Codebase_Draft.rmd

2024-11-11

Air quality Data Analysis Project

This project performs exploratory data analysis (EDA), statistical analysis, and make predictions with regression models on a time-series air quality dataset.

Part 1: Loading Required packages

Loading Required R Packages:

1. tidyverse
2. zoo
3. ggfotify
4. car
5. caret
6. glmnet

```
# Install and load required libraries (install only if not already installed)
required_packages <- c("tidyverse", "zoo", "ggfortify", "car", "caret", "glmnet")
new_packages <- required_packages[!(required_packages %in% installed.packages()[, "Package"])]
if (length(new_packages)) install.packages(new_packages)
lapply(required_packages, library, character.only = TRUE)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
##
## Attaching package: 'zoo'
##
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
##
## Loading required package: carData
##
##
## Attaching package: 'car'
##
##
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
##
##
## The following object is masked from 'package:purrr':
##
##     some
##
##
## Loading required package: lattice
##
##
## Attaching package: 'caret'
##
##
## The following object is masked from 'package:purrr':
##
##     lift
##
##
## Loading required package: Matrix
##
##
## Attaching package: 'Matrix'
##
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
##
## Loaded glmnet 4.1-8
```

Part 2: Outlier and Influential Observation Detection

Steps Done:

1. Data Colleciton
2. Data Preprocessing (Cleaning, imputation, transformation)
3. Outlier and Influtnetial Observation Detection

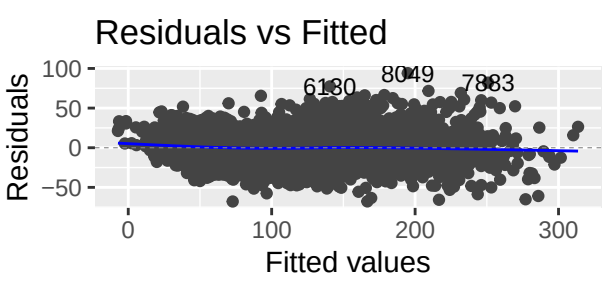
```
# Load data and check if loaded correctly
# data <- read_csv(file.choose())
data<- read_csv("/Users/alexmak/Desktop/CMPUT/School/Year_5/STAT_537/project/PersonalProject/Air-Quality
```

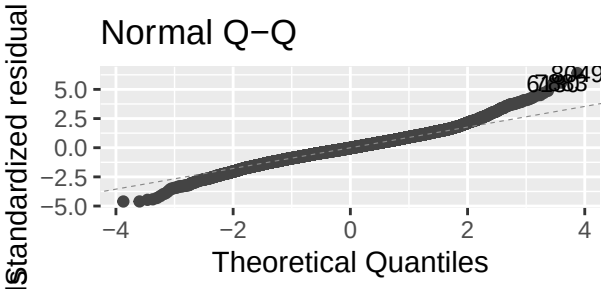
```
## New names:
## Rows: 9326 Columns: 16
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): Date dbl (14): ...1, CO(GT), PT08.S1(CO), NMHC(GT), C6H6(GT),
## PT08.S2(NMHC), NOx... time (1): Time
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
```

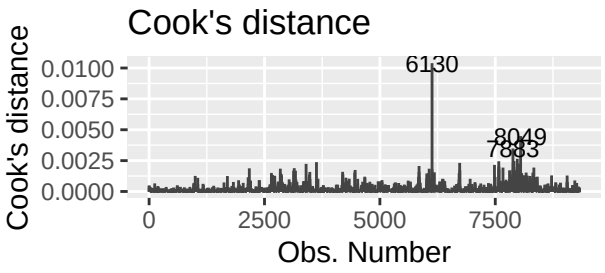
```r
# stopifnot(class(data) == "data.frame") # Ensure the data type is correct
print(summary(data)) # Check initial summary
```

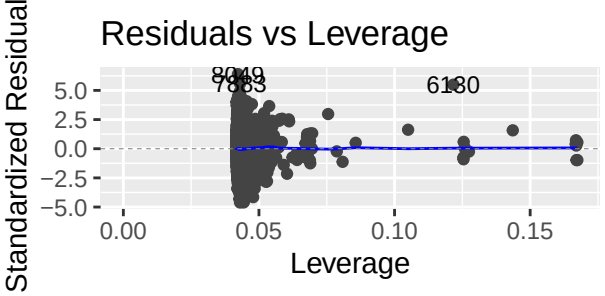
```
##       ...1          Date                Time              CO(GT)
##  Min.   :   0   Length:9326        Length:9326       Min.   : 0.100
##  1st Qu.:2345   Class :character   Class1:hms        1st Qu.: 1.091
##  Median :4694   Mode  :character   Class2:difftime   Median : 1.800
##  Mean   :4683                      Mode  :numeric    Mean   : 2.127
##  3rd Qu.:7025                                        3rd Qu.: 2.900
##  Max.   :9356                                        Max.   :11.900
##   PT08.S1(CO)      NMHC(GT)         C6H6(GT)     PT08.S2(NMHC)
##  Min.   : 647   Min.   :   7.0   Min.   : 0.10   Min.   : 383.0
##  1st Qu.: 937   1st Qu.: 275.0   1st Qu.: 4.50   1st Qu.: 736.0
##  Median :1066   Median : 275.0   Median : 8.30   Median : 910.0
##  Mean   :1103   Mean   : 269.8   Mean   :10.12   Mean   : 940.3
##  3rd Qu.:1237   3rd Qu.: 275.0   3rd Qu.:14.00   3rd Qu.:1117.0
##  Max.   :2040   Max.   :1189.0   Max.   :63.70   Max.   :2214.0
##     NOx(GT)         PT08.S3(NOx)       NO2(GT)       PT08.S4(NO2)
##  Min.   :   2.00   Min.   : 322.0   Min.   :  2.0   Min.   : 551
##  1st Qu.:  95.31   1st Qu.: 654.0   1st Qu.: 76.0   1st Qu.:1226
##  Median : 179.21   Median : 803.0   Median :104.4   Median :1459
##  Mean   : 241.75   Mean   : 832.8   Mean   :109.5   Mean   :1453
##  3rd Qu.: 325.00   3rd Qu.: 968.0   3rd Qu.:136.0   3rd Qu.:1668
##  Max.   :1479.00   Max.   :2683.0   Max.   :340.0   Max.   :2775
##   PT08.S5(O3)           T               RH              AH
##  Min.   : 221.0   Min.   :-1.90   Min.   : 9.20   Min.   :0.1847
##  1st Qu.: 733.0   1st Qu.:11.70   1st Qu.:36.00   1st Qu.:0.7325
##  Median : 968.5   Median :17.50   Median :49.70   Median :0.9906
##  Mean   :1031.1   Mean   :18.21   Mean   :49.26   Mean   :1.0202
##  3rd Qu.:1290.0   3rd Qu.:24.27   3rd Qu.:62.30   3rd Qu.:1.3078
##  Max.   :2523.0   Max.   :44.60   Max.   :88.70   Max.   :2.2310
```

