Data Mining Homework 3

Group 7

11/8/2020

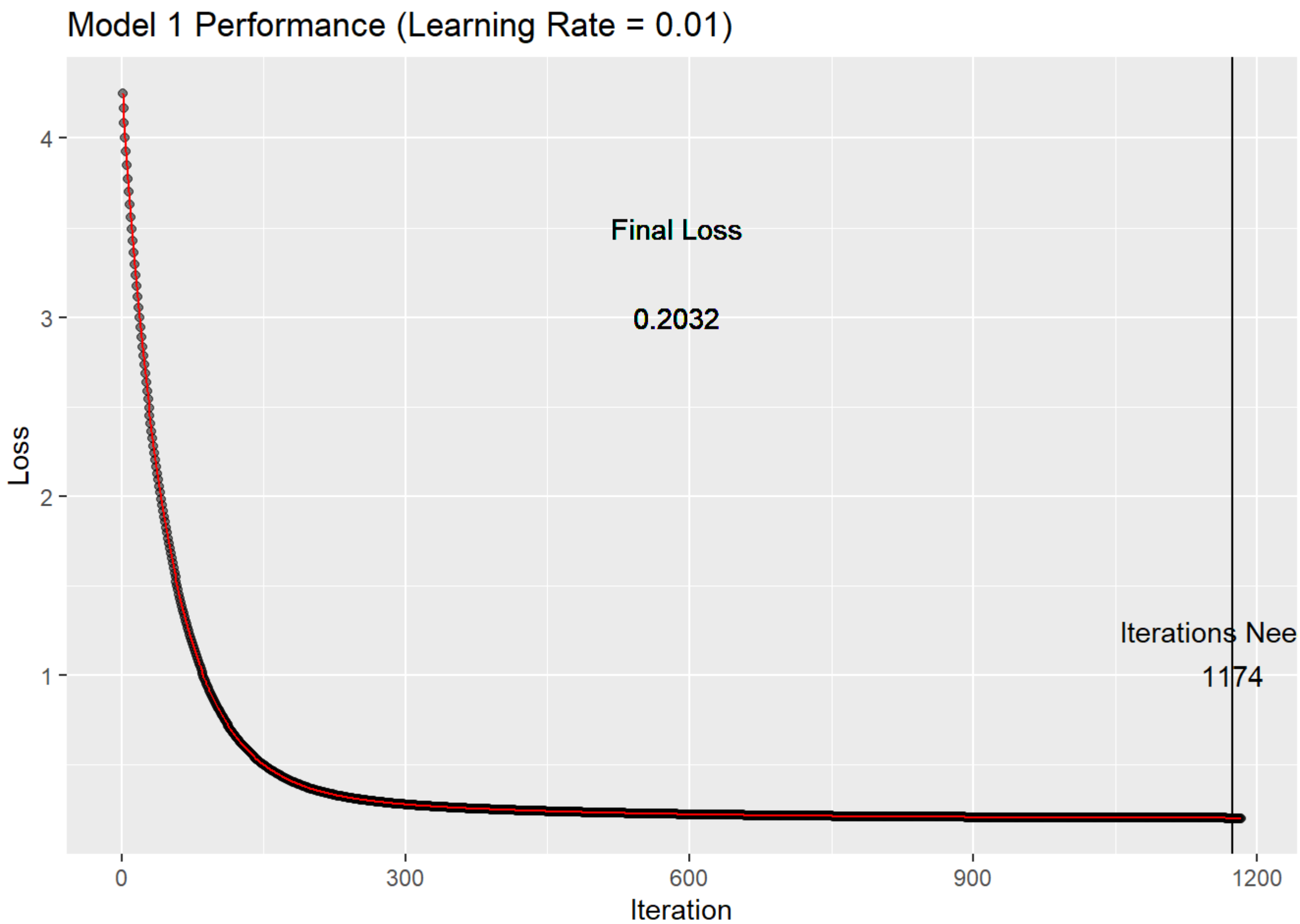
```
# Read the csv file
concrete <- data.table(read.csv("concrete.csv"))

# Scale the dataframe
concrete <- as.data.frame(scale(concrete))

# Split into train and validation datasets
training_rows <- sample(seq_len(nrow(concrete)), size = floor(0.6 * nrow(concrete)))

train_data <- concrete[training_rows, ]
validation_data <- concrete[-training_rows, ]
```

```r
# Define the gradient descent function
gradient_desc <- function(x, y, lr, iters) {
  # First we create a list to keep the track
  # of the cost function for each iteration
  losses <- list()

  # Convert y to a matrix
  y <- as.matrix(y)

  # create a column of 1
  ones <- rep(1, dim(x)[[1]])
  # append it to the input (this is our X0)
  X <- as.matrix(cbind(ones, x))
  # Calculate number of samples
  n <- length(y)

  # Initialize model parameters/coefficients
  theta <- as.matrix(rnorm(n = dim(X)[2], 0, 1))

  # Calculate model predictions
  y_hat <- X %*% theta

  # calculate the loss using OLS cost function
  loss <- sum((y_hat - y)^2) / (2 * n)

  # Calculate the gradients of the cost function
  grads <- t(X) %*% (y_hat - y)

  # Update theta
  theta <- theta - lr * (1 / n) * grads


  # That was the first iteration of the gradient descent algorithm
  # Let's add the cost function to the list
  losses[[1]] <- loss


  counter <- 0
  # Number of iterations required to get the lowest loss
  sufficient_iterations <- 0
  for (i in 1:iters) {
    # Calculate model predictions
    y_hat <- X %*% theta

    # Calculate the loss using OLS cost function
    loss <- sum((y_hat - y)^2) / (2 * n)

    # Calculate the gradients
    grads <- t(X) %*% (y_hat - y)

    # Update theta
    theta <- theta - lr * (1 / n) * grads
```

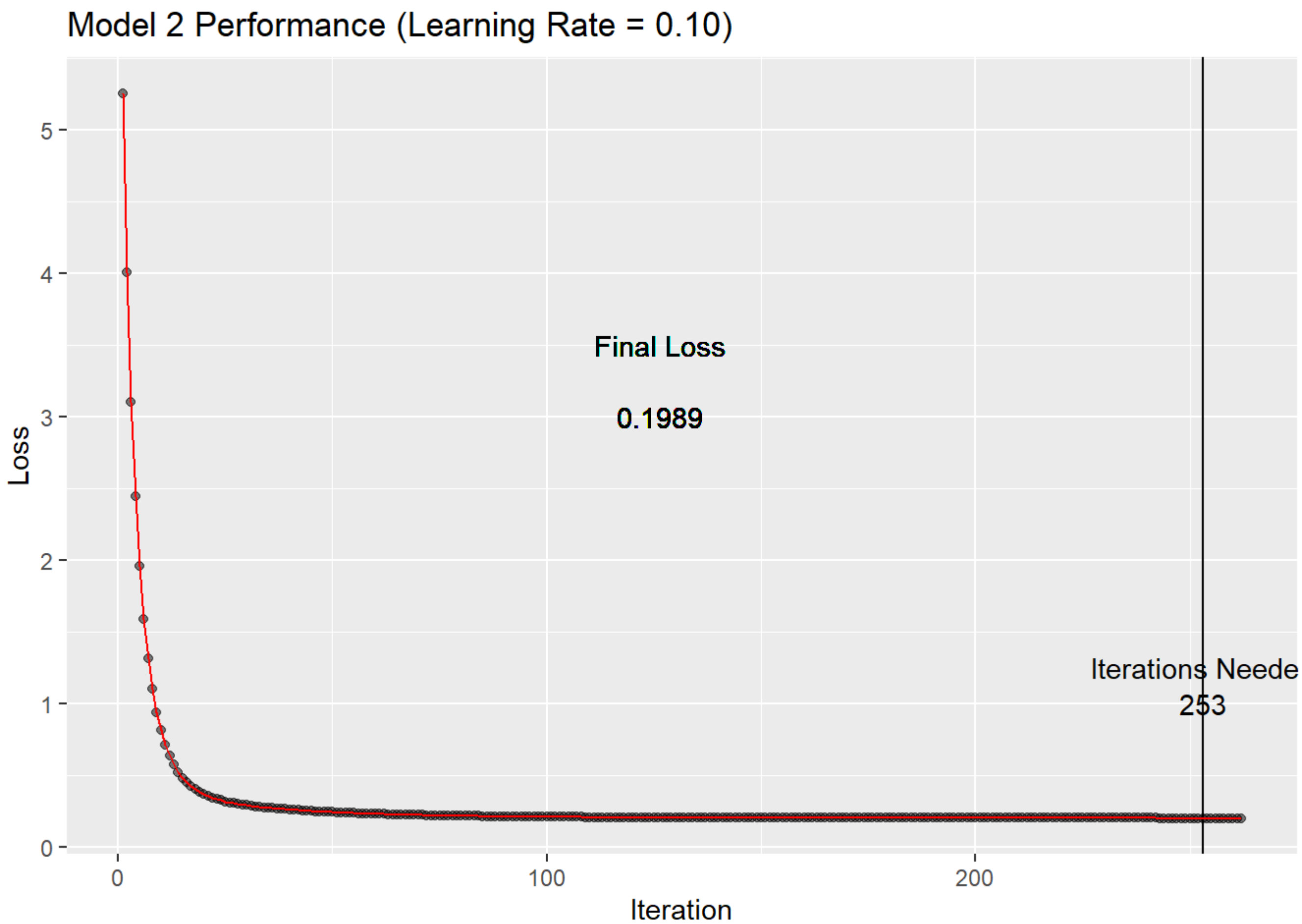
```
    # Add cost to the list
    losses[[i + 1]] <- loss

    if (round(losses[[i]], 4) <= round(loss, 4)) {
      if (counter > 6) {
        break
      } else {
        counter <- counter + 1
        sufficient_iterations <- sufficient_iterations + 1
      }
    } else {
      counter <- 0
      sufficient_iterations <- sufficient_iterations + 1
    }
  }


  sufficient_iterations <- sufficient_iterations - counter
  # return the theta (aka model weights)
  return(list(
    "coeffs" = theta,
    "losses" = losses,
    "iterations_required" = sufficient_iterations,
    "final_loss" = loss
  ))
}

# Predict function
predict <- function(x, theta) {
  ones <- rep(1, dim(x)[[1]])
  # append it to the input (this is our X0)
  X <- as.matrix(cbind(ones, x))

  return(X %*% t(theta))
}
```

```
# Model 1, lr = 0.01
model1 <- gradient_desc(train_data[, 1:8], train_data$strength, lr = 0.01, iters = 10000)

model1_weights <- t(model1$coeffs)
model1_losses <- melt(data.frame(model1$losses))
model1_losses$index <- 1:dim(model1_losses)[[1]]


# Model 2, lr = 0.10
model2 <- gradient_desc(train_data[, 1:8], train_data$strength, lr = 0.10, iters = 10000)

model2_weights <- t(model2$coeffs)
model2_losses <- melt(data.frame(model2$losses))
model2_losses$index <- 1:dim(model2_losses)[[1]]

# Model 3, lr = 0.30
model3 <- gradient_desc(train_data[, 1:8], train_data$strength, lr = 0.30, iters = 10000)

model3_weights <- t(model3$coeffs)
model3_losses <- melt(data.frame(model3$losses))
model3_losses$index <- 1:dim(model3_losses)[[1]]

# Model 4, lr = 0.50
model4 <- gradient_desc(train_data[, 1:8], train_data$strength, lr = 0.50, iters = 10000)

model4_weights <- t(model4$coeffs)
model4_losses <- melt(data.frame(model4$losses))
model4_losses$index <- 1:dim(model4_losses)[[1]]
```

