method = c("NA", "MICE"), mse = c(MSE_NA, MSE_mice))

# Plot the MSE values as a line plot
ggplot(df_mse, aes(x = reorder(method,-mse), y = mse, group = 1)) +
  geom_line() +
  geom_point() +
  labs(x = "", y = "MSE", title = "Comparison of MSE") +
  theme_bw()
```

RESULTS

In our analysis of the airquality dataset, techniques like multiple imputation, linear regression, bootstrap resampling, and elastic net regression were used to predict ozone levels. Employing the mice imputations, the MSE decreased significantly (Fig. 11) indicating the correctness of the data produced by the imputation technique.

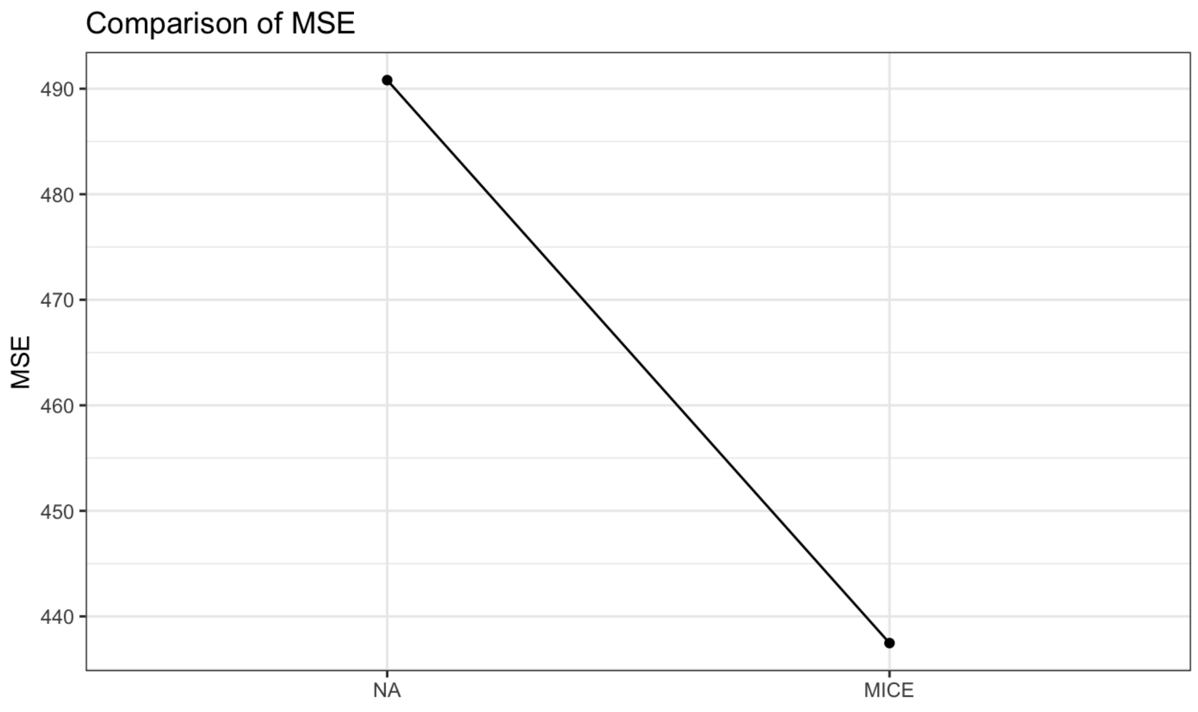


Fig. 11

The visualization of the bootstrap distributions enabled us to determine the range of values we were confident in and provided insight into the variability of the estimates.

Furthermore, elastic net regression allowed us to identify the most important predictor variables while accounting for collinearity among the predictors and also reducing the overfitting that was a concern for the linear regression model. This resulted in a more stable and parsimonious model than the linear regression model.

All in all, the techniques utilized in this analysis have enabled us to construct a model that not only exhibits high accuracy in making predictions, but also effectively handles missing data. This is an important accomplishment, as missing data is a common issue in many datasets and can compromise the reliability and generalizability of any statistical analysis.

DISCUSSION

The analysis of the airquality dataset using multiple imputation, linear regression, bootstrap resampling, and elastic net regression provided us with a comprehensive approach for exploring and predicting ozone levels. The use of multiple imputation allowed us to handle missing data, and the bootstrap resampling technique helped to reduce bias and provide more accurate estimates of the linear regression model. Elastic net regression further addressed multicollinearity issues and improved the model's predictive performance. Our results showed that the elastic net model outperformed the linear regression model, indicating the importance of variable selection in regression analysis. These techniques can be applied to other datasets, providing researchers with a powerful set of tools to overcome common challenges in regression modeling. Overall, this analysis highlights the benefits of using a combination of techniques to improve the accuracy and reliability of regression models.

LITERATURE CITED

1. https://trsekhar123.medium.com/missing-data-mcar-mar-mnar-d99e8f218f71
2. https://medium.com/@ofirdi/mice-is-nice-but-why-should-you-care-e66698f245a3
3. Prof. Jack Mardekian Lecture Notes