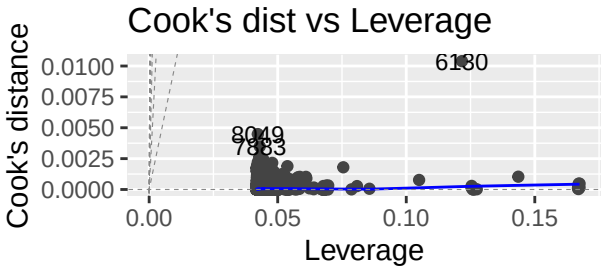
```r
# Data Preprocessing
# Drop last 2 columns and remove rows where all specific columns contain -200
updated_data <- data[1:15]
updated_data <- updated_data[rowSums(updated_data[4:15]) > -2600, ]

# Replace -200 values with NA and interpolate using last observation carry forward
updated_data[updated_data == -200.00] <- NA
updated_data <- na.locf(updated_data)

# Outlier Detection and Data Cleaning
# lm_prep <- lm(`NO2(GT)` ~ ., data = updated_data) # Simplified model formula for all columns

lm_prep=lm(`NO2(GT)` ~ Date+Time+`CO(GT)`+`PT08.S1(CO)`+`NMHC(GT)`+`C6H6(GT)`+`PT08.S2(NMHC)`+`NOx(GT)`+

# Visualize model diagnostics
autoplot(lm_prep, which = c(1, 2, 4, 5:6), label.size = 3)
```



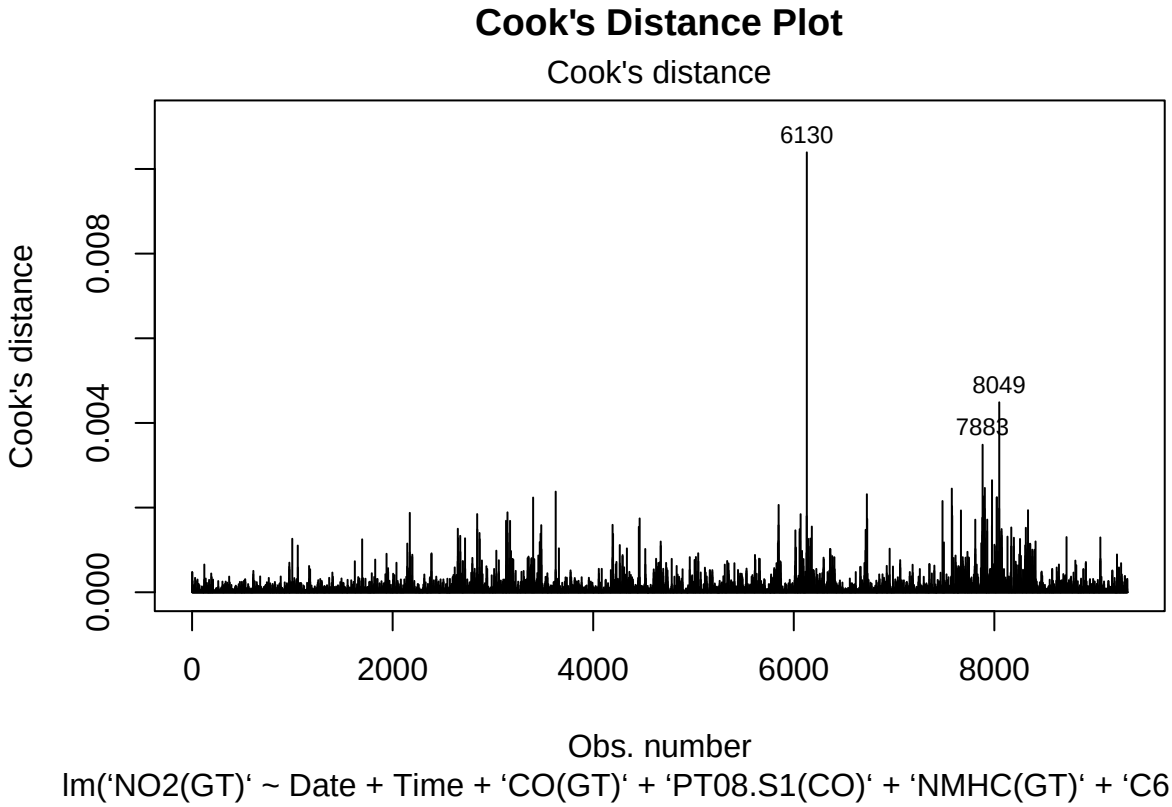









```r
# Identify influential observations and handle outliers
cutoff <- 4 / (nrow(updated_data) - length(lm_prep$coefficients))
plot(lm_prep, which = 4, cook.levels = cutoff, main = "Cook's Distance Plot")
```