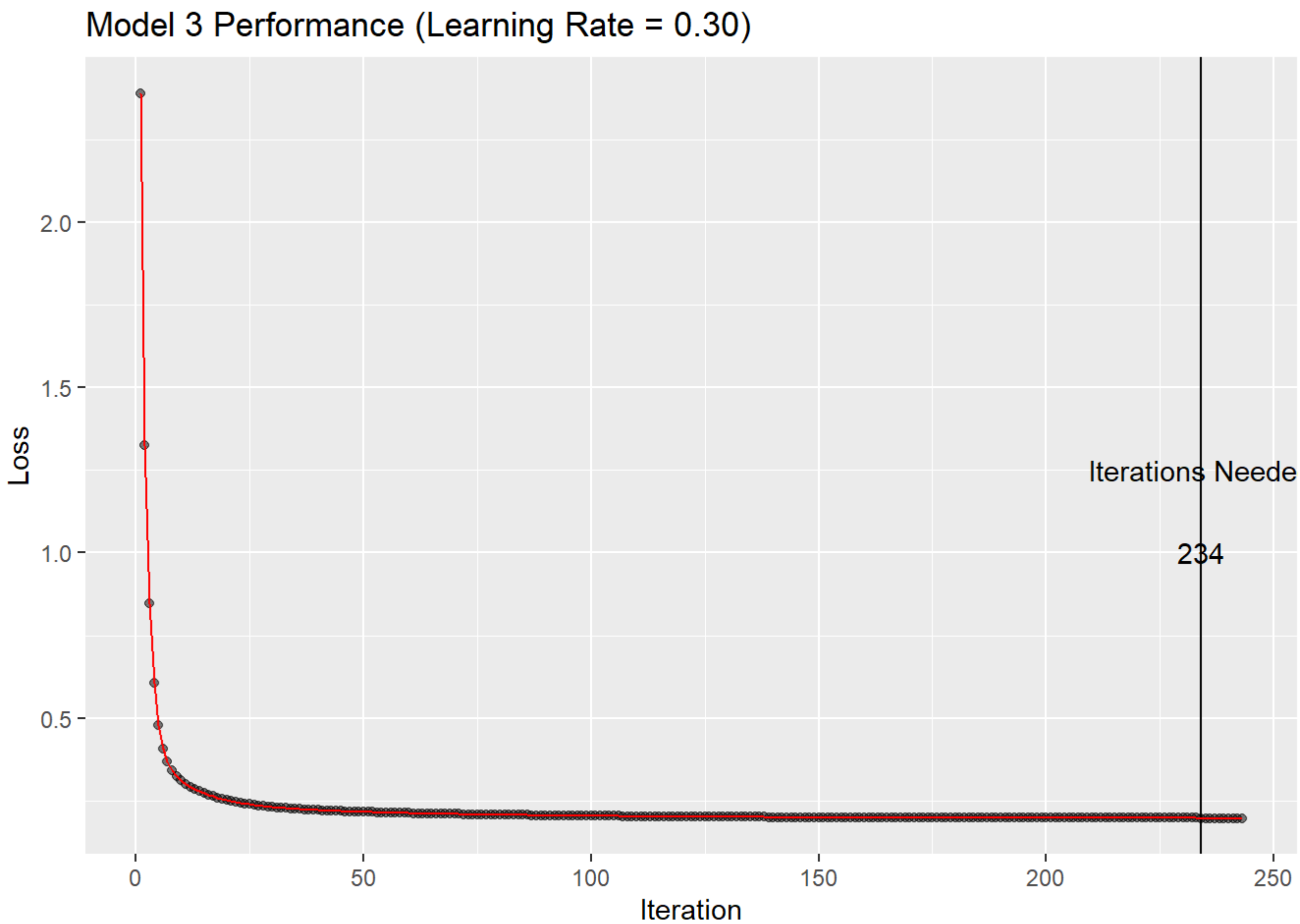
```
# Model 1
ggplot(model1_losses, aes(x = index, y = value)) +
  geom_point(alpha = 0.5) +
  geom_vline(xintercept = model1$iterations_required) +
  geom_text(x = model1$iterations_required / 2, y = 3.5, label = "Final Loss") +
  geom_text(x = model1$iterations_required / 2, y = 3, label = as.character(round(model1$fin
al_loss, 4))) +
  geom_text(
    x = model1$iterations_required,
    y = 1,
    label = as.character(model1$iterations_required),
    check_overlap = TRUE
  ) +
  geom_text(
    x = model1$iterations_required,
    y = 1.25,
    label = "Iterations Needed",
    check_overlap = TRUE
  ) +
  geom_line(color = "red") +
  labs(x = "Iteration", y = "Loss") +
  ggtitle("Model 1 Performance (Learning Rate = 0.01)")
```



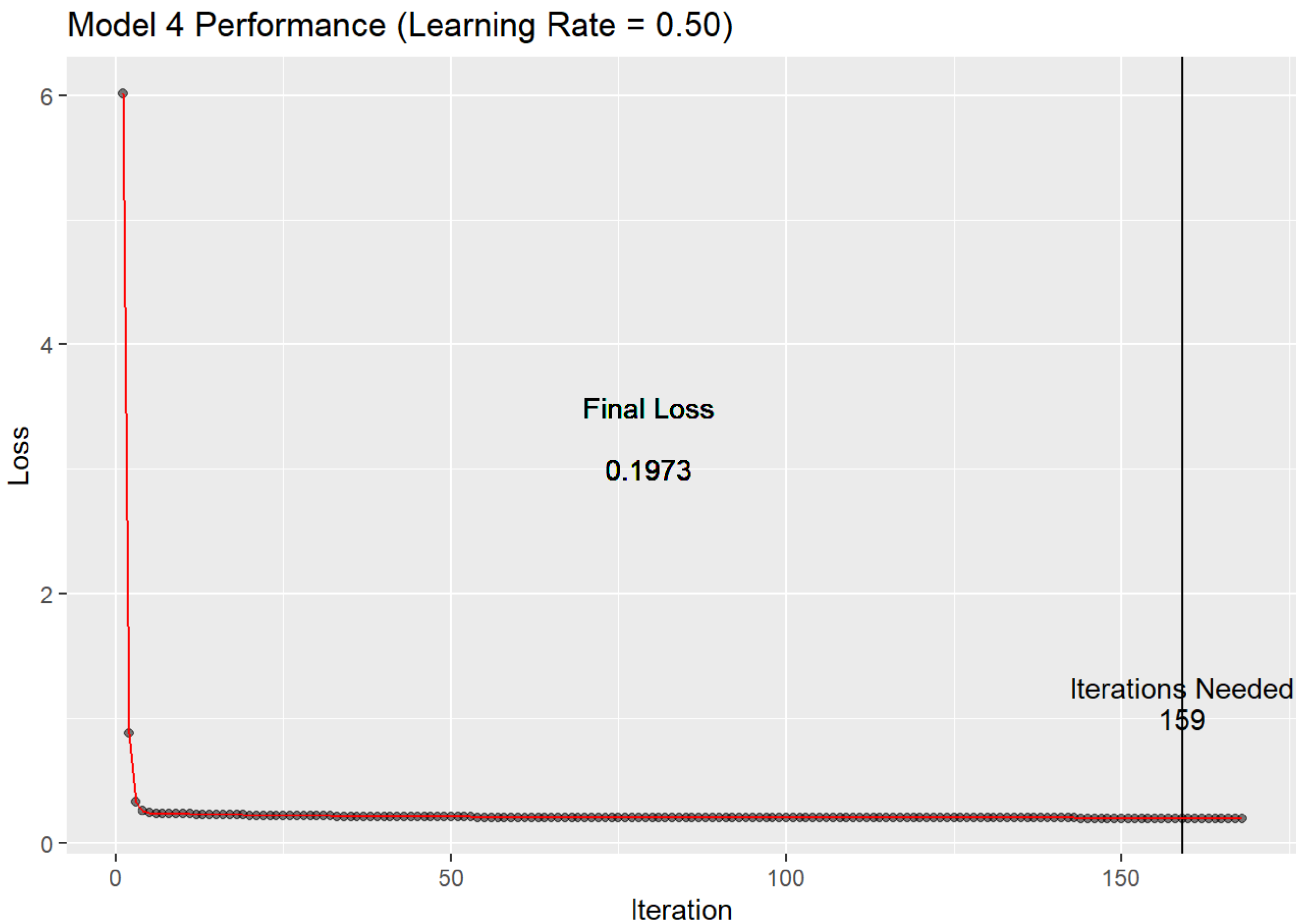

```
# Model 2
ggplot(model2_losses, aes(x = index, y = value)) +
  geom_point(alpha = 0.5) +
  geom_vline(xintercept = model2$iterations_required) +
  geom_text(x = model2$iterations_required / 2, y = 3.5, label = "Final Loss") +
  geom_text(x = model2$iterations_required / 2, y = 3, label = as.character(round(model2$fin
al_loss, 4))) +
  geom_text(
    x = model2$iterations_required,
    y = 1,
    label = as.character(model2$iterations_required),
    check_overlap = TRUE
  ) +
  geom_text(
    x = model2$iterations_required,
    y = 1.25,
    label = "Iterations Needed",
    check_overlap = TRUE
  ) +
  geom_line(color = "red") +
  labs(x = "Iteration", y = "Loss") +
  ggtitle("Model 2 Performance (Learning Rate = 0.10)")
```




```
# Model 3
ggplot(model3_losses, aes(x = index, y = value)) +
  geom_point(alpha = 0.5) +
  geom_vline(xintercept = model3$iterations_required) +
  geom_text(x = model3$iterations_required / 2, y = 3.5, label = "Final Loss") +
  geom_text(x = model3$iterations_required / 2, y = 3, label = as.character(round(model3$fin
al_loss, 4))) +
  geom_text(
    x = model3$iterations_required,
    y = 1,
    label = as.character(model3$iterations_required),
    check_overlap = TRUE
  ) +
  geom_text(
    x = model3$iterations_required,
    y = 1.25,
    label = "Iterations Needed",
    check_overlap = TRUE
  ) +
  geom_line(color = "red") +
  labs(x = "Iteration", y = "Loss") +
  ggtitle("Model 3 Performance (Learning Rate = 0.30)")
```




```
    # Model 4
    ggplot(model4_losses, aes(x = index, y = value)) +
      geom_point(alpha = 0.5) +
      geom_vline(xintercept = model4$iterations_required) +
      geom_text(x = model4$iterations_required / 2, y = 3.5, label = "Final Loss") +
      geom_text(x = model4$iterations_required / 2, y = 3, label = as.character(round(model4$fin
al_loss, 4))) +
      geom_text(
        x = model4$iterations_required,
        y = 1,
        label = as.character(model4$iterations_required),
        check_overlap = TRUE
      ) +
      geom_text(
        x = model4$iterations_required,
        y = 1.25,
        label = "Iterations Needed",
        check_overlap = TRUE
      ) +
      geom_line(color = "red") +
      labs(x = "Iteration", y = "Loss") +
      ggtitle("Model 4 Performance (Learning Rate = 0.50)")
```




```
cat("Number of iterations required for each model are :\n")
```

```
## Number of iterations required for each model are :
```

```
cat("Model 1:", as.character(model1$iterations_required), "\n")
```

```
## Model 1: 1174
```

```
cat("Model 2:", as.character(model2$iterations_required), "\n")
```

```
## Model 2: 253
```

```
cat("Model 3:", as.character(model3$iterations_required), "\n")
```

```
## Model 3: 234
```

```
cat("Model 4:", as.character(model4$iterations_required), "\n")
```

```
## Model 4: 159
```

```
<!-- As observed, the model converges faster at the minimum loss as the learning rate increases.
-->
```

```
    # We define the Mean Error function
    ME <- function(y_hat, y) {
      sum(y - y_hat) / length(y)
    }

    # We define the Mean Percentage Error Function
    MPE <- function(y_hat, y) {
      (sum((y - y_hat) / y)) / length(y)
    }
```

```
<!-- Model: -  -->
```

```
    model1_predictions <- predict(validation_data[, 1:8], model1_weights)

    cat("----Model 1 Summary ----\n")
```

```
## ----Model 1 Summary ----
```

```
    cat("MAE:", MAE(model1_predictions, validation_data[, 9]), "\n")
```

```
## MAE: 0.4841188
```

```
    cat("RMSE:", RMSE(model1_predictions, validation_data[, 9]), "\n")
```

```
## RMSE: 0.6288409
```

```
    cat("ME:", ME(model1_predictions, validation_data[, 9]), "\n")
```

```
## ME: 0.005420344
```

```
    cat("MPE:", MPE(model1_predictions, validation_data[, 9]), "\n")
```

```
## MPE: 1.112925
```

```
    cat("MPAE", MAPE(model1_predictions, validation_data[, 9]), "\n")
```

```
## MPAE 2.007763
```

```
model2_predictions <- predict(validation_data[, 1:8], model2_weights)

cat("----Model 2 Summary ---- \n")
```

```
## ----Model 2 Summary ----
```

```
cat("MAE:", MAE(model2_predictions, validation_data[, 9]), "\n")
```

```
## MAE: 0.4767761
```

```
cat("RMSE:", RMSE(model2_predictions, validation_data[, 9]), "\n")
```

```
## RMSE: 0.6182023
```

```
cat("ME:", ME(model2_predictions, validation_data[, 9]), "\n")
```

```
## ME: 0.008696758
```

```
cat("MPE:", MPE(model2_predictions, validation_data[, 9]), "\n")
```

```
## MPE: 0.907251
```

```
cat("MPAE", MAPE(model2_predictions, validation_data[, 9]), "\n")
```

```
## MPAE 1.811604
```

```
model3_predictions <- predict(validation_data[, 1:8], model3_weights)

cat("----Model 3 Summary ----\n")
```

```
## ----Model 3 Summary ----
```

```
cat("MAE:", MAE(model3_predictions, validation_data[, 9]), "\n")
```

```
## MAE: 0.4774931
```

```
cat("RMSE:", RMSE(model3_predictions, validation_data[, 9]), "\n")
```

```
## RMSE: 0.6120372
```

```
cat("ME:", ME(model3_predictions, validation_data[, 9]), "\n")
```

```
## ME: 0.008683658
```

```
cat("MPE:", MPE(model3_predictions, validation_data[, 9]), "\n")
```

```
## MPE: 0.6390469
```

```
cat("MPAE", MAPE(model3_predictions, validation_data[, 9]), "\n")
```

```
## MPAE 1.98074
```

```
model4_predictions <- predict(validation_data[, 1:8], model4_weights)

cat("----Model 4 Summary ----\n")
```

```
## ----Model 4 Summary ----
```

```
cat("MAE:", MAE(model4_predictions, validation_data[, 9]), "\n")
```

```
## MAE: 0.4768407
```

```
cat("RMSE:", RMSE(model4_predictions, validation_data[, 9]), "\n")
```

```
## RMSE: 0.6120375
```

```
cat("ME:", ME(model4_predictions, validation_data[, 9]), "\n")
```

```
## ME: 0.008714432
```

```
cat("MPE:", MPE(model4_predictions, validation_data[, 9]), "\n")
```

```
## MPE: 0.6677231
```

```
cat("MPAE", MAPE(model4_predictions, validation_data[, 9]), "\n")
```

```
## MPAE 1.945207
```

