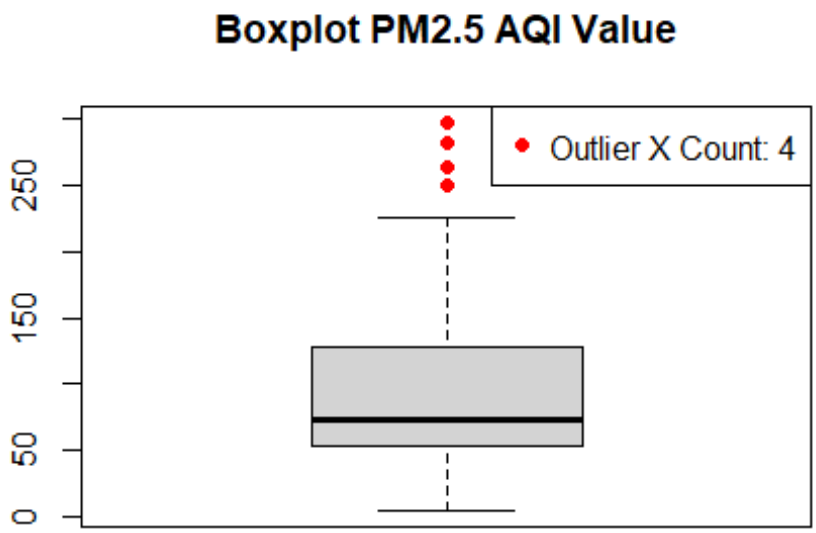
Project-UAS-Nonpar-Data-Monotonik.R

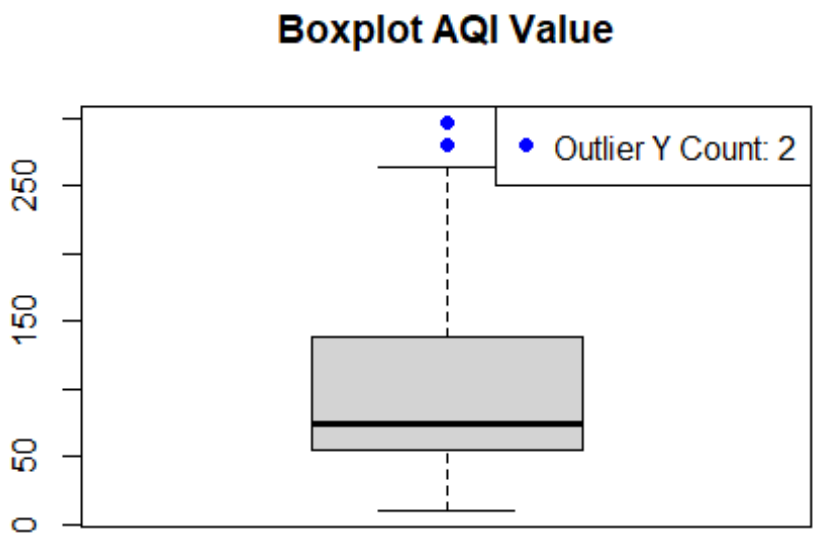
Kelompok H

Kelas Nonparametrik A

```
# Import Library
library(readxl)
library(ggplot2)
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

# Load Data
lokasi_file <- "D:/data_Indonesia.xlsx"
df1 <- read_excel(lokasi_file)
df2 <- df1 %>%
  select('PM2.5 AQI Value','AQI Value')
head(df2)

## # A tibble: 6 × 2
##   `PM2.5 AQI Value` `AQI Value`
##               <dbl>       <dbl>
## 1                44          44
## 2                21          21
## 3                17          17
## 4                88          88
## 5                92          92
## 6                54          54

# Periksa Outlier
# Menghitung outlier dan membuat boxplot untuk PM2.5 AQI Value
outliers_x <- boxplot(df2$`PM2.5 AQI Value`, plot=FALSE)$out
boxplot(df2$`PM2.5 AQI Value`, main="Boxplot PM2.5 AQI Value")
if(length(outliers_x) > 0) {
  points(rep(1, length(outliers_x)), outliers_x, col="red", pch=19)
  legend("topright", legend=paste("Outlier X Count:", length(outliers_x)), co
l="red", pch=19)
}
```




```r
# Menghitung outlier dan membuat boxplot untuk AQI Value
outliers_y <- boxplot(df2$`AQI Value`, plot=FALSE)$out
boxplot(df2$`AQI Value`, main="Boxplot AQI Value")
if(length(outliers_y) > 0) {
  points(rep(1, length(outliers_y)), outliers_y, col="blue", pch=19)
  legend("topright", legend=paste("Outlier Y Count:", length(outliers_y)), co
l="blue", pch=19)
}
```



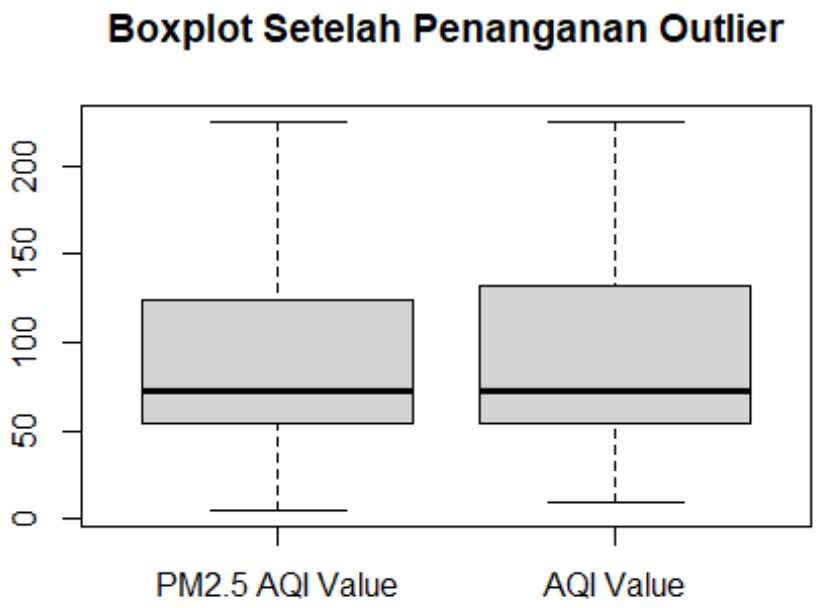

```
# Karena outlier hanya sedikit, maka outlier akan dibuang berdasarkan outlier
dari variabel X dan Y
# Menghapus baris yang mengandung outlier pada kedua variabel
data <- df2[!df2$`PM2.5 AQI Value` %in% outliers_x & !df2$`AQI Value` %in% ou