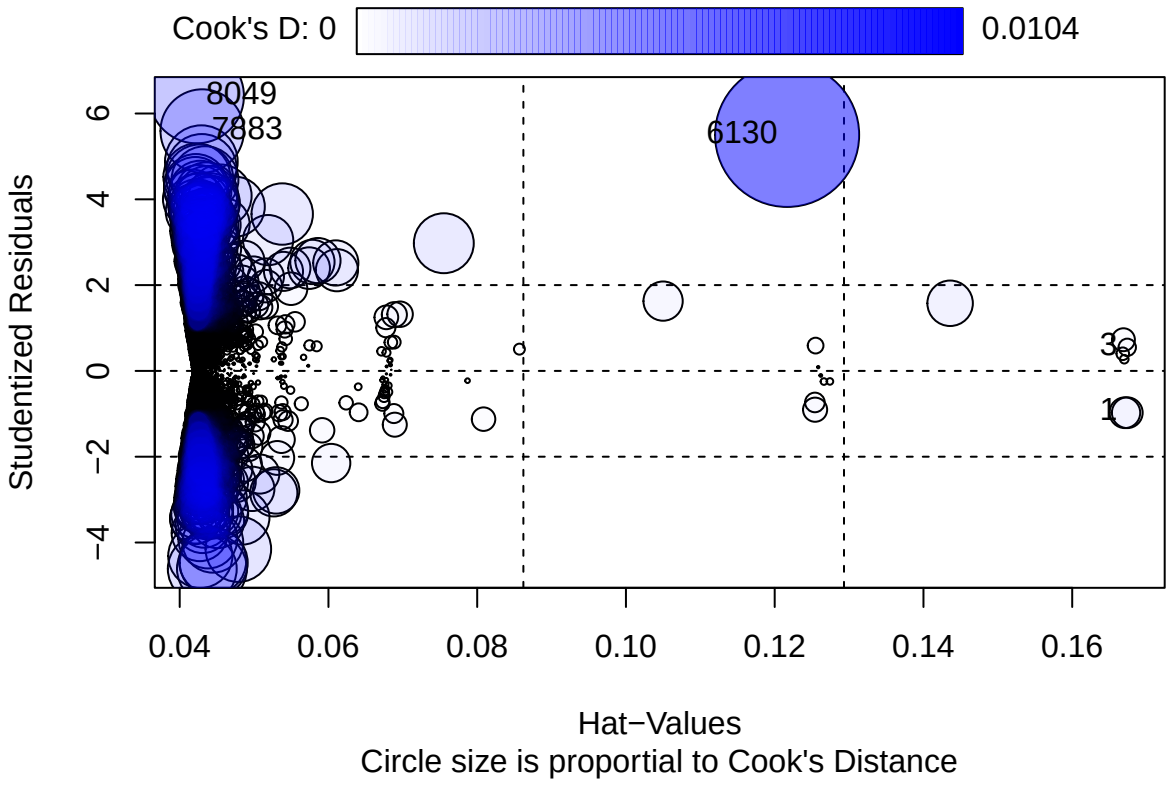

```
# Remove specific influential rows based on Cook's distance or other criteria
influential_points <- which(cooks.distance(lm_prep) > cutoff)
updated_data <- updated_data[-influential_points, ]
```

Part 3: Residual Analysis

Analyze residuals through:

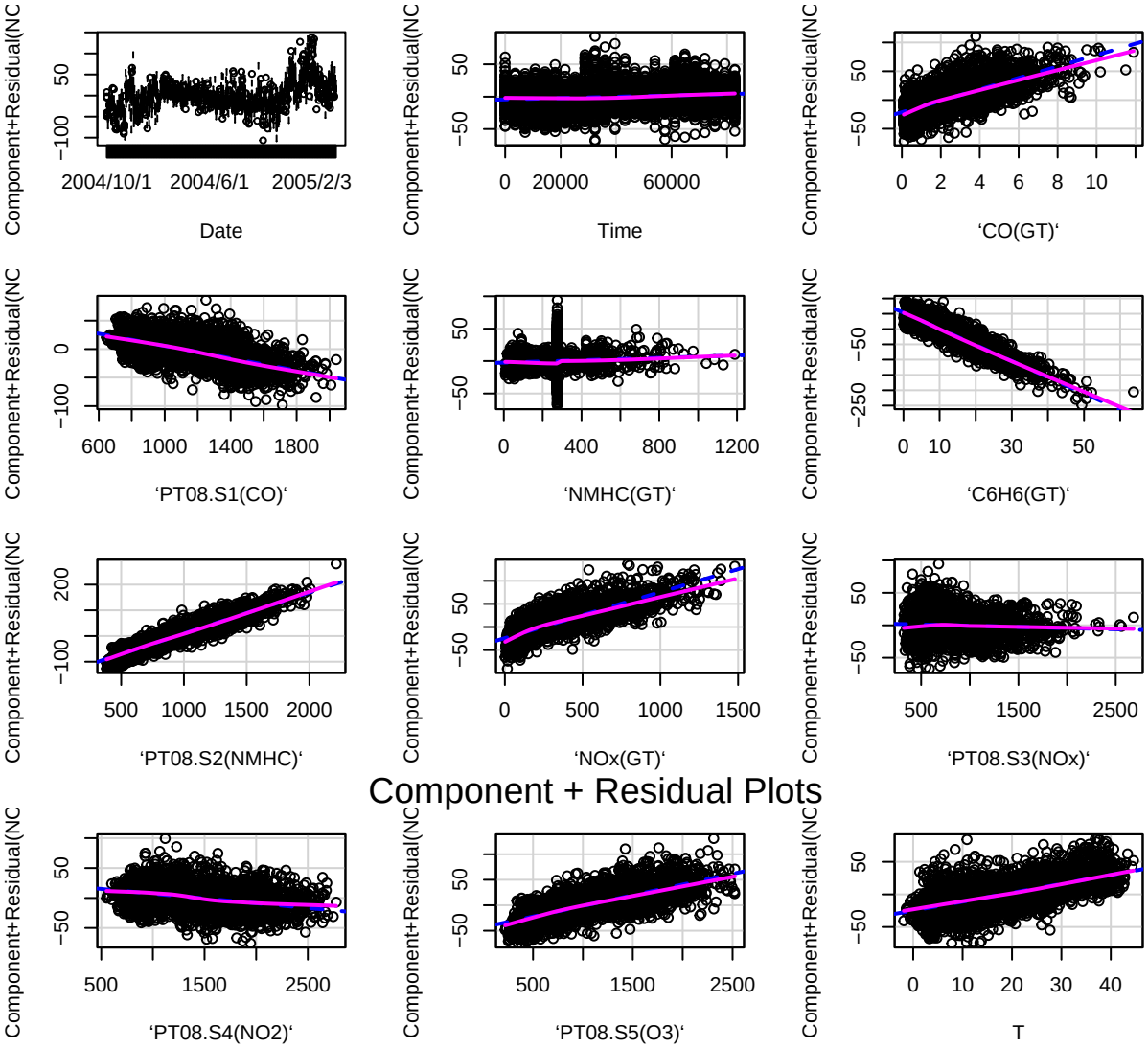
1. Influence plot
2. Partial resdual plots.
3. Durbin Watson's test to check independence
4. Normality test

```
# Residual Analysis and Collinearity Check

# Influence Plot
if(require(car)){ # Use the car library
  influencePlot(lm_prep, id.method="identify", sub="Circle size is proportial to Cook's Distance")
}
```




```
##          StudRes       Hat        CookD
## 1    -0.9742282 0.1674063 0.0004747190
## 3     0.5497699 0.1674012 0.0001511794
## 6130  5.5008252 0.1216688 0.0103927166
## 7883  5.5918783 0.0430250 0.0034852913
## 8049  6.4074693 0.0422714 0.0044875182
```

```r
if(require(car)){ # Use the car library
  crPlots(lm_prep) # Draw partial residual plots.
}
```