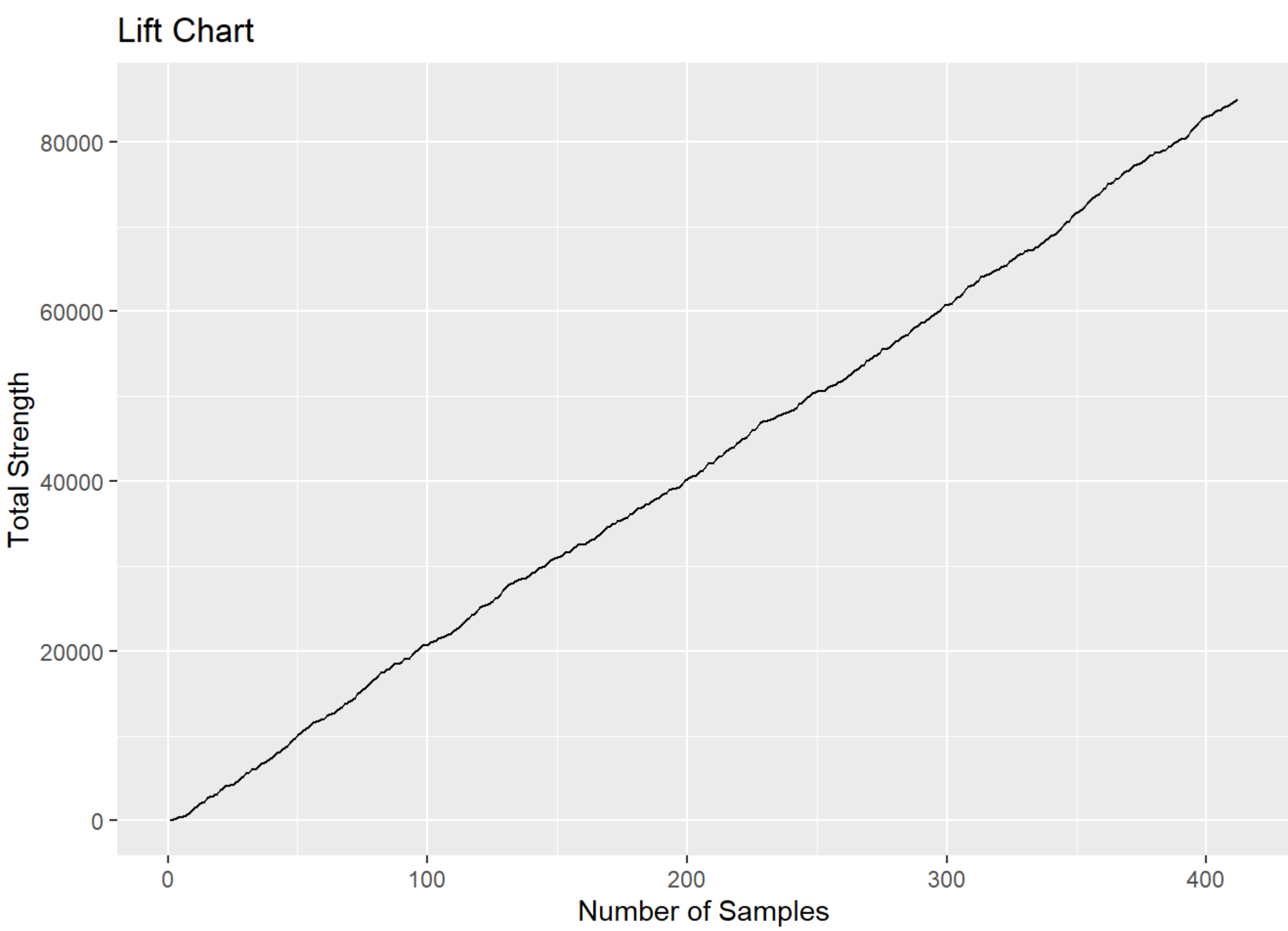
```
<!-- Learning rate has no effect on the accuracy -->


<!-- Calculating the correlation between predicted strength and actual strength -->
```

```
cat("The correlation is :", cor(model1_predictions, validation_data[, 9]), "\n")
```

```
## The correlation is : 0.7599455
```

```
Plotting a lift chart
```

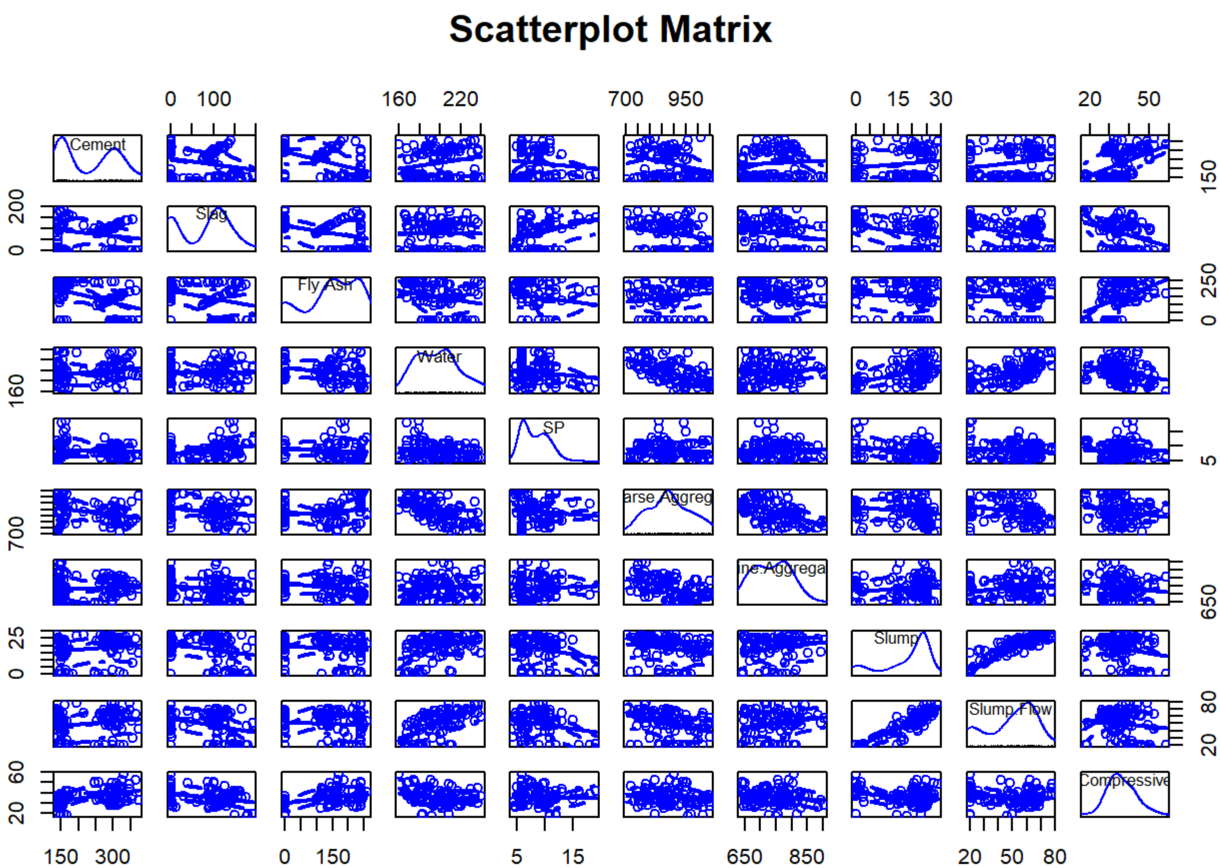
```
# Create a temp data frame to calculate the sumulative strength
temp <- data.frame("strength" = order(validation_data[, 9]))
temp$cumstrength <- cumsum(temp$strength)
temp$samples <- 1:dim(temp)[[1]]

# Plot the lift chart
ggplot(temp, aes(x = samples, y = cumstrength)) +
  geom_line() +
  labs(x = "Number of Samples", y = "Total Strength") +
  ggtitle("Lift Chart")
```



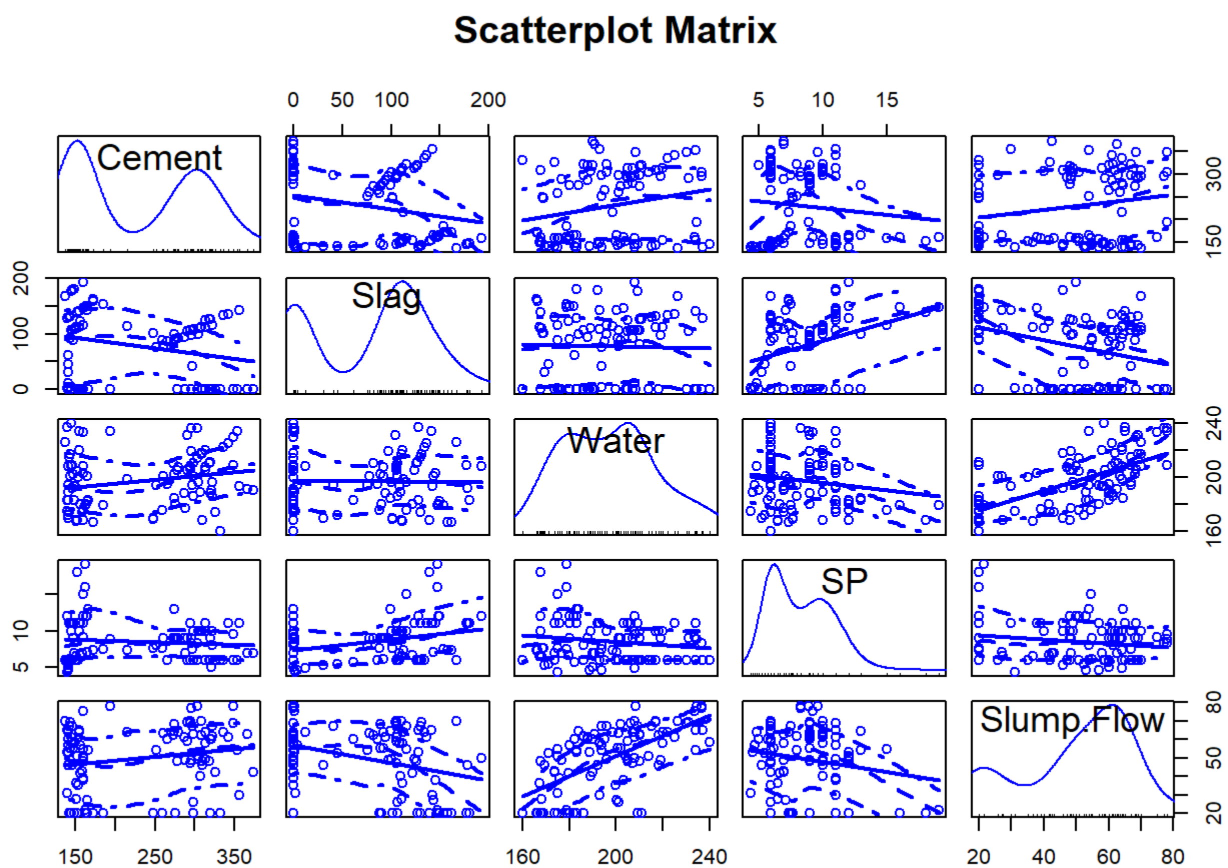
```r
# Delete all environment variables
rm(list = ls())
```

```
---
```

```r
conc_slump <- readxl::read_xlsx("Concrete Slump Test Data.xlsx", sheet = "Concrete slump")
conc_slump <- conc_slump[, 2:11]

# Let's plot the scatterplot matrix
scatterplotMatrix(conc_slump, main = "Scatterplot Matrix")
```




```r
# Since the above matrix is hard to interpret, we only plot it for a select
# variables
scatterplotMatrix(~ Cement + Slag + Water + SP + `Slump Flow`,
  data = conc_slump,
  main = "Scatterplot Matrix"
)
```




```
fit1 <- lm(`Slump Flow` ~ Water + `Coarse Aggregate` + `Fine Aggregate`, data = conc_slump)

summary(fit1)
```

```
##
## Call:
## lm(formula = `Slump Flow` ~ Water + `Coarse Aggregate` + `Fine Aggregate`,
##     data = conc_slump)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -30.163  -8.837   1.799   9.869  24.383
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -162.70980   44.17173  -3.684 0.000375 ***
## Water                 0.64760    0.08495   7.623 1.53e-11 ***
## `Coarse Aggregate`    0.04545    0.02211   2.055 0.042476 *
## `Fine Aggregate`      0.06011    0.02480   2.424 0.017165 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.37 on 99 degrees of freedom
## Multiple R-squared:  0.4376, Adjusted R-squared:  0.4205
## F-statistic: 25.67 on 3 and 99 DF,  p-value: 2.28e-12
```

```
fit2 <- lm(`Slump Flow` ~ Water + Slag + `Fine Aggregate`, data = conc_slump)

summary(fit2)
```