tliers_y, ]
# Menampilkan data yang sudah dibersihkan
print("Data setelah menghapus outlier pada kedua variabel:")
```

```
## [1] "Data setelah menghapus outlier pada kedua variabel:"
```

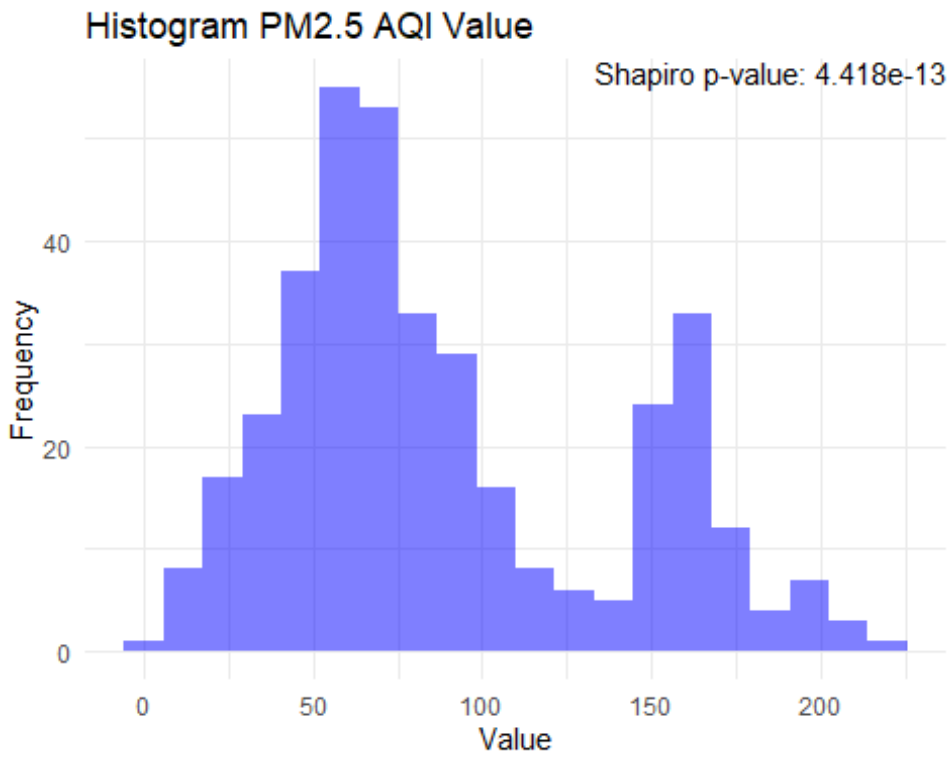
```
print(data)
```

```
## # A tibble: 375 × 2
##    `PM2.5 AQI Value` `AQI Value`
##                <dbl>       <dbl>
##  1                44          44
##  2                21          21
##  3                17          17
##  4                88          88
##  5                92          92
##  6                54          54
##  7                49          49
##  8                63          63
##  9                98          98
## 10               155         155
## # ℹ 365 more rows
```

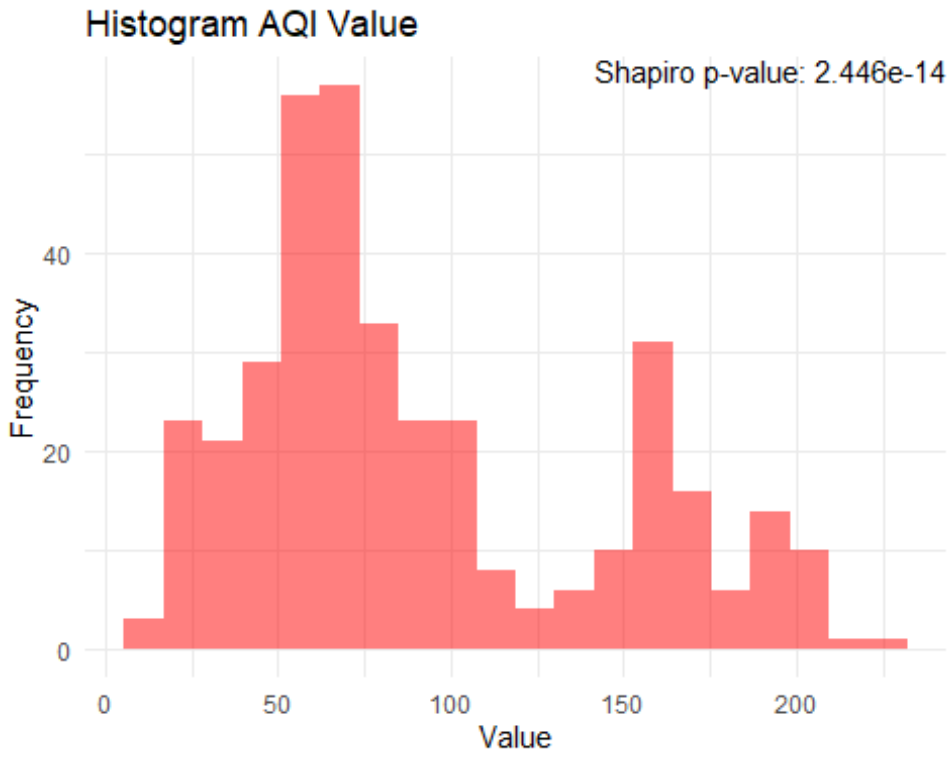
```r
boxplot(data, main='Boxplot Setelah Penanganan Outlier') # sudah tidak ada ou
tlier
```




```r
# Menyederhanakan variabel
data_y <- data$`AQI Value`
data_x <- data$`PM2.5 AQI Value`

# Uji asumsi data tidak berdistribusi normal
# Uji Shapiro-Wilk untuk data_x
shapiro_x <- shapiro.test(data_x)
# Plot histogram untuk data_x dengan anotasi Shapiro p-value
histogram_x <- ggplot(data.frame(value = data_x), aes(x = value)) +
  geom_histogram(bins = 20, fill = "blue", alpha = 0.5) +
  labs(title = "Histogram PM2.5 AQI Value", x = "Value", y = "Frequency") +
  annotate("text", x = Inf, y = Inf, label = paste("Shapiro p-value:", format
(shapiro_x$p.value, digits = 4)), hjust = 1, vjust = 1) +
  theme_minimal()
print(histogram_x) # Tampilkan histogram untuk data_x
```




```r
# Uji Shapiro-Wilk untuk data_y
shapiro_y <- shapiro.test(data_y)
# Plot histogram untuk data_y dengan anotasi Shapiro p-value
histogram_y <- ggplot(data.frame(value = data_y), aes(x = value)) +