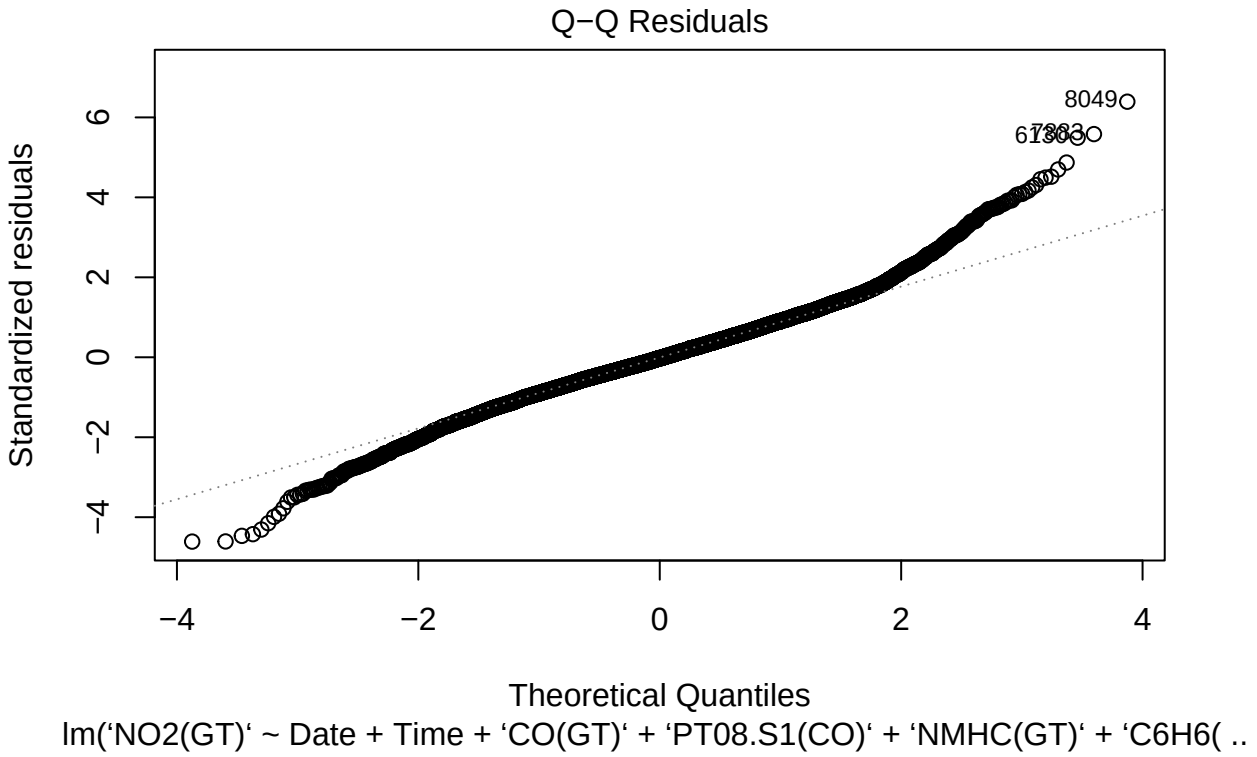


Component + Residual Plots

```
# Check independence
durbinWatsonTest(lm_prep)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.7021919     0.5953485       0
##  Alternative hypothesis: rho != 0
```

```
# Normality plots
plot(lm_prep, which=2)
```