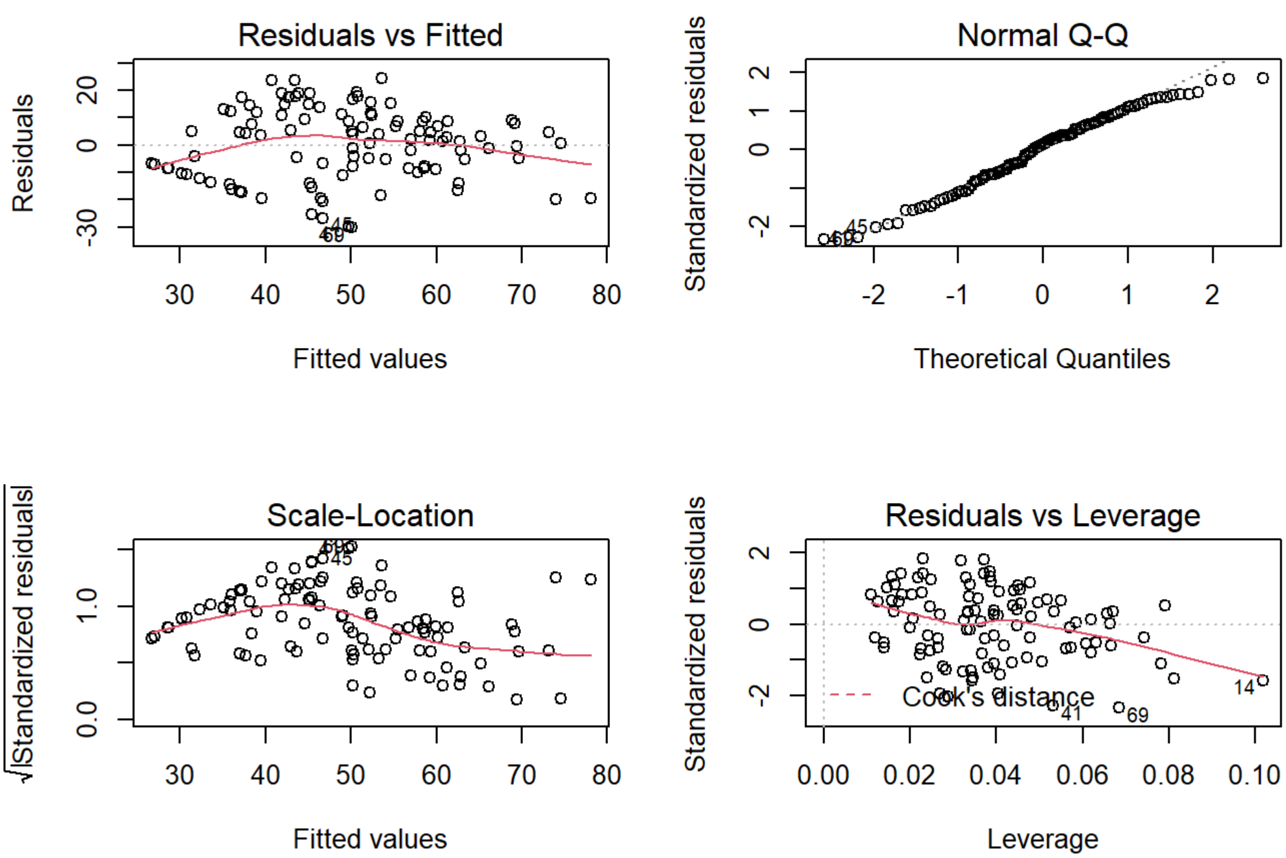
```
##
## Call:
## lm(formula = `Slump Flow` ~ Water + Slag + `Fine Aggregate`,
##     data = conc_slump)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -32.470 -10.428   2.035   9.123  22.867
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -62.61966   18.59310  -3.368  0.00108 **
## Water              0.53605    0.06221   8.617 1.12e-13 ***
## Slag              -0.08683    0.02101  -4.133 7.51e-05 ***
## `Fine Aggregate`   0.01799    0.02018   0.892  0.37477
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.61 on 99 degrees of freedom
## Multiple R-squared:  0.4998, Adjusted R-squared:  0.4847
## F-statistic: 32.98 on 3 and 99 DF,  p-value: 7.292e-15
```

```
<!-- Fitting a quadratic model -->
```

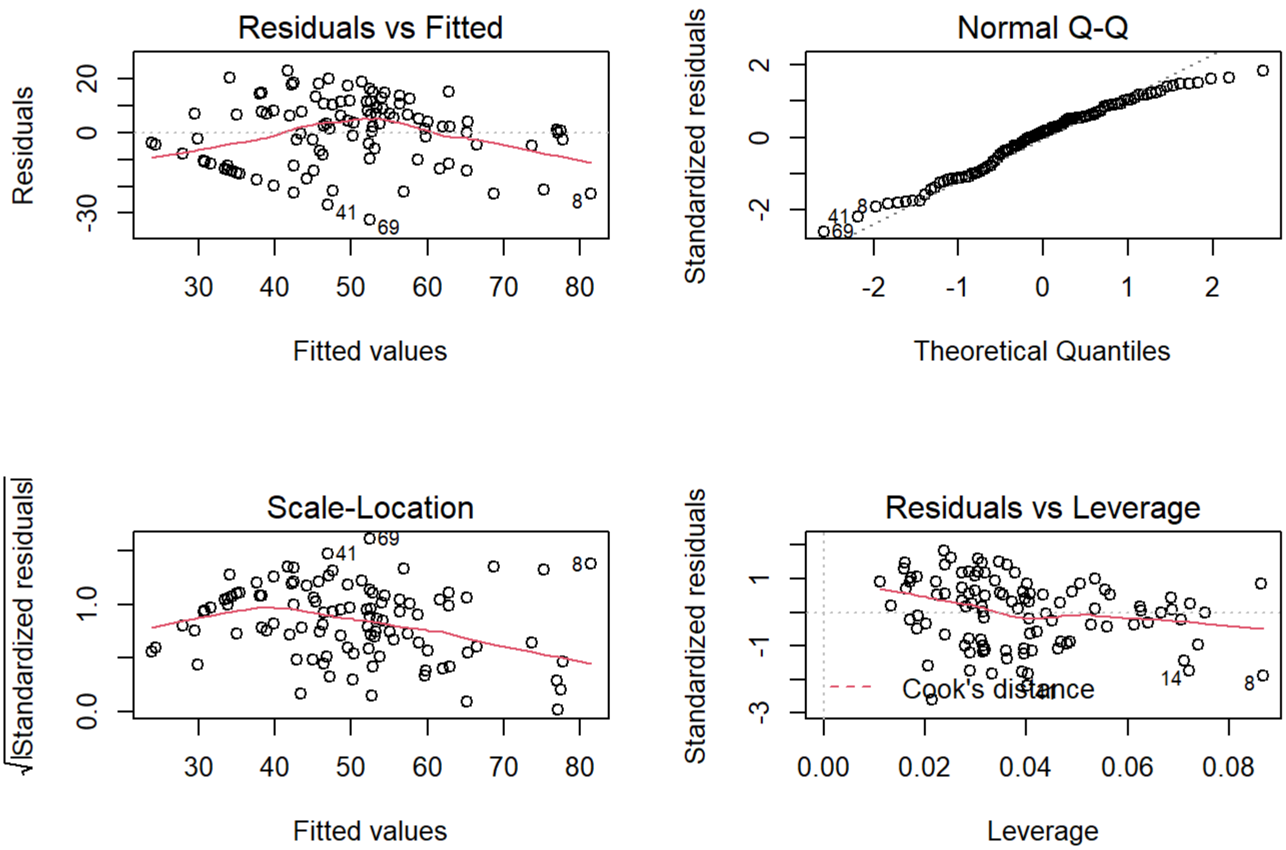
```
fit3 <- lm(`Slump Flow` ~ (Water^2) + Water + Slag, data = conc_slump)

summary(fit3)
```

```
##
## Call:
## lm(formula = `Slump Flow` ~ (Water^2) + Water + Slag, data = conc_slump)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -32.687 -10.746   2.010   9.224  23.927
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -50.26656   12.38669  -4.058 9.83e-05 ***
## Water         0.54224    0.06175   8.781 4.62e-14 ***
## Slag         -0.09023    0.02064  -4.372 3.02e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.6 on 100 degrees of freedom
## Multiple R-squared:  0.4958, Adjusted R-squared:  0.4857
## F-statistic: 49.17 on 2 and 100 DF,  p-value: 1.347e-15
```

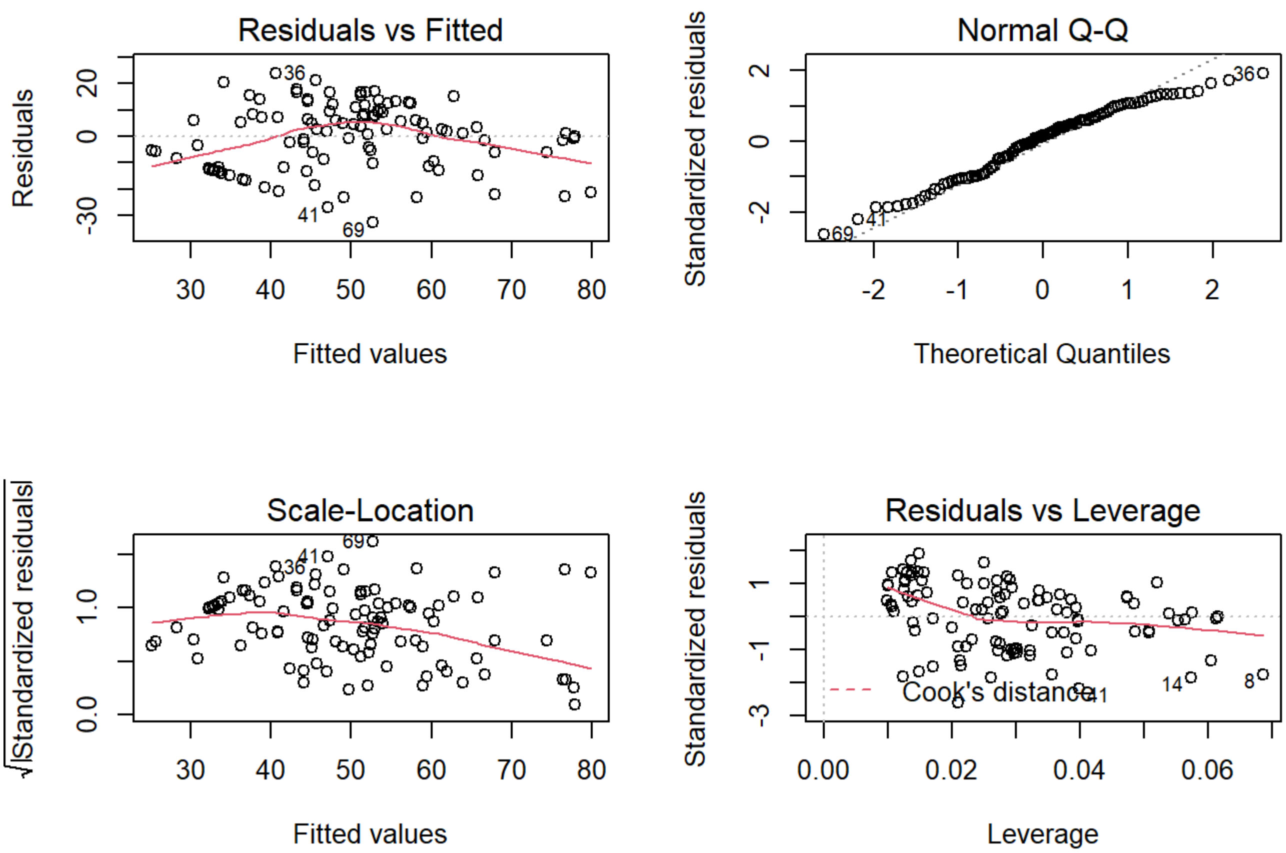
```
# Model 1
par(mfrow = c(2, 2))
plot(fit1)
```



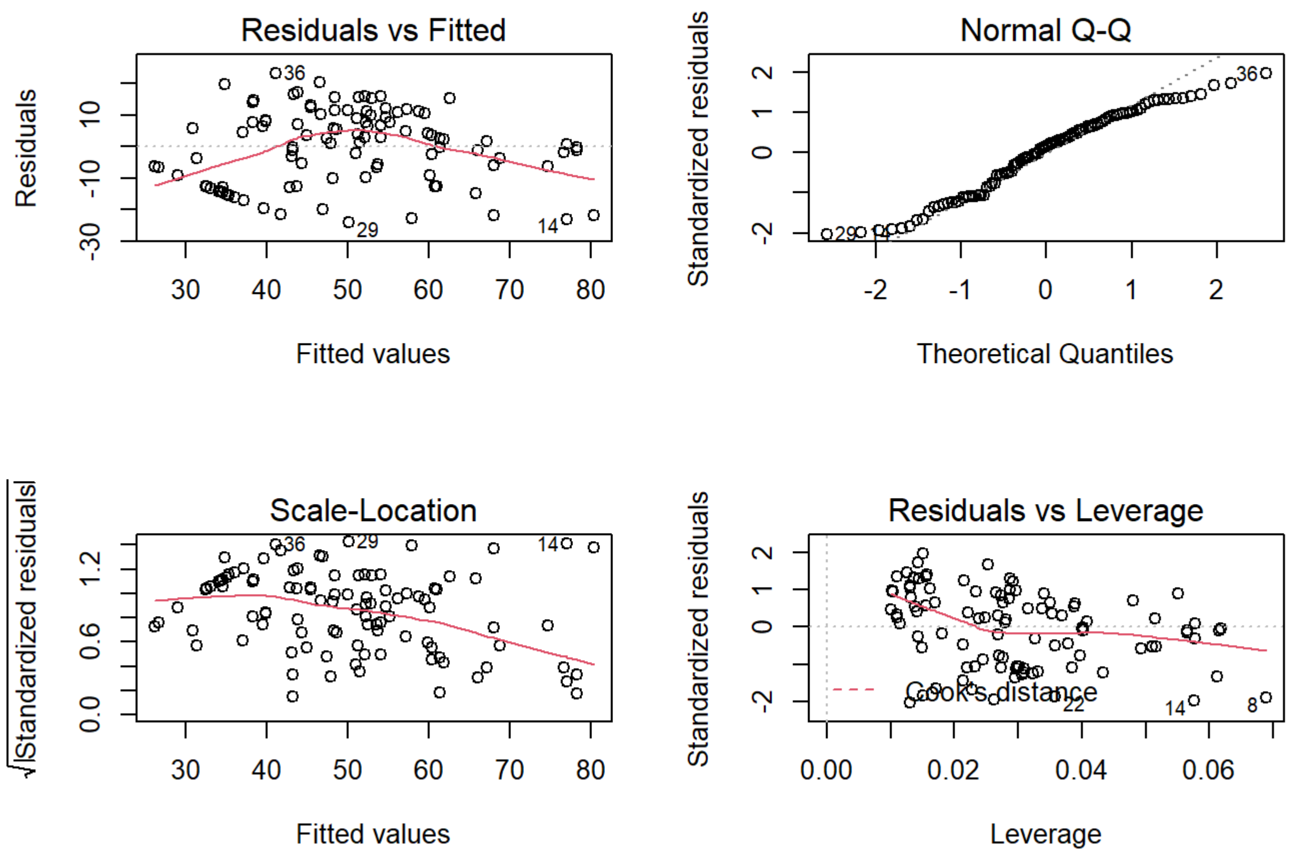
```
# Model 2
par(mfrow = c(2, 2))
plot(fit2)
```