  geom_histogram(bins = 20, fill = "red", alpha = 0.5) +
  labs(title = "Histogram AQI Value", x = "Value", y = "Frequency") +
  annotate("text", x = Inf, y = Inf, label = paste("Shapiro p-value:", format
(shapiro_y$p.value, digits = 4)), hjust = 1, vjust = 1) +
  theme_minimal()
print(histogram_y) # Tampilkan histogram untuk data_y
```




```r
# Terbukti bahwa data x dan y, keduanya tidak berdistribusi normal sehingga d
apat digunakan metode nonparametrik

# Menguji apakah data terdapat tren dengan Monotonicity Test
# Membuat fungsi untuk uji Monotonik
monotonicity_test <- function(x, y) {
  n <- length(x)
  m <- length(y)

  if (n != m) {
    stop("Jumlah data pada kedua rangkaian tidak sama")
  }

  concordant <- 0
  discordant <- 0

  for (i in 1:(n - 1)) {
    for (j in (i + 1):n) {
      if ((x[i] - x[j]) * (y[i] - y[j]) > 0) {
        concordant <- concordant + 1
      } else {
        discordant <- discordant + 1
      }
    }
  }
```

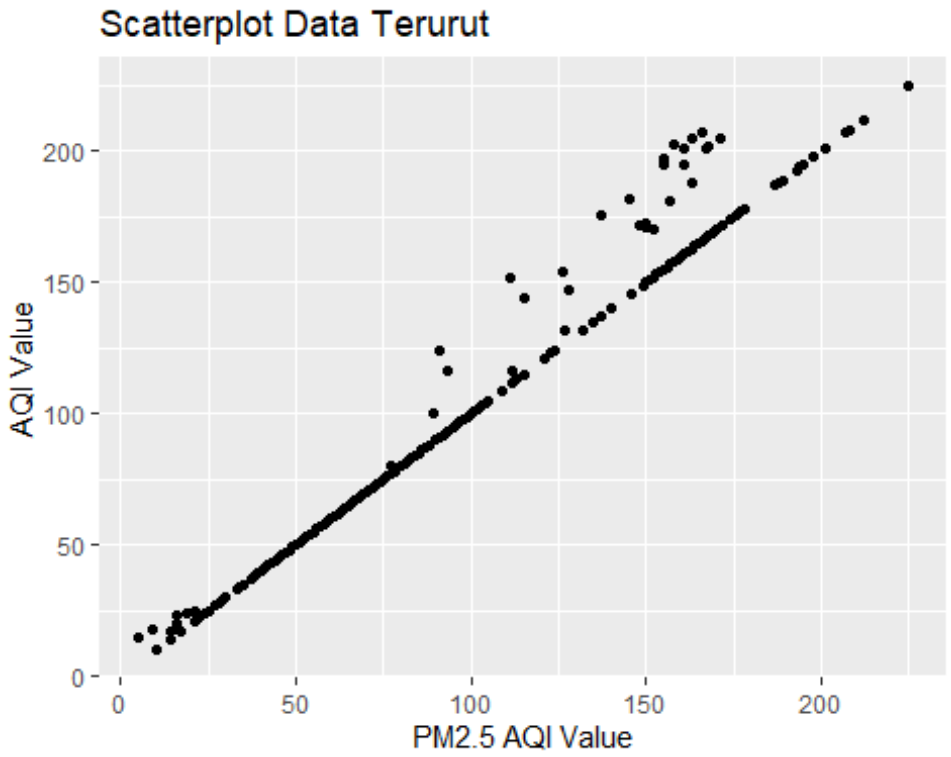
```r
  statistic <- (concordant - discordant) / (concordant + discordant)

  return(statistic)
}
# Melakukan Monotonicity Test
result <- monotonicity_test(data_x, data_y)
print(result)
```

```
## [1] 0.9592157
```

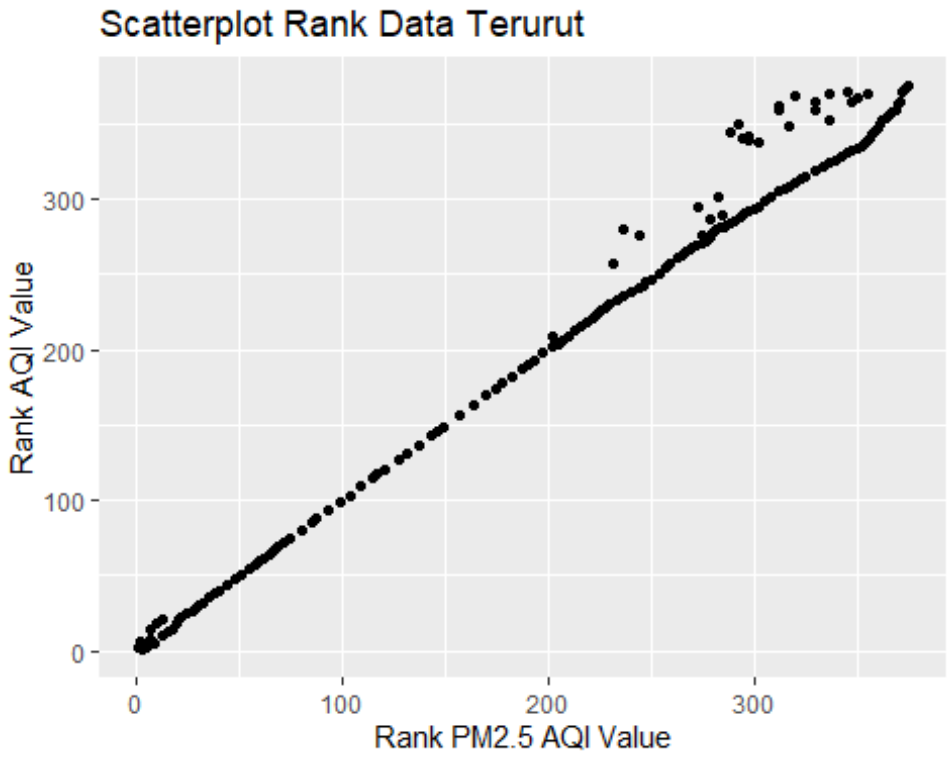
```r
# Karena nilai korelasi dengan metode nonparametrik mendekati 1, maka dapat d
iketahui bahwa terdapat hubungan keteraturan atau tren naik (karena nilainya
positif) yang sangat kuat

# Mengurutkan data berdasarkan nilai variabel X
urutan <- order(data_x)
data_urut <- data[order(data_x), ]
head(data_urut)
```

```