Part 4: Multicolinearity Assessment

Access Multicolinearity through Variance Inflation Factor (VIF)

```
library(car)
# Assess colinearity using VIF:
vif(lm_prep)
```

```
##                         GVIF  Df GVIF^(1/(2*Df))
## Date            46208.928623 390        1.013866
## Time                1.578334   1        1.256318
## `CO(GT)`           11.756820   1        3.428822
## `PT08.S1(CO)`      43.127773   1        6.567174
## `NMHC(GT)`          2.204034   1        1.484599
## `C6H6(GT)`        100.280637   1       10.014022
## `PT08.S2(NMHC)`   138.999703   1       11.789814
## `NOx(GT)`          10.880503   1        3.298561
## `PT08.S3(NOx)`     18.594108   1        4.312089
## `PT08.S4(NO2)`     74.138104   1        8.610349
## `PT08.S5(O3)`      18.465769   1        4.297182
## T                  10.432769   1        3.229980
```

```
vif(lm_prep)>5
```

```
##                   GVIF    Df GVIF^(1/(2*Df))
## Date              TRUE  TRUE           FALSE
## Time             FALSE FALSE           FALSE
## `CO(GT)`          TRUE FALSE           FALSE
## `PT08.S1(CO)`     TRUE FALSE            TRUE
## `NMHC(GT)`       FALSE FALSE           FALSE
## `C6H6(GT)`        TRUE FALSE            TRUE
```

```
## `PT08.S2(NMHC)`  TRUE FALSE            TRUE
## `NOx(GT)`        TRUE FALSE           FALSE
## `PT08.S3(NOx)`   TRUE FALSE           FALSE
## `PT08.S4(NO2)`   TRUE FALSE            TRUE
## `PT08.S5(O3)`    TRUE FALSE           FALSE
## T                TRUE FALSE           FALSE
```

```
summary(updated_data)
```

```
##       ...1          Date                Time              CO(GT)
##  Min.   :   2   Length:8842        Length:8842       Min.   : 0.100
##  1st Qu.:2254   Class :character   Class1:hms        1st Qu.: 1.100
##  Median :4596   Mode  :character   Class2:difftime   Median : 1.800
##  Mean   :4616                      Mode  :numeric    Mean   : 2.092
##  3rd Qu.:6945                                        3rd Qu.: 2.800
##  Max.   :9356                                        Max.   :11.900
##   PT08.S1(CO)      NMHC(GT)         C6H6(GT)      PT08.S2(NMHC)
##  Min.   : 647   Min.   :   7.0   Min.   : 0.100   Min.   : 383.0
##  1st Qu.: 937   1st Qu.: 275.0   1st Qu.: 4.500   1st Qu.: 736.0
##  Median :1063   Median : 275.0   Median : 8.154   Median : 905.0
##  Mean   :1098   Mean   : 269.4   Mean   : 9.915   Mean   : 934.8
##  3rd Qu.:1228   3rd Qu.: 275.0   3rd Qu.:13.600   3rd Qu.:1105.0
##  Max.   :2008   Max.   :1189.0   Max.   :52.100   Max.   :2007.0
##     NOx(GT)        PT08.S3(NOx)       NO2(GT)       PT08.S4(NO2)
##  Min.   :   6.0   Min.   : 322.0   Min.   :  2.0   Min.   : 551
##  1st Qu.:  95.0   1st Qu.: 662.0   1st Qu.: 75.6   1st Qu.:1233
##  Median : 176.0   Median : 808.5   Median :104.0   Median :1460
##  Mean   : 234.6   Mean   : 835.8   Mean   :107.8   Mean   :1452
##  3rd Qu.: 315.1   3rd Qu.: 969.0   3rd Qu.:134.0   3rd Qu.:1663
##  Max.   :1479.0   Max.   :2683.0   Max.   :340.0   Max.   :2775
##   PT08.S5(O3)         T               RH
##  Min.   : 221   Min.   :-1.90   Min.   : 9.20
##  1st Qu.: 730   1st Qu.:12.00   1st Qu.:36.00
##  Median : 961   Median :17.70   Median :49.70
##  Mean   :1021   Mean   :18.33   Mean   :49.16
##  3rd Qu.:1268   3rd Qu.:24.27   3rd Qu.:62.20
##  Max.   :2523   Max.   :44.60   Max.   :88.70
```

```
# Save cleaned data
write_csv(updated_data, "ProcessedInput2.csv")
```

Part 5: Variable Selection

Perform variable selection by implementing

1. Forward Selection
2. Backward Elimination
3. Stepwise Regression (Both-direction variable selection)

Then determining the best model, which is the one with backward elimination

```
# Variable Selection
air_quality_data <- updated_data # Ensure consistency with processed data
full_model <- lm(`NO2(GT)` ~ ., data = air_quality_data)
null_model <- lm(`NO2(GT)` ~ 1, data = air_quality_data)

# Forward, backward, and stepwise selection
```

```r
forward_model <- step(null_model, direction = 'forward', scope = formula(full_model), trace = 0)
backward_model <- step(full_model, direction = 'backward', trace = 0)
both_model <- step(null_model, direction = 'both', scope = formula(full_model), trace = 0)

# Display selection summaries
print(summary(forward_model))
print(summary(backward_model))
print(summary(both_model))
```

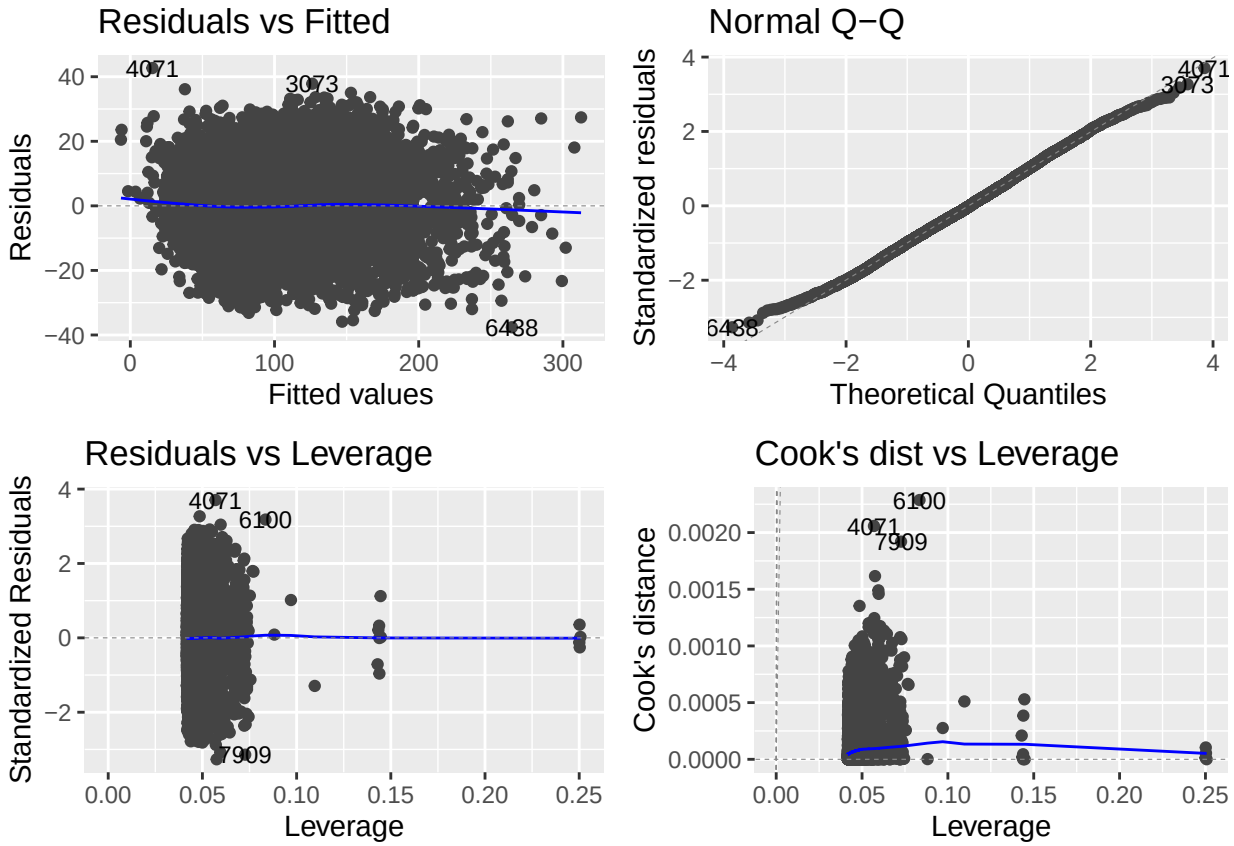
Part 6: Residual analysis for the final model