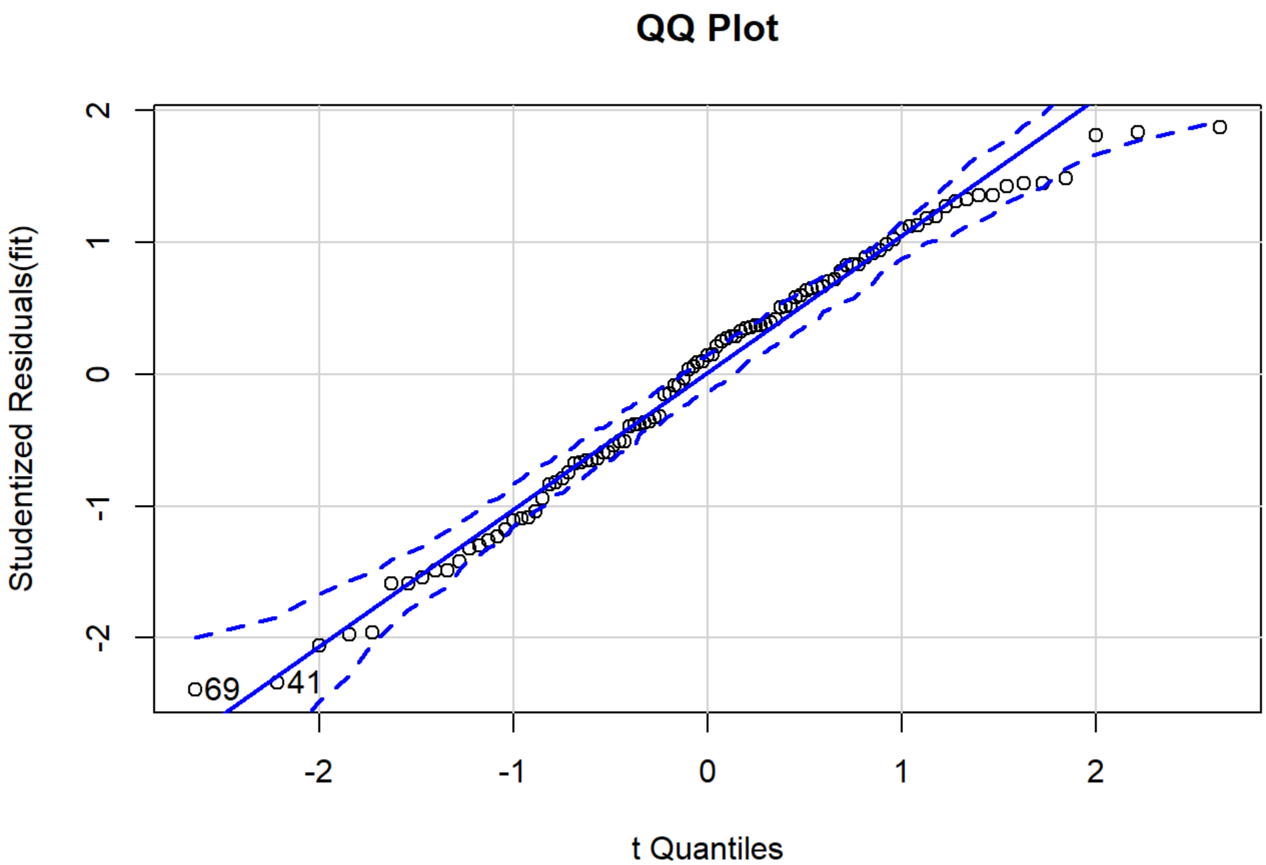

```
# Model 3
par(mfrow = c(2, 2))
plot(fit3)
```




```
fit3 <- lm(`Slump Flow` ~ (Water^2) + Water + Slag, data = conc_slump[-c(41, 69), ])

# Model 3
par(mfrow = c(2, 2))
plot(fit3)
```




```
fit <- lm(`Slump Flow` ~ Water + `Coarse Aggregate` + `Fine Aggregate`, data = conc_slump)

qqPlot(fit, labels = rownames(df), id.method = "identify", simulate = TRUE, main = "QQ Plot"
)
```



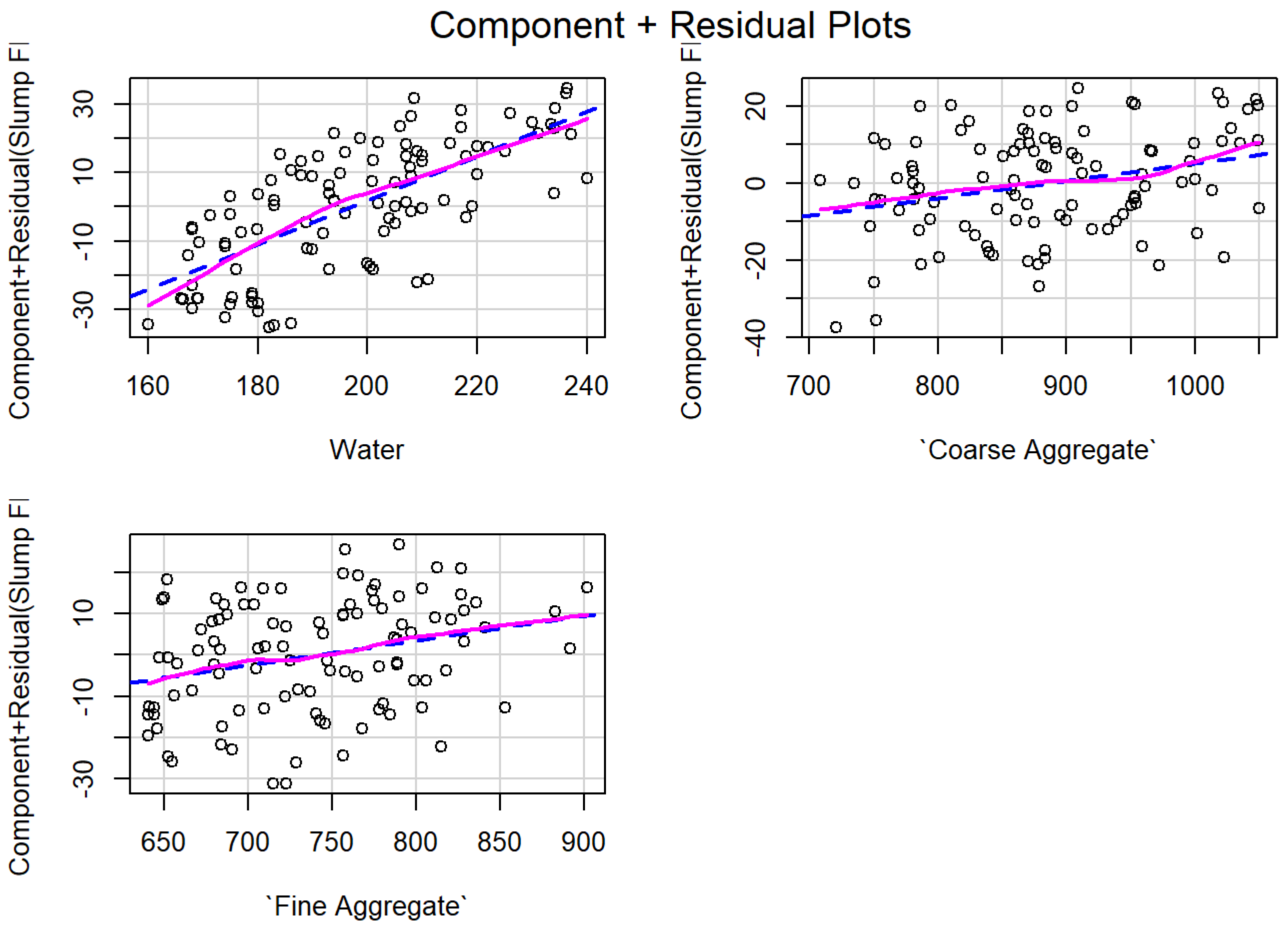

```
## [1] 41 69
```

```
durbinWatsonTest(fit)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1      0.06668866      1.830473   0.344
##  Alternative hypothesis: rho != 0
```

```
<!-- Since the p-value is insignificant, there is no auto correlation and hence and independence of errors. -->
```

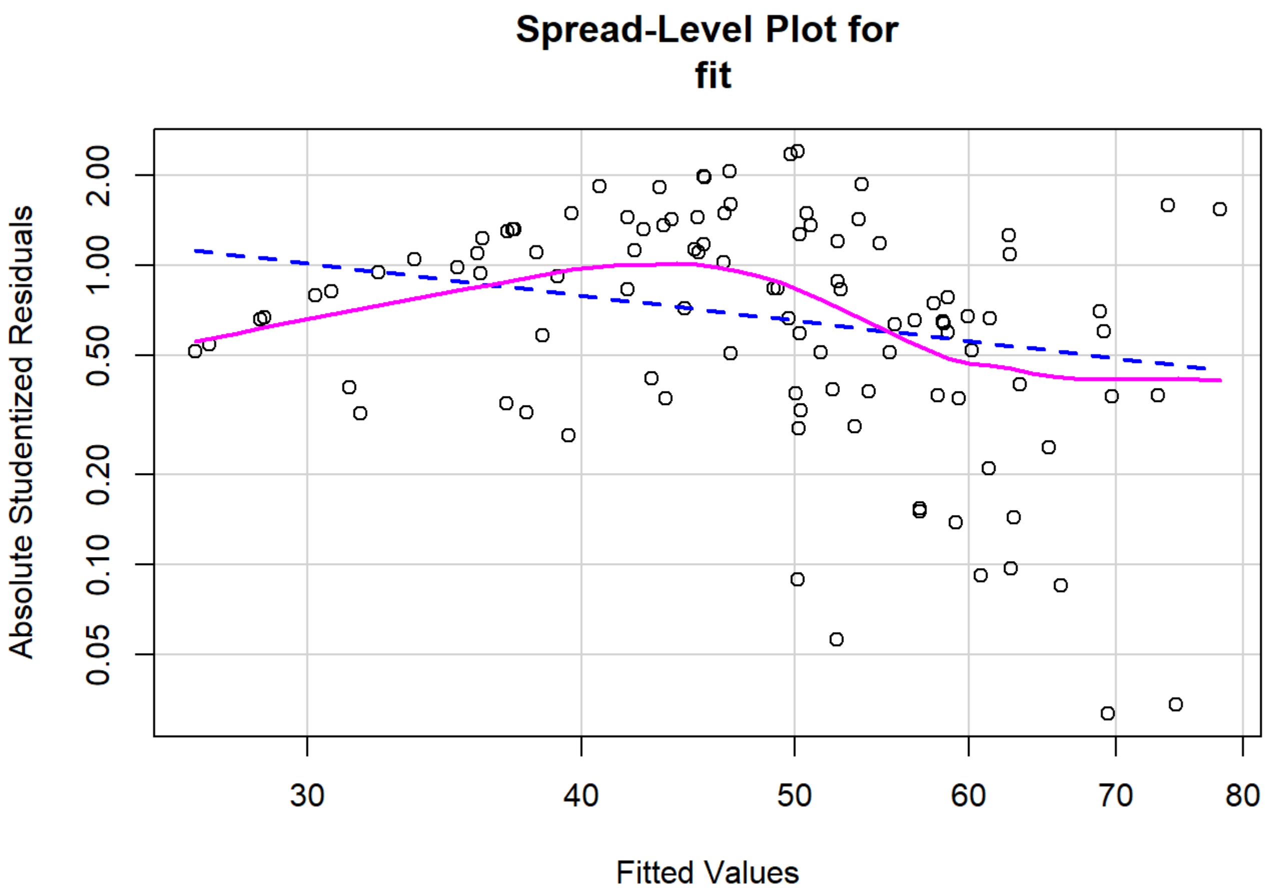
```
crPlots(fit)
```




```
ncvTest(fit)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 1.533171, Df = 1, p = 0.21564
```

```
spreadLevelPlot(fit)
```




```
##
## Suggested power transformation:  1.866028
```

```
outlierTest(fit)
```

```
## No Studentized residuals with Bonferroni p < 0.05
## Largest |rstudent|:
##     rstudent unadjusted p-value Bonferroni p
## 69 -2.391905           0.018669           NA
```