## # A tibble: 6 × 2
##   `PM2.5 AQI Value` `AQI Value`
##               <dbl>       <dbl>
## 1                 5          15
## 2                 9          18
## 3                10          10
## 4                14          17
## 5                14          14
## 6                16          23
```

```r
data_urut_y <- data_urut$'AQI Value'
data_urut_x <- data_urut$'PM2.5 AQI Value'

# Plot Data Terurut
ggplot(data = data_urut, aes(x = data_urut_x, y = data_urut_y)) +
  geom_point() +
  labs(x = "PM2.5 AQI Value", y = "AQI Value", title = "Scatterplot Data Teru
rut")
```




```r
# Aplikasi Regresi Monotonik
# Menghitung ranking untuk setiap kolom di data_urut
data_urut$rank_y <- rank(data_urut_y)
data_urut$rank_x <- rank(data_urut_x)

# Menyimpan peringkat ke dalam variabel terpisah
rank_y <- as.numeric(data_urut$rank_y)
rank_x <- as.numeric(data_urut$rank_x)

# Plot Rank Data Terurut
ggplot(data = data_urut, aes(x = rank_x, y = rank_y)) +
  geom_point() +
  labs(x = "Rank PM2.5 AQI Value", y = "Rank AQI Value", title = "Scatterplot
Rank Data Terurut")
```




```r
# Membentuk persamaan regresi monotonik
# Menghitung jumlah perkalian dari setiap pasangan nilai ranking X dan Y
jumlah_perkalian <- sum(rank_x * rank_y)
print(jumlah_perkalian)
```

```
## [1] 17628206
```

```r
# Menghitung jumlah kuadrat nilai ranking X
rank_x_squared <- sum(rank_x^2)
print(rank_x_squared)
```

```
## [1] 17647852
```

```r
n <- length(rank_x)
# Menghitung pembilang dan penyebut untuk b2
pembilang_b2 <- sum(jumlah_perkalian - (n * (n + 1)^2 / 4))
penyebut_b2 <- sum(rank_x_squared - (n * (n + 1)^2 / 4))
b2 <- sum(pembilang_b2 / penyebut_b2)
# Menghitung a2
a2 <- sum((1 - b2) * (n + 1) * (1 / 2))
# Menampilkan hasil
b2
```

```
## [1] 0.9955289
```

```r
a2
```

```
## [1] 0.8405625
```

```r
# Diperoleh persamaan regresi monotonik y=0.8405625+0.9955289x

# Menghitung rank_y_cap dan rank_x_cap
rank_y_cap <- a2 + b2 * rank_x