Analzyed residual of the chosen model after variable selection (backward elimination)

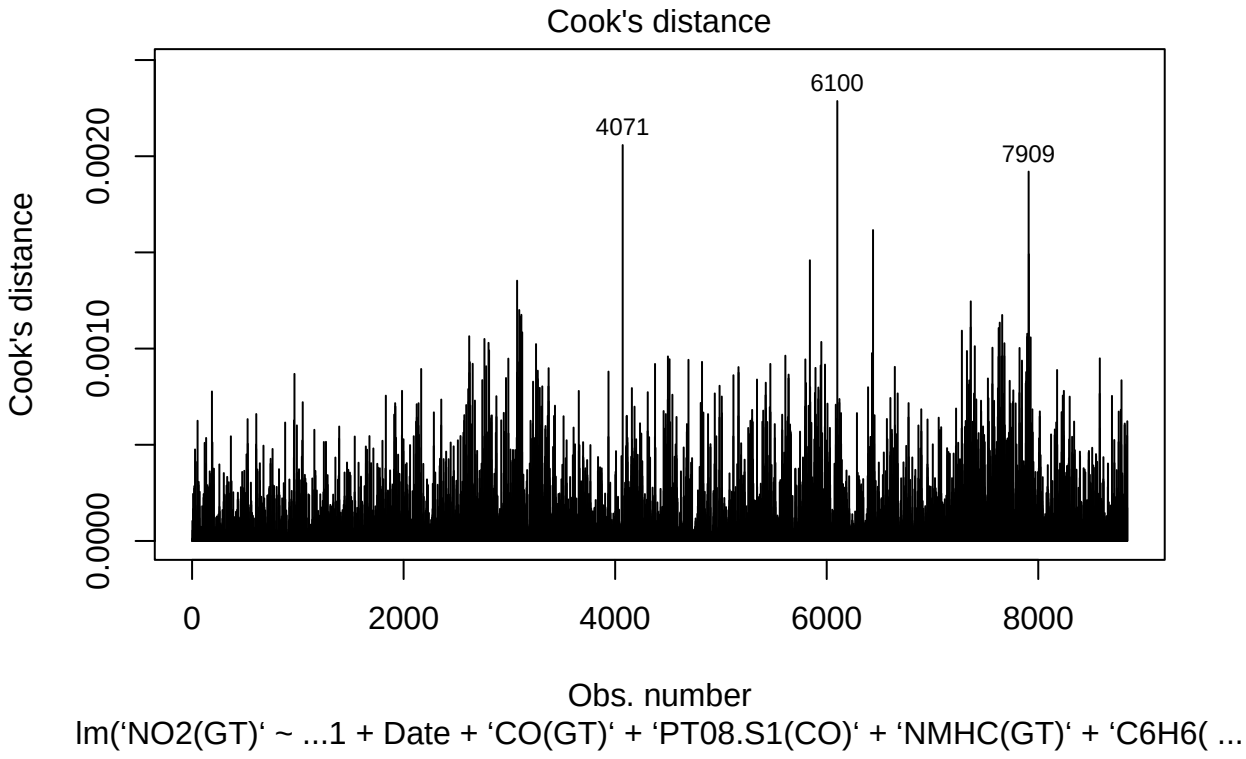
Make 4 plots at the same time:

1. Residuals vs Fitted
2. Q-Q plot
3. Residuals vs Leverage
4. Cook's dist vs Leverage

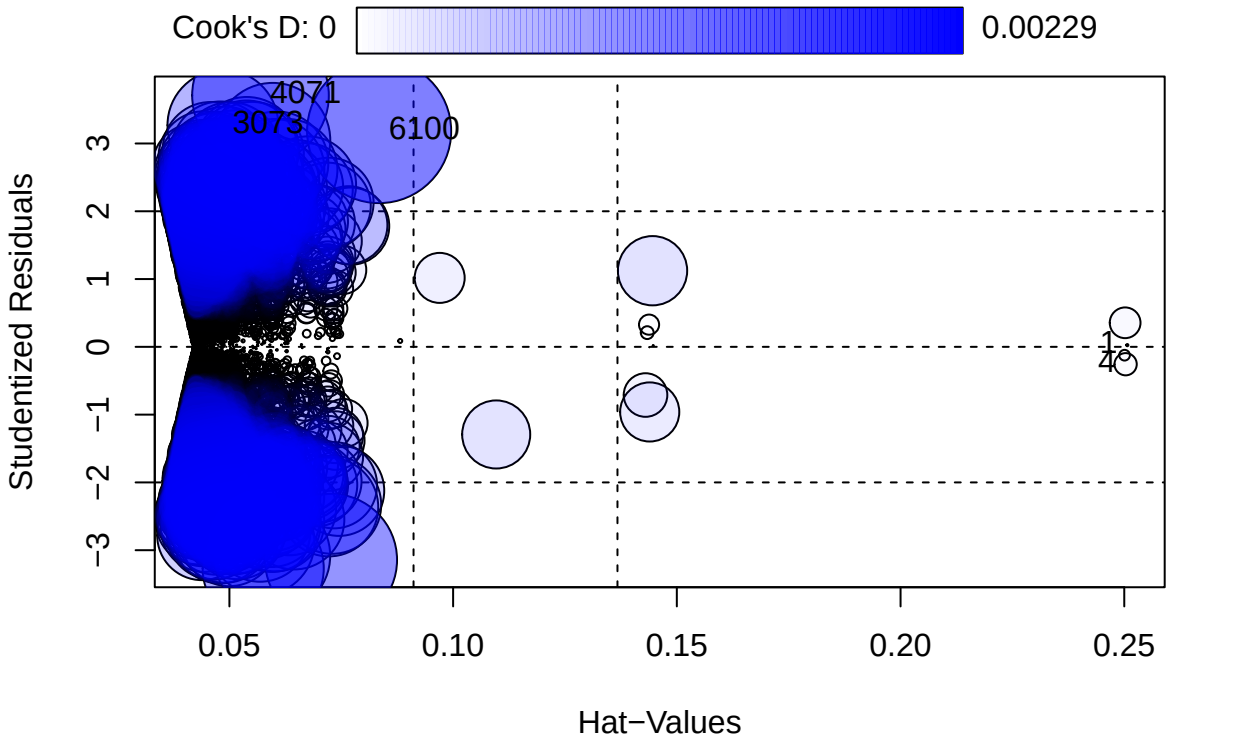
```r
# Residual analysis for the final model
autoplot(backward_model, which = c(1:2, 5:6), label.size = 3)
```



```
plot(backward_model, which = 4)
```



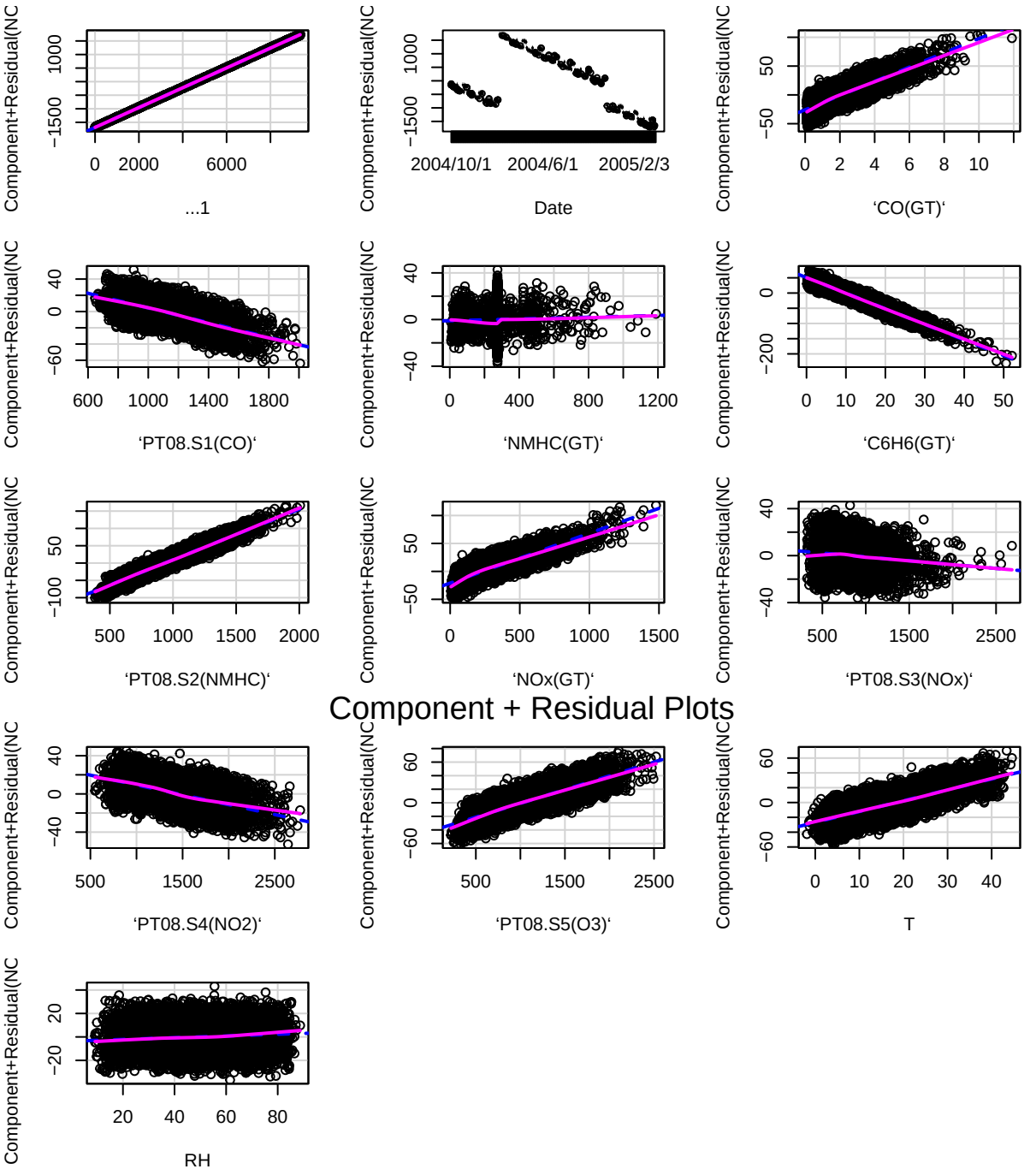

```
influencePlot(backward_model)
```




```
##          StudRes        Hat        CookD
## 1     0.02517709 0.25067563 5.262590e-07
```

```
## 4    -0.25658573 0.25030596 5.455005e-05
## 3073  3.27087829 0.04854483 1.352943e-03
## 4071  3.70875456 0.05695069 2.058068e-03
## 6100  3.18286379 0.08346885 2.286855e-03
```

```
if (require(car)) crPlots(backward_model)
```