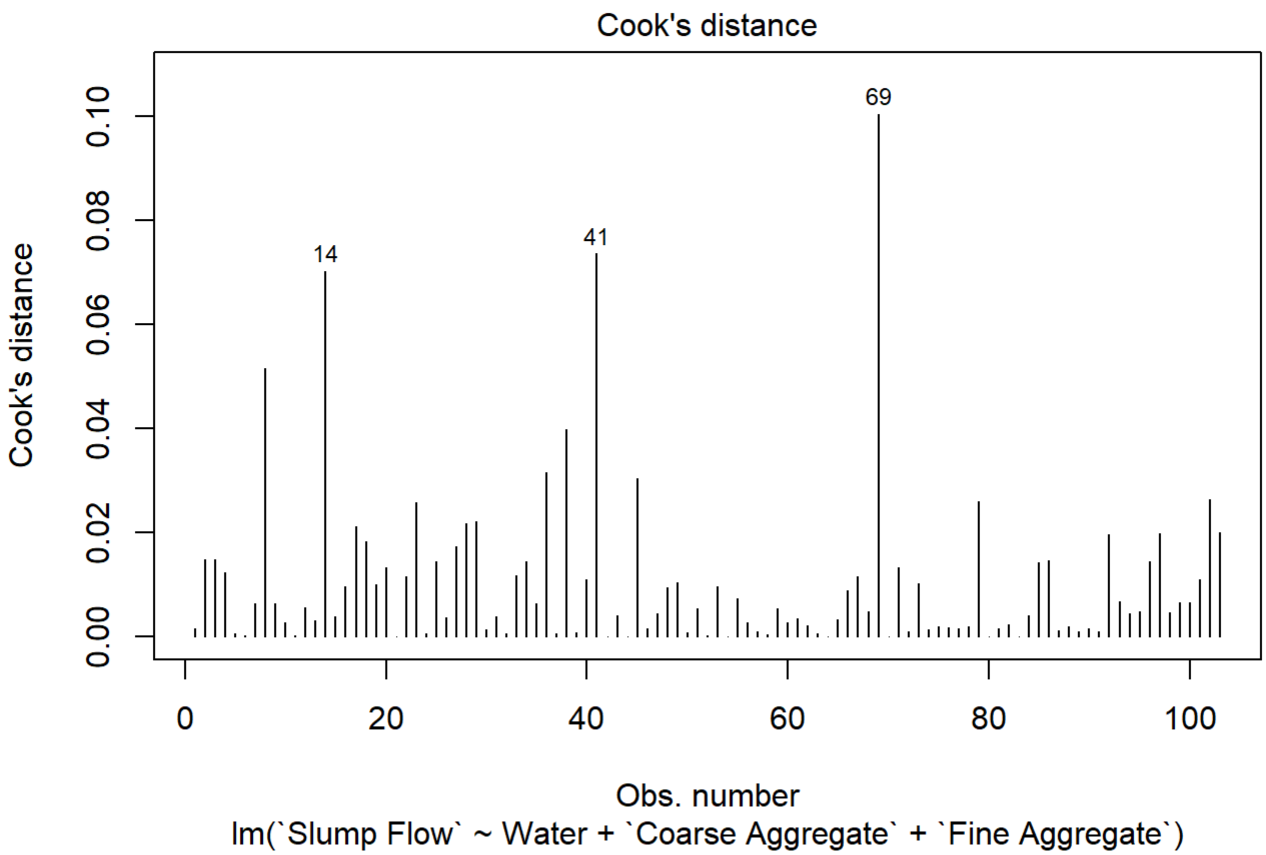
```
<!-- Let's search for High Leverage points -->
```

```
hat.plot <- function(fit) {
  p <- length(coefficients(fit))
  n <- length(fitted(fit))
  plot(hatvalues(fit),
    main = "Index Plot of Hat Values"
  )
  abline(h = c(2, 3) * p / n, col = "red", lty = 2)
  identify(1:n, hatvalues(fit), names(hatvalues(fit)))
}

hat.plot(fit)
```

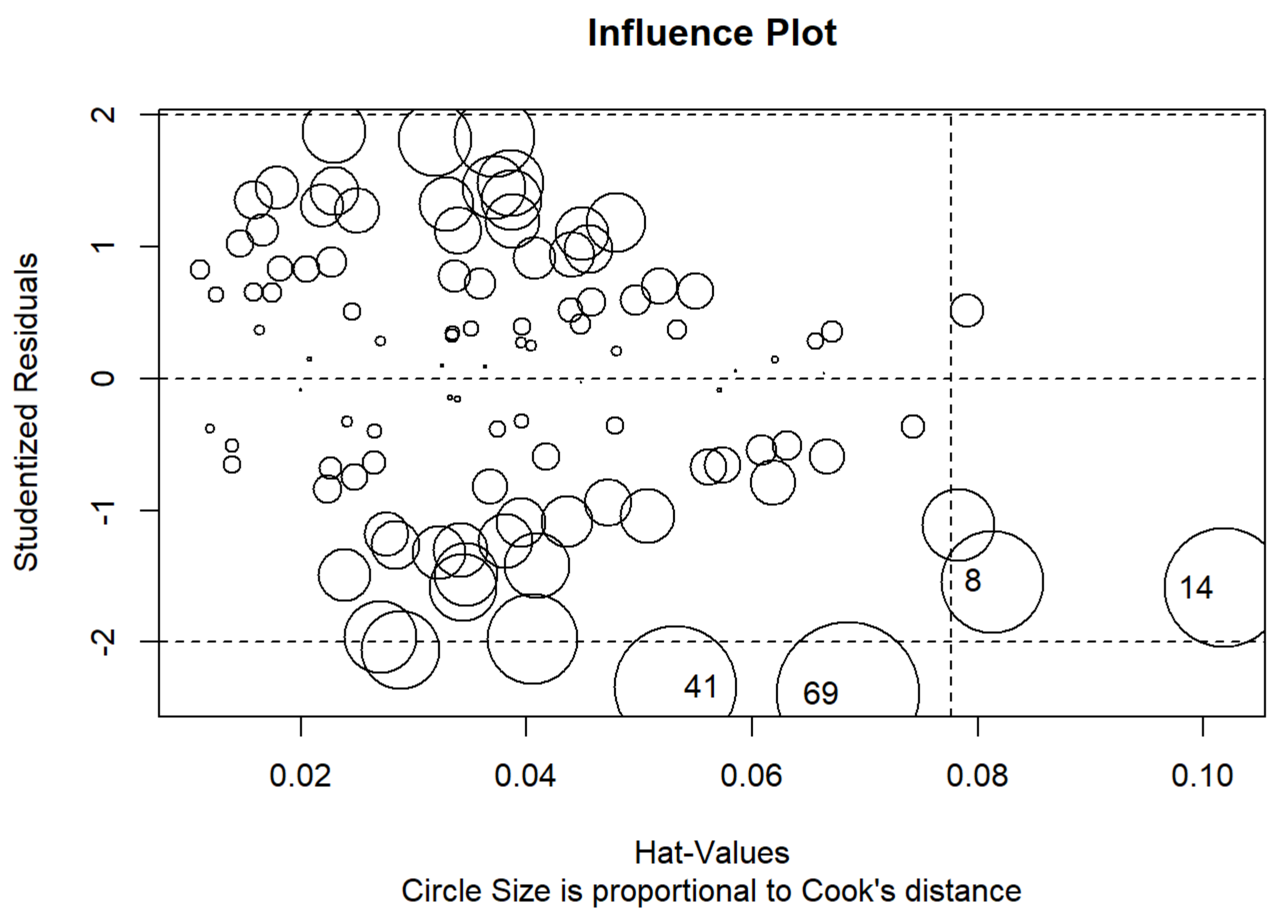

```
## integer(0)
```

```
<!-- We can see that points 8, 12, 14, and 78 are unusual when it comes to their predicted value
s. -->
```

```
    cutoff <- 4 / (nrow(df) - length(fit$coefficients) - 2)
    plot(fit, which = 4, cook.levels = cutoff)
    abline(h = cutoff, lty = 2, col = "red")
```




```
influencePlot(fit,
  main = "Influence Plot",
  sub = "Circle Size is proportional to Cook's distance"
)
```




```
##      StudRes        Hat      CookD
## 8  -1.537566 0.08127700 0.05157597
## 14 -1.586084 0.10183117 0.07022906
## 41 -2.340784 0.05315091 0.07356555
## 69 -2.391905 0.06853364 0.10044585
```

```
    fit <- lm(`Slump Flow` ~ Water + `Coarse Aggregate` + `Fine Aggregate`, data = conc_slump[-c
(14, 41), ])
    fit2 <- lm(`Slump Flow` ~ Water + Slag + `Coarse Aggregate` + `Fine Aggregate`, data = conc_
slump[-c(14, 41), ])
    fit3 <- lm(`Slump Flow` ~ (Water^2) + Water + Slag, data = conc_slump[-c(14, 41), ])
```

```
    anova(fit2, fit)
```

```
## Analysis of Variance Table
##
## Model 1: `Slump Flow` ~ Water + Slag + `Coarse Aggregate` + `Fine Aggregate`
## Model 2: `Slump Flow` ~ Water + `Coarse Aggregate` + `Fine Aggregate`
##   Res.Df   RSS Df Sum of Sq      F    Pr(>F)
## 1     96 14491
## 2     97 16353 -1   -1861.5 12.332 0.0006804 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(fit2, fit3)
```

```
## Analysis of Variance Table
##
## Model 1: `Slump Flow` ~ Water + Slag + `Coarse Aggregate` + `Fine Aggregate`
## Model 2: `Slump Flow` ~ (Water^2) + Water + Slag
##   Res.Df   RSS Df Sum of Sq      F Pr(>F)
## 1     96 14491
## 2     98 14566 -2   -74.709 0.2475 0.7813
```

```
AIC(fit, fit2, fit3)
```

```
##      df      AIC
## fit   5 810.4155
## fit2  6 800.2096
## fit3  4 796.7290
```

```
summary(fit3)
```

```
##
## Call:
## lm(formula = `Slump Flow` ~ (Water^2) + Water + Slag, data = conc_slump[-c(14,
##     41), ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -33.736  -9.846   1.477   9.286  23.750
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -55.78057   12.15765  -4.588 1.33e-05 ***
## Water         0.57170    0.06086   9.393 2.51e-15 ***
## Slag         -0.08749    0.02041  -4.287 4.24e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.19 on 98 degrees of freedom
## Multiple R-squared:  0.5237, Adjusted R-squared:  0.514
## F-statistic: 53.87 on 2 and 98 DF,  p-value: < 2.2e-16
```

```
predictions <- predict(fit3, conc_slump)

head(predictions)
```

```
##        1        2        3        4        5        6
## 57.10139 34.08842 33.60422 33.60422 60.19356 51.91536
```

```
rm(list = ls())
```

```
<!-- We can infer from the model coefficients Water is the most important predictor in calculati
ng the value of the Slump Flow. 1 kg per M cube change in Water results to 0.57 cm change in the
Slump Flow. Slag is a less important predictor. -->



---
```

```
insurance <- read.csv("insurance.csv", stringsAsFactors = TRUE)

ins <- read.csv("insurance.csv", stringsAsFactors = TRUE)
```

```
<!-- Summary Statistics -->
```

```
mean(insurance$charges)
```

```
## [1] 13270.42
```

```
median(insurance$charges)
```

```
## [1] 9382.033
```

```
min(insurance$charges)
```

```
## [1] 1121.874
```

```
max(insurance$charges)
```

```
## [1] 63770.43
```

```
quantile(insurance$charges, 0.25)
```

```
##      25%
## 4740.287
```

```
quantile(insurance$charges, 0.75)
```

```
##      75%
## 16639.91
```

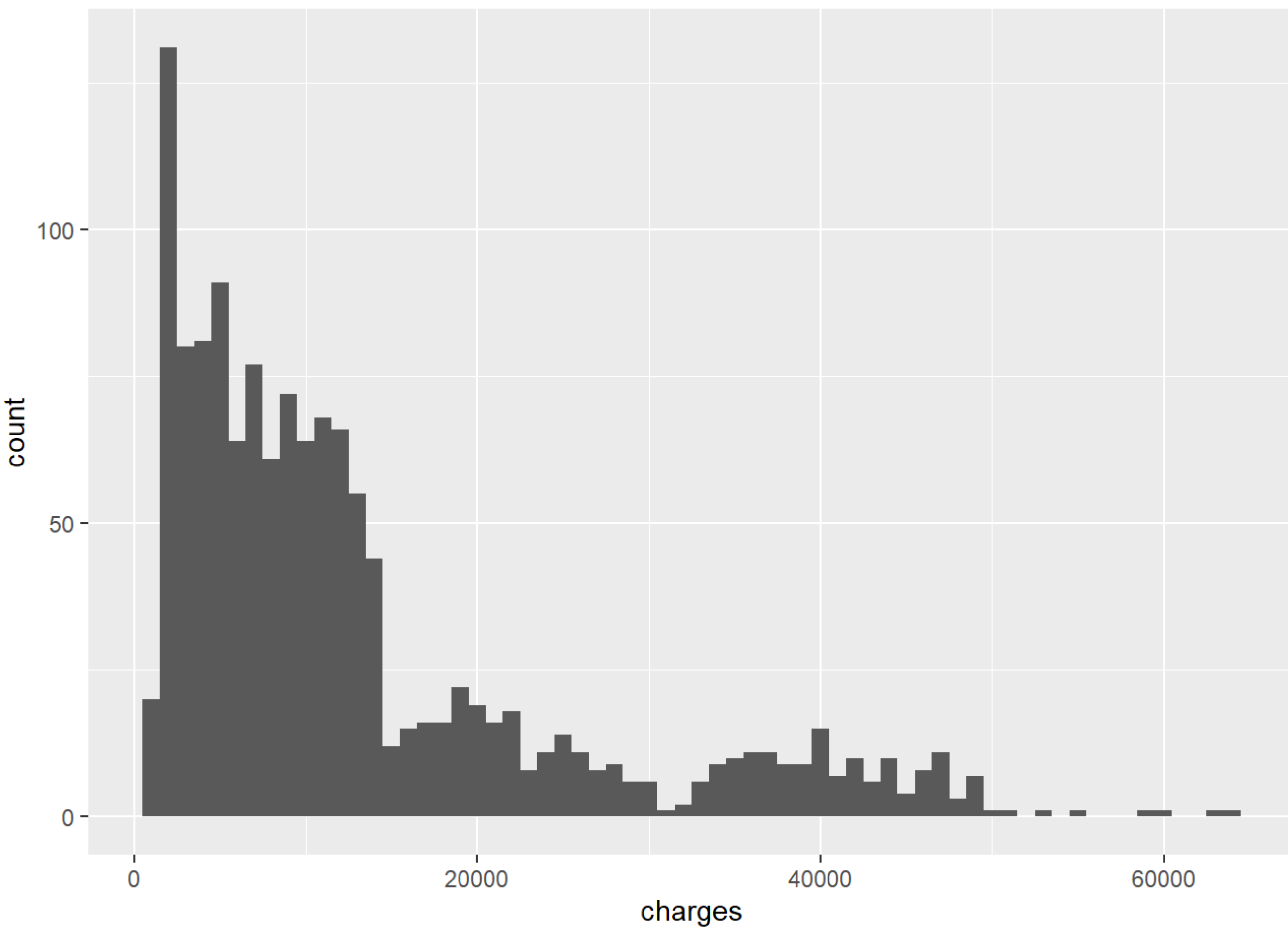
```
skewness(insurance$charges)
```

```
## [1] 1.51418
```

```
kurtosis(insurance$charges)
```

```
## [1] 4.595821
```

```
ggplot(insurance, aes(x = charges)) +
  geom_histogram(binwidth = 1000)
```