rank_x_cap <- (rank_y - b2) / a2
# Menyimpan hasil ke dalam dataframe
data_urut$rank_y_cap <- rank_y_cap
data_urut$rank_x_cap <- rank_x_cap

# Fungsi untuk mencari x_cap
cari_x_cap <- function(rank_x_cap, rank_x, x) {
  n <- length(rank_x)
  x_cap <- numeric(length(rank_x_cap))

  for (i in 1:length(rank_x_cap)) {
    if (rank_x_cap[i] %in% rank_x) {
      # Jika rank_x_cap[i] sama dengan salah satu rank_x[j]
      x_cap[i] <- x[which(rank_x == rank_x_cap[i])]
    } else if (rank_x_cap[i] < min(rank_x) || rank_x_cap[i] > max(rank_x)) {
      # Jika rank_x_cap[i] kurang dari rank_x terkecil atau lebih dari rank_x
terbesar
      x_cap[i] <- NA
    } else {
      # Jika rank_x_cap[i] berada di antara dua nilai rank_x[j] dan rank_x[k]
      j <- max(rank_x[rank_x < rank_x_cap[i]])
      k <- min(rank_x[rank_x > rank_x_cap[i]])
      index_j <- which(rank_x == j)
      index_k <- which(rank_x == k)
      x_cap[i] <- x[index_j] + (rank_x_cap[i] - j) / (k - j) * (x[index_k] -
x[index_j])
    }
  }

  return(x_cap)
}

# Menggunakan fungsi untuk mencari x_cap
hasil_x_cap <- cari_x_cap(rank_x_cap, rank_x, data_urut_x)

head(hasil_x_cap)

## [1]  9.384678 16.321698        NA 14.264037  5.779994 22.580416

# Fungsi untuk mencari y_cap
cari_y_cap <- function(rank_y_cap, rank_y, y) {
  n <- length(rank_y)
  y_cap <- numeric(length(rank_y_cap))

  for (i in 1:length(rank_y_cap)) {
    if (rank_y_cap[i] %in% rank_y) {
```

```
      # Jika rank_y_cap[i] sama dengan salah satu rank_y[j]
      y_cap[i] <- y[which(rank_y == rank_y_cap[i])]
    } else if (rank_y_cap[i] < min(rank_y) || rank_y_cap[i] > max(rank_y)) {
      # Jika rank_y_cap[i] kurang dari rank_y terkecil atau lebih dari rank_y
terbesar