Part 7: Train-test split (80-20)

Split Dataset to training and testing subset with 80-20 ratio

```r
# Train-test split (80-20)
splitIndex <- createDataPartition(air_quality_data$`NO2(GT)`, p = 0.8, list = FALSE)
train_data <- air_quality_data[splitIndex, ]
test_data <- air_quality_data[-splitIndex, ]
```

Part 8: Linear Regression Prediction and evaluation

Compare the R-squared & root mean sqaure error (RMSE) values between the full model and the reduced model built after variable selection

```r
cat("Regression Performance of the Full Model:\n")
```

```
## Regression Performance of the Full Model:
```

```r
lm_prep=lm(`NO2(GT)` ~ Date+Time+`CO(GT)`+`PT08.S1(CO)`+`NMHC(GT)`+`C6H6(GT)`+`PT08.S2(NMHC)`+`NOx(GT)`+

regular_predictions <- predict(lm_prep, newdata = test_data)

regular_mse <- mean((test_data$`NO2(GT)` - regular_predictions)^2)
cat("Root Mean Squared Error (RMSE):", sqrt(regular_mse), "\n")
```

```
## Root Mean Squared Error (RMSE): 12.43348
```

```r
# Compute the total sum of squares (TSS)
tss <- sum((test_data$`NO2(GT)` - mean(test_data$`NO2(GT)`))^2)

# Compute the residual sum of squares (RSS)
rss_before <- sum((test_data$`NO2(GT)` - regular_predictions)^2)

# Calculate R-squared
r_squared_before <- 1 - (rss_before / tss)

# Print the R-squared value
cat("R-Squared Value:", r_squared_before, "\n")
```

```
## R-Squared Value: 0.9197058
```

```r
# For the refined model
cat("\nRegression Performance of the Refined Model:\n")
```

```
##
## Regression Performance of the Refined Model:
```

```r
# Train model and make predictions
model_train <- lm(`NO2(GT)` ~ Date + `CO(GT)` + `NOx(GT)` + `PT08.S5(O3)` + RH, data = train_data)
refined_predictions <- predict(model_train, newdata = test_data)

# Evaluate model performance
mse <- mean((test_data$`NO2(GT)` - refined_predictions)^2)
rmse <- sqrt(mse)
cat("Root Mean Squared Error (RMSE):", rmse, "\n")
```

```
## Root Mean Squared Error (RMSE): 14.96114
```

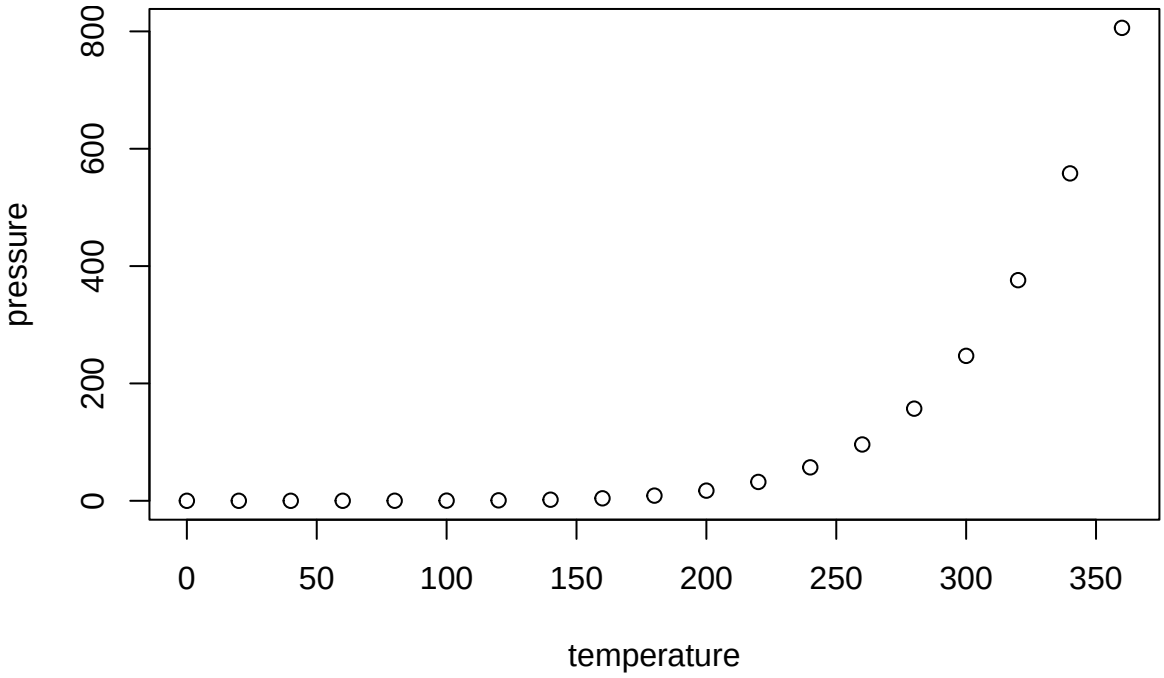
```r
# # Compute the total sum of squares (TSS)
# tss <- sum((testData$`NO2(GT)` - mean(testData$`NO2(GT)`))^2)

# Compute the residual sum of squares (RSS)
```

```r
rss_after <- sum((test_data$`NO2(GT)` - refined_predictions)^2)

# Calculate R-squared
r_squared_after <- 1 - (rss_after / tss)

# Print the R-squared value
cat("R-Squared Value:", r_squared_after, "\n")
```

```
## R-Squared Value: 0.8837404
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.