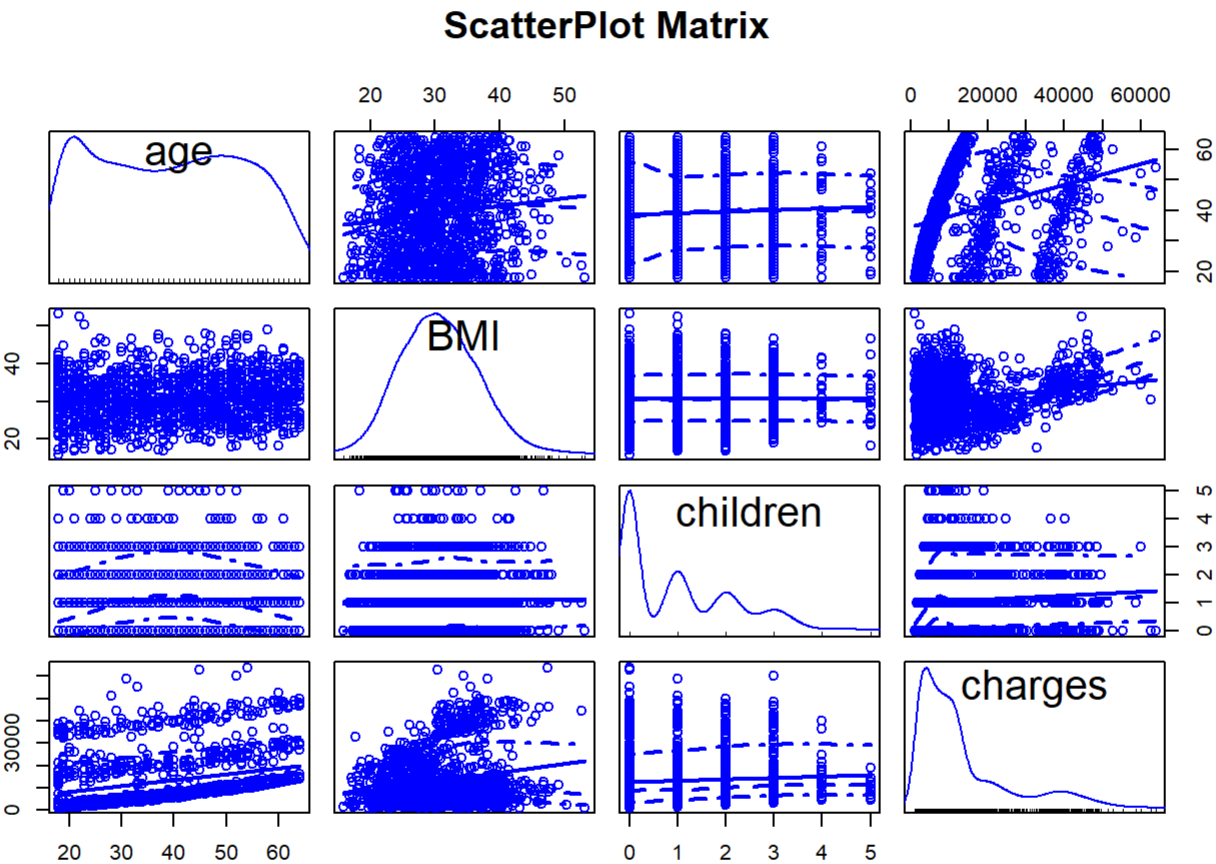


```
<!-- Interpretation-  -->


<!-- the kurtosis value of 4.6 tells us that the data has a heavier tail than the normal distrib
ution. -->

<!--    The summary statistics, namely the mean and median indicate skewness -->
<!-- in the dependent variable and the skewness of 1.51418 tells us that it is highly skewed -->

<!-- the histogram reinforces the above my showing a left skewed distribution with a heavy right
tail. -->
```

```r
    attach(insurance)

    x <- cbind(age, BMI, children, charges)

    cor(x)
```

```
##                 age       BMI   children    charges
## age      1.0000000 0.1092719 0.04246900 0.29900819
## BMI      0.1092719 1.0000000 0.01275890 0.19834097
## children 0.0424690 0.0127589 1.00000000 0.06799823
## charges  0.2990082 0.1983410 0.06799823 1.00000000
```

```r
    scatterplotMatrix(x, spread = FALSE, col = "blue", main = "ScatterPlot Matrix")
```




```
detach(insurance)
```

```
<!-- Interpretation- The scatter plot matrix shows a clear correlation between age-BMI, age-char
ges and BMI-charges.  -->


<!-- The values in correlation  are indicative of the same. -->


<!-- ## Building Regression model -->
```

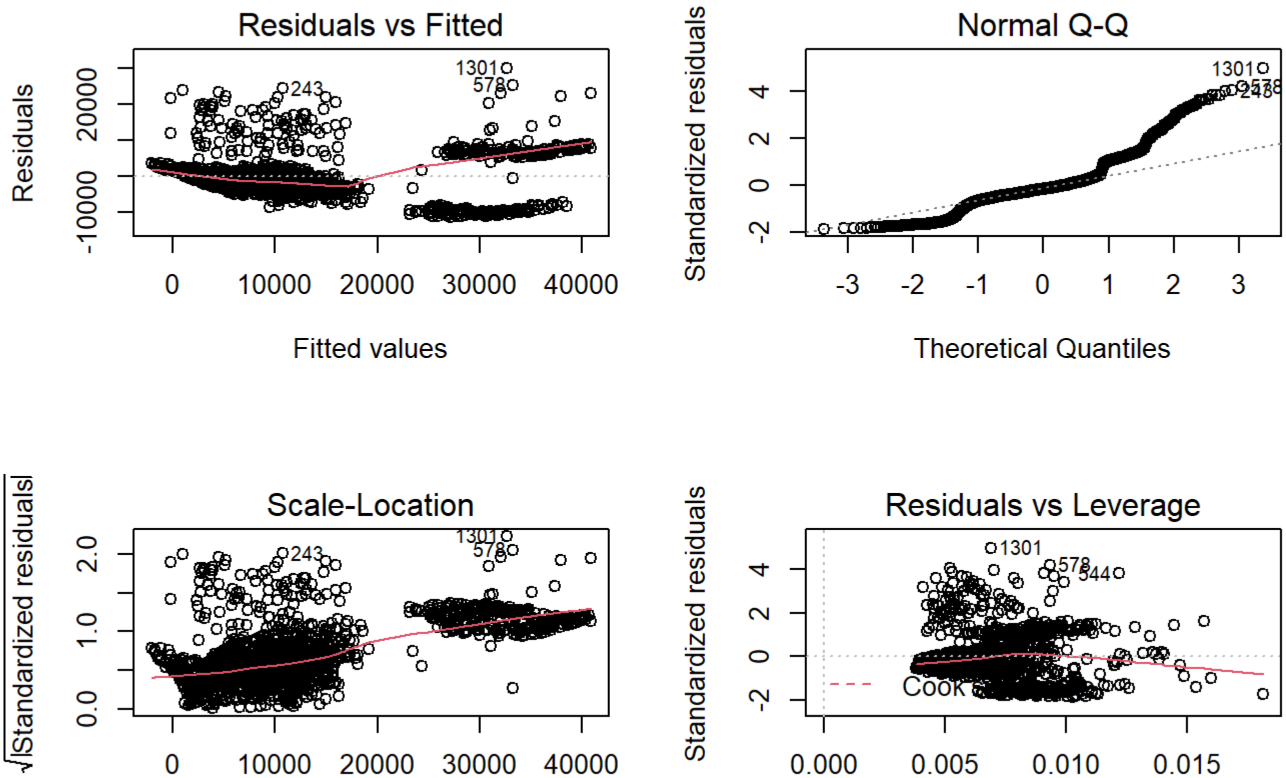
```
fit1 <- lm(charges ~ ., data = insurance)

summary(fit1)
```

```
##
## Call:
## lm(formula = charges ~ ., data = insurance)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11304.9  -2848.1   -982.1   1393.9  29992.8
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -11938.5      987.8 -12.086  < 2e-16 ***
## age                256.9       11.9  21.587  < 2e-16 ***
## sexmale           -131.3      332.9  -0.394 0.693348
## BMI                339.2       28.6  11.860  < 2e-16 ***
## children           475.5      137.8   3.451 0.000577 ***
## smokeryes        23848.5      413.1  57.723  < 2e-16 ***
## regionnorthwest   -353.0      476.3  -0.741 0.458769
## regionsoutheast  -1035.0      478.7  -2.162 0.030782 *
## regionsouthwest   -960.0      477.9  -2.009 0.044765 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6062 on 1329 degrees of freedom
## Multiple R-squared:  0.7509, Adjusted R-squared:  0.7494
## F-statistic: 500.8 on 8 and 1329 DF,  p-value: < 2.2e-16
```

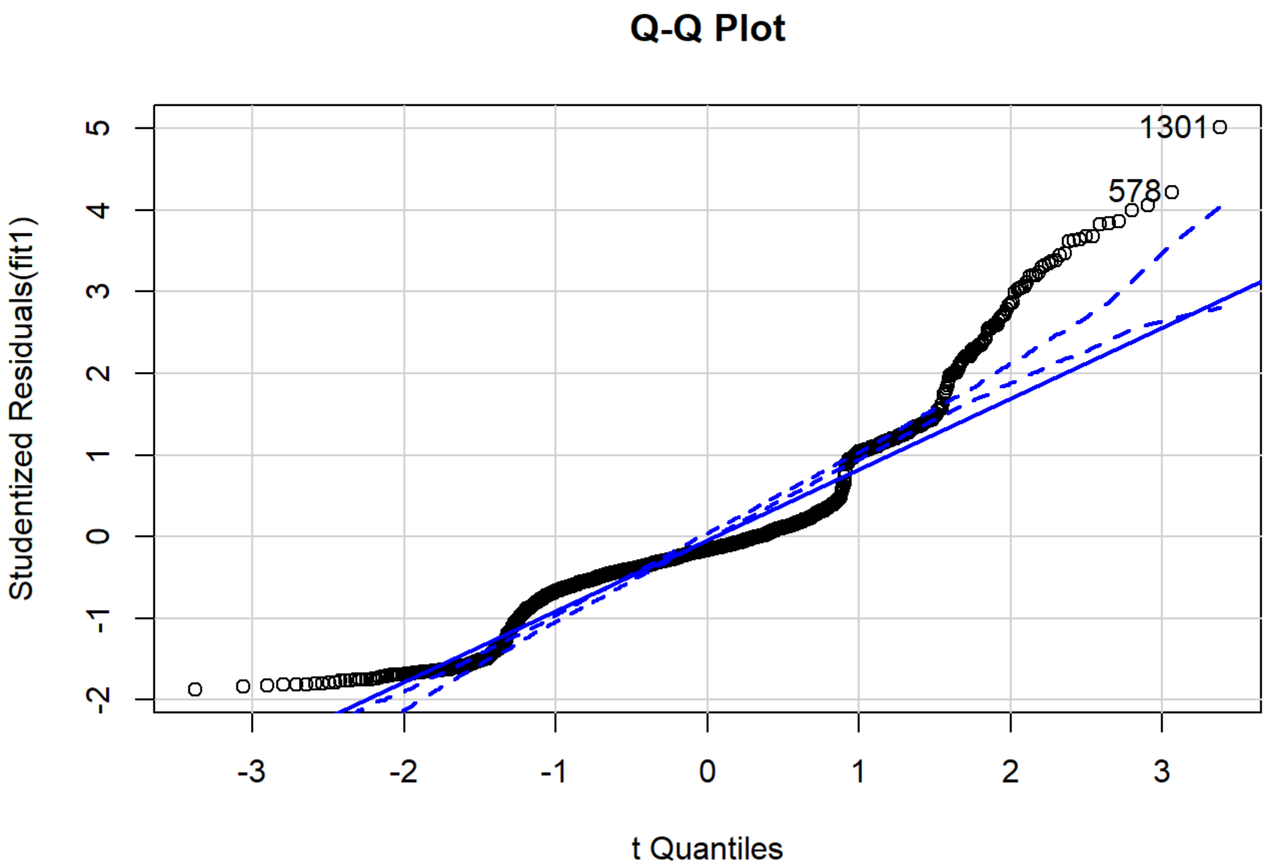
```
6062 / mean(insurance$charges)
```

```
## [1] 0.4568054
```

```
par(mfrow = c(2, 2))
plot(fit1)
```