      y_cap[i] <- NA
    } else {
      # Jika rank_y_cap[i] berada di antara dua nilai rank_y[j] dan rank_y[k]
      j <- max(rank_y[rank_y < rank_y_cap[i]])
      k <- min(rank_y[rank_y > rank_y_cap[i]])
      index_j <- which(rank_y == j)
      index_k <- which(rank_y == k)
      y_cap[i] <- y[index_j] + (rank_y_cap[i] - j) / (k - j) * (y[index_k] -
y[index_j])
    }
  }

  return(y_cap)
}

# Menggunakan fungsi untuk mencari y_cap
hasil_y_cap <- cari_y_cap(rank_y_cap, rank_y, data_urut_y)

head(hasil_y_cap)

## [1] 13.34437 14.83162 15.82715 17.16022 17.16022 18.62300

# Menyimpan hasil ke dalam dataframe
data_urut$y_cap <- hasil_y_cap
data_urut$x_cap <- hasil_x_cap

head(data_urut)

## # A tibble: 6 × 8
##   `PM2.5 AQI Value` `AQI Value` rank_y rank_x rank_y_cap rank_x_cap y_cap
x_cap
##               <dbl>       <dbl>  <dbl>  <dbl>      <dbl>      <dbl> <dbl>
<dbl>
## 1                 5          15      3    1         1.84    2.38     13.3
9.38
## 2                 9          18      7    2         2.83    7.14     14.8
16.3
## 3                10          10      1    3         3.83    0.00532  15.8
NA
## 4                14          17      5    4.5       5.32    4.76     17.2
14.3
## 5                14          14      2    4.5       5.32    1.19     17.2
5.78
## 6                16          23     15    6.5       7.31   16.7      18.6
22.6
```

```r
# Dalam proses regresi nonparametrik monotonik besar kemungkinan hasil x_cap
dan y_cap bernilai Na
# Menghapus baris dengan nilai NA dari dataframe
data_cleaned <- na.omit(data_urut)
data_cleaned
```

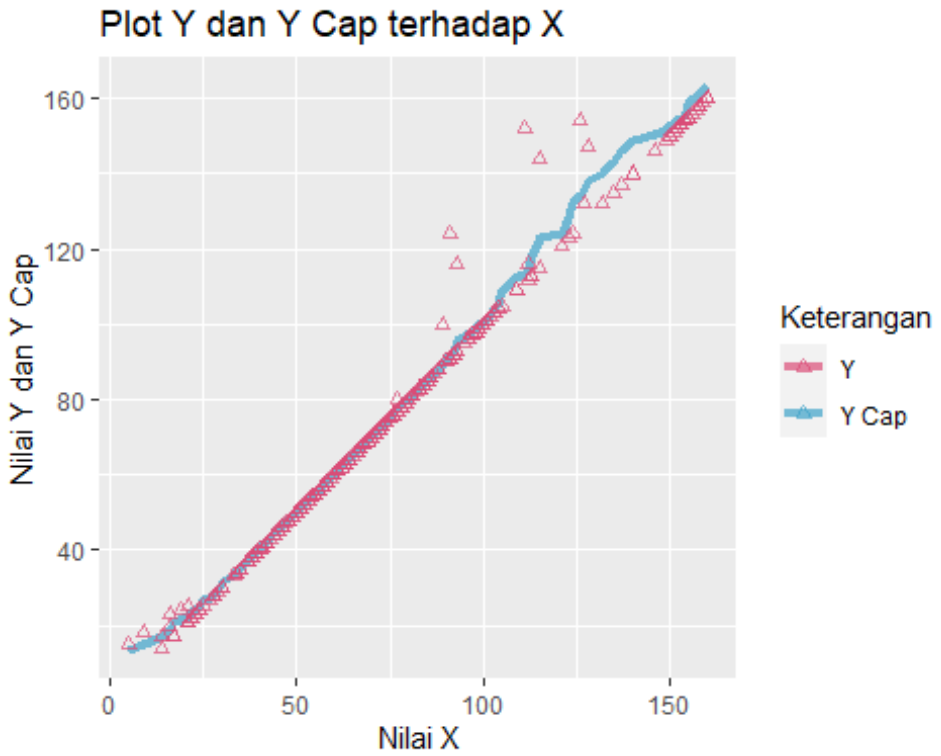
```
## # A tibble: 315 × 8
##    `PM2.5 AQI Value` `AQI Value` rank_y rank_x rank_y_cap rank_x_cap y_cap
x_cap
##                <dbl>       <dbl>  <dbl>  <dbl>      <dbl>      <dbl> <dbl>
<dbl>
##  1                 5          15      3    1         1.84       2.38  13.3
9.38
##  2                 9          18      7    2         2.83       7.14  14.8
16.3
##  3                14          17      5    4.5       5.32       4.76  17.2
14.3
##  4                14          14      2    4.5       5.32       1.19  17.2
5.78
##  5                16          23     15    6.5       7.31      16.7   18.6
22.6
##  6                16          20      8    6.5       7.31       8.33  18.6
16.9
##  7                17          17      5    8.5       9.30       4.76  20.7
14.3
##  8                17          17      5    8.5       9.30       4.76  20.7
14.3
##  9                19          24     18   10        10.8       20.2   21.3
24.5
## 10                21          21     10   12.5      13.3       10.7   22.3
19.6
## # ℹ 305 more rows
```

```r
# Menghitung Mean Squared Error (MSE) antara hasil_y_cap dan y
mse_y <- mean((data_cleaned$y_cap - data_cleaned$`AQI Value`)^2, na.rm = TRUE
)
print(paste("MSE antara hasil_y_cap dan y:", mse_y))
```

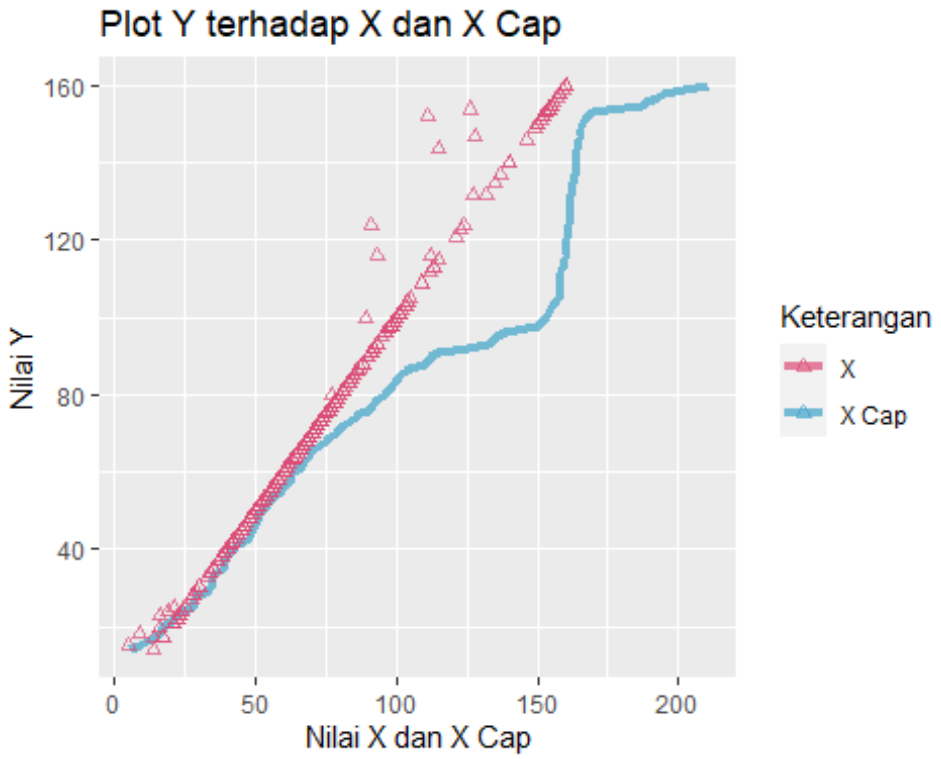
```
## [1] "MSE antara hasil_y_cap dan y: 16.0499654513127"
```

```r
# Membuat plot perbandingan y_cap dan y
ggplot(data_cleaned, aes(x = `PM2.5 AQI Value`)) +
  geom_line(aes(y = y_cap, color = "Y Cap"), alpha = 0.7, size = 1.5) +
  geom_point(aes(y = `AQI Value`, color = "Y"), shape = 2, alpha = 0.7) +
  xlab("Nilai X") +
  ylab("Nilai Y dan Y Cap") +
  ggtitle("Plot Y dan Y Cap terhadap X") +
  labs(color = "Keterangan") +
  scale_color_manual(values = c("Y Cap" = "#3CA2C8", "Y" = "#DB4C77"))
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




```r
# Membuat plot perbandingan x_cap dan x
ggplot(data_cleaned, aes(y = `AQI Value`)) +
  geom_line(aes(x = x_cap, color = "X Cap"), alpha = 0.7, size = 1.5) +
  geom_point(aes(x = `PM2.5 AQI Value`, color = "X"), shape = 2, alpha = 0.7)
+
  xlab("Nilai X dan X Cap") +
  ylab("Nilai Y") +
  ggtitle("Plot Y terhadap X dan X Cap") +
  labs(color = "Keterangan") +
  scale_color_manual(values = c("X Cap" = "#3CA2C8", "X" = "#DB4C77"))
```

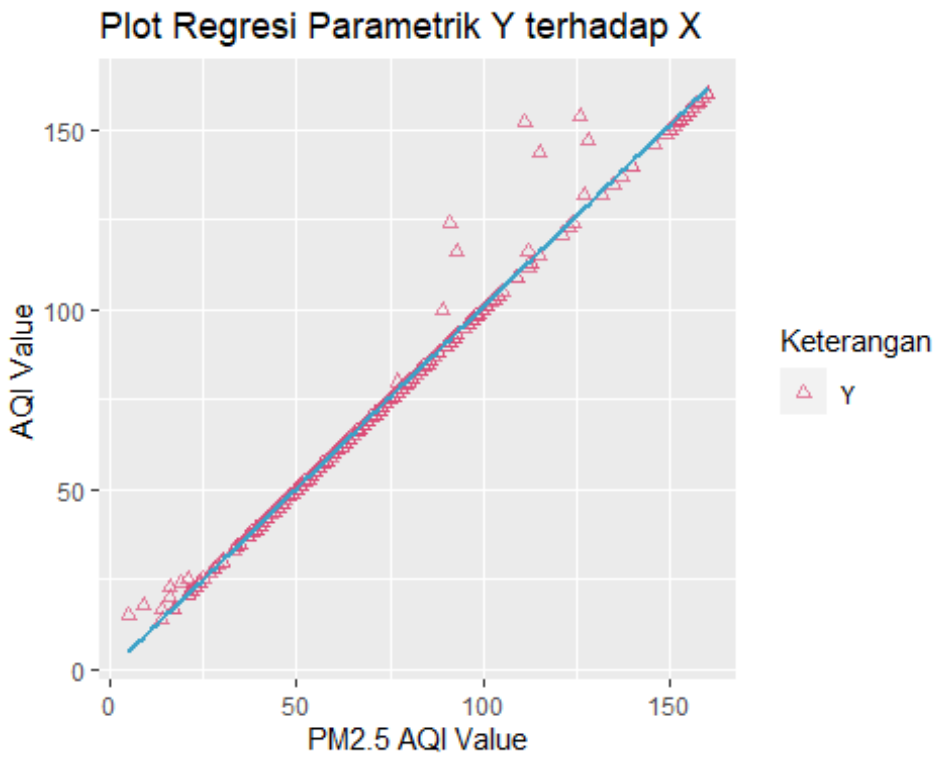



```
# Metode Parametrik: Least Square Estimator
# Membuat model regresi dengan metode parametrik least square
lm_model <- lm(`AQI Value` ~ `PM2.5 AQI Value`, data = data_cleaned)
# Menghitung MSE dari model lm
predicted_values <- predict(lm_model, data_cleaned)
mse_lm <- mean((predicted_values - data_cleaned$`AQI Value`)^2)
print(paste("MSE dari model lm:", mse_lm))
```

```
## [1] "MSE dari model lm: 17.5398834696552"
```

```
# Membuat plot scatter plot antara Y dan X dengan garis regresi
ggplot(data_cleaned, aes(x = `PM2.5 AQI Value`, y = `AQI Value`)) +
  geom_point(aes(color = "Y"), shape = 2, alpha = 0.7) +
  geom_smooth(method = "lm", se = FALSE, color = "#3CA2C8", size = 0.8) + # M
engubah size menjadi 0.8
  xlab("PM2.5 AQI Value") +
  ylab("AQI Value") +
  ggtitle("Plot Regresi Parametrik Y terhadap X") +
  labs(color = "Keterangan") +
  scale_color_manual(values = c("Y" = "#DB4C77"))
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
# Kesimpulan
if (mse_y <= mse_lm) {
  print("Model Regresi Nonparametrik Monotonik Lebih Baik dari Model Regresi
Parametrik")
} else {
  print("Model Regresi Nonparametrik Monotonik Tidak Lebih Baik dari Model Re
gresi Parametrik.")
}
```

```
## [1] "Model Regresi Nonparametrik Monotonik Lebih Baik dari Model Regresi P
arametrik"
```