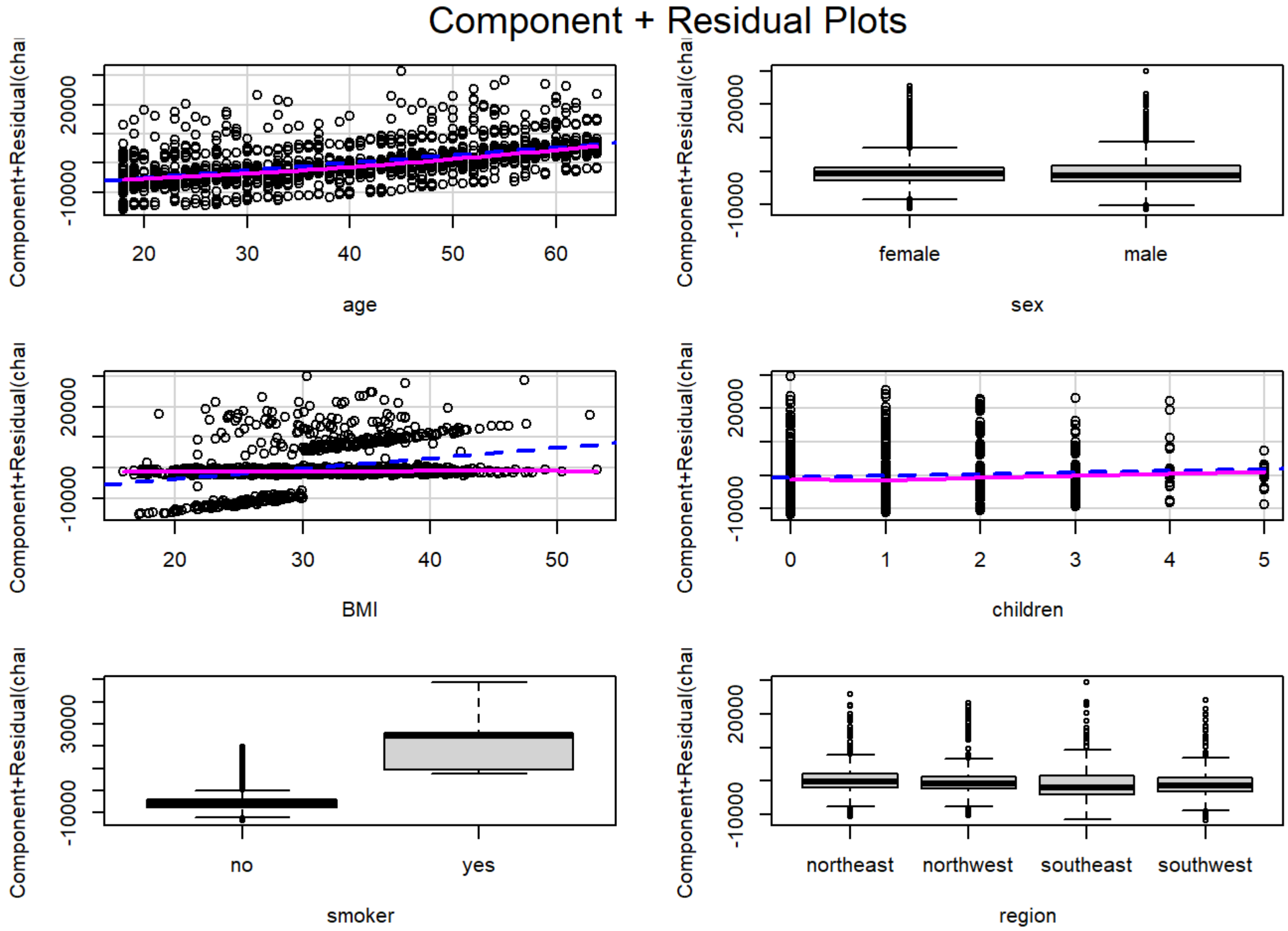

```
qqPlot(fit1, labels = row.names(insurance), id.method = "identify", simulate = TRUE, main =
"Q-Q Plot")
```




```
## [1]  578 1301
```

```
<!-- most of the points dont fall close to the line and inside the confidence interval suggestin
g that -->
<!-- the normality assumption has not been met -->


<!--  Linearity -->
```

```
    crPlots(fit1)
```

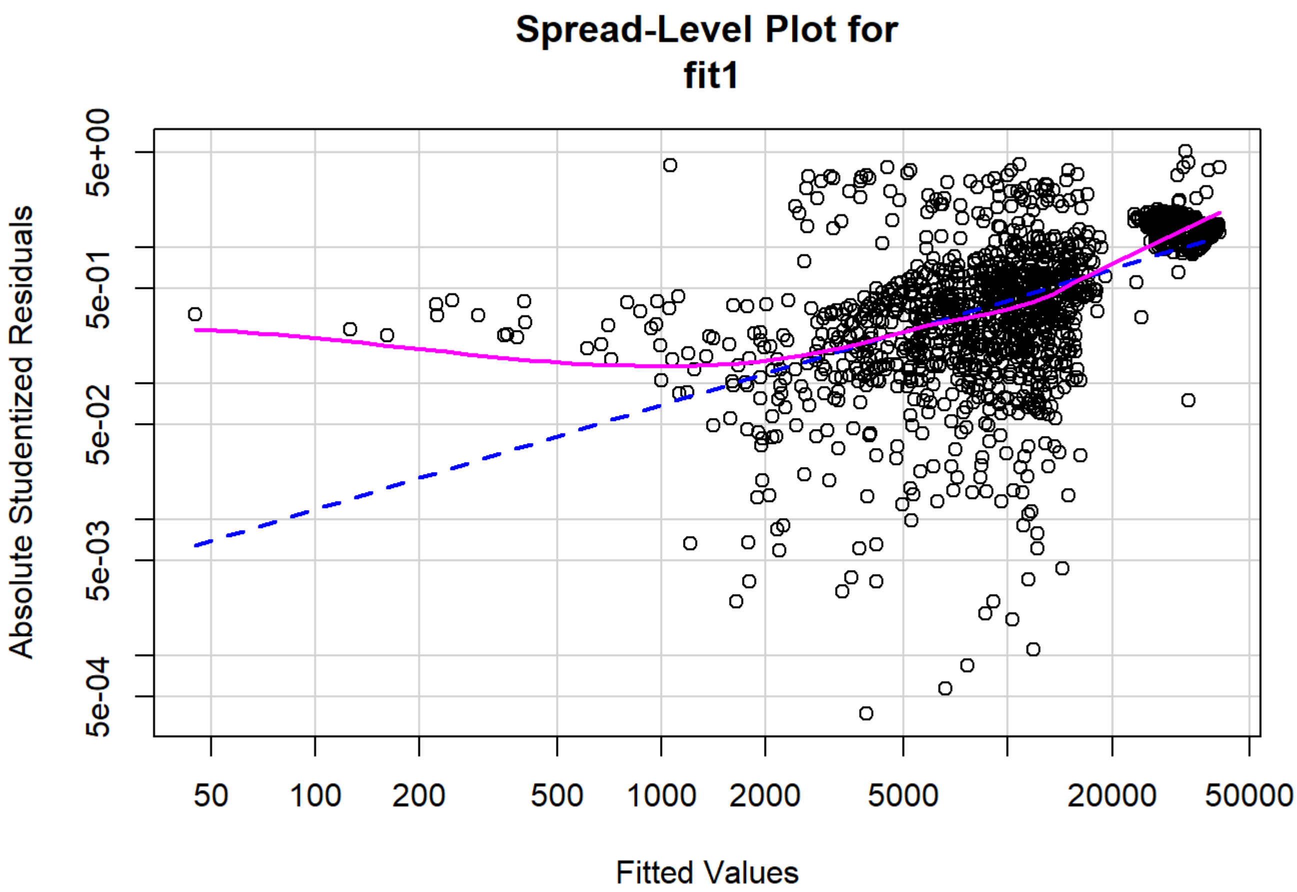



```
ncvTest(fit1)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 236.1255, Df = 1, p = < 2.22e-16
```

```
spreadLevelPlot(fit1)
```

```
## Warning in spreadLevelPlot.lm(fit1):
## 20 negative fitted values removed
```




```
##
## Suggested power transformation:  0.2331668
```

```
ins$BMI <- findInterval(ins$BMI, c(0, 30))

ins$BMI <- as.factor(ins$BMI)

levels(ins$BMI) <- c(0, 1)


fit2 <- lm(charges ~ age + I(age^2) + sex + BMI + children + smoker + region, data = ins)

summary(fit2)
```

```
##
## Call:
## lm(formula = charges ~ age + I(age^2) + sex + BMI + children +
##     smoker + region, data = ins)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -13593  -3406    452   1066  28347
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       650.483   1483.996   0.438 0.661217
## age               -17.774     80.687  -0.220 0.825689
## I(age^2)            3.505      1.007   3.481 0.000516 ***
## sexmale          -149.720    329.532  -0.454 0.649658
## BMI1             4173.898    336.771  12.394  < 2e-16 ***
## children          630.934    142.910   4.415 1.09e-05 ***
## smokeryes       23844.170    408.889  58.314  < 2e-16 ***
## regionnorthwest  -416.894    471.407  -0.884 0.376661
## regionsoutheast  -570.432    464.921  -1.227 0.220061
## regionsouthwest  -861.471    472.307  -1.824 0.068382 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6000 on 1328 degrees of freedom
## Multiple R-squared:  0.7562, Adjusted R-squared:  0.7545
## F-statistic: 457.7 on 9 and 1328 DF,  p-value: < 2.2e-16
```

```
rm(list = ls())
```

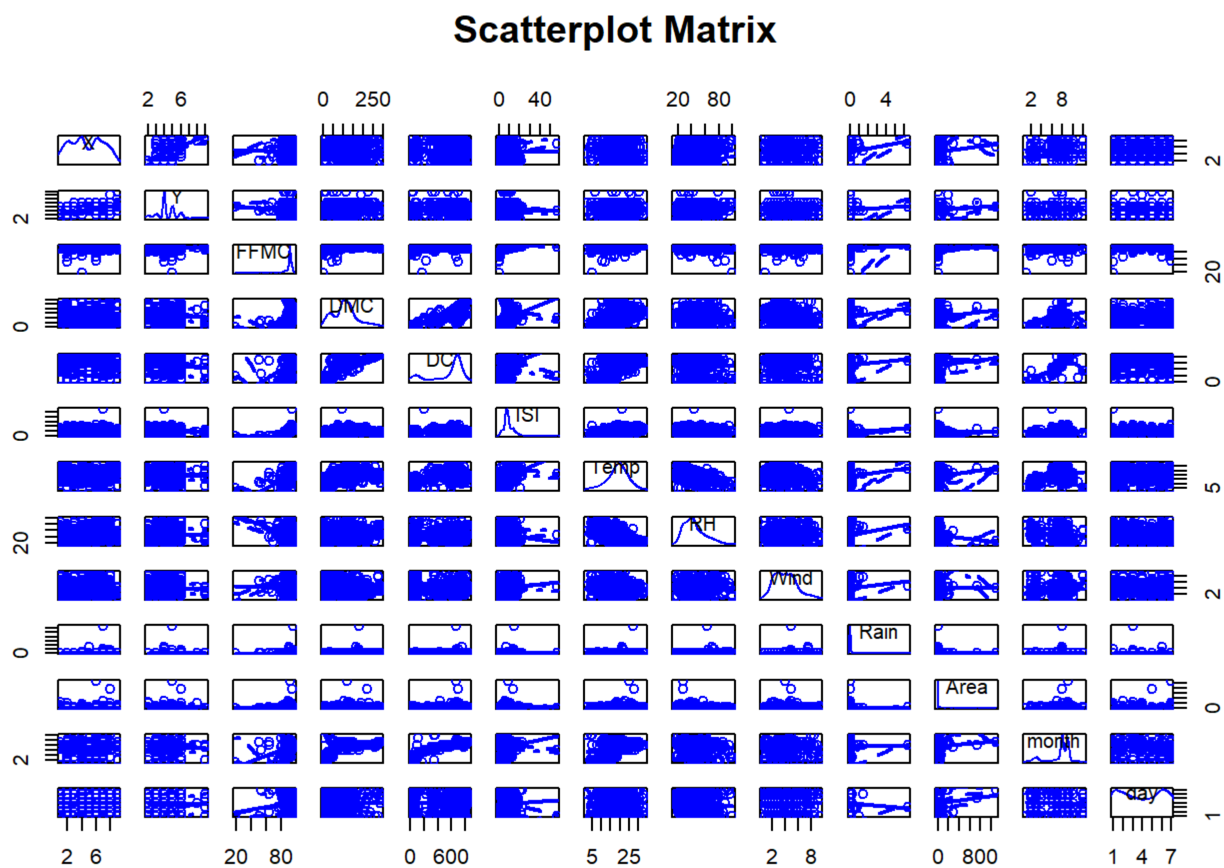
```
<!-- Problem 4 - . Multiple Linear Regression Model for Forest Fire Data -->


<!-- Importing the dataset -->
```

```
Forest_Fires <- read_excel("Forest Fires Data.xlsx")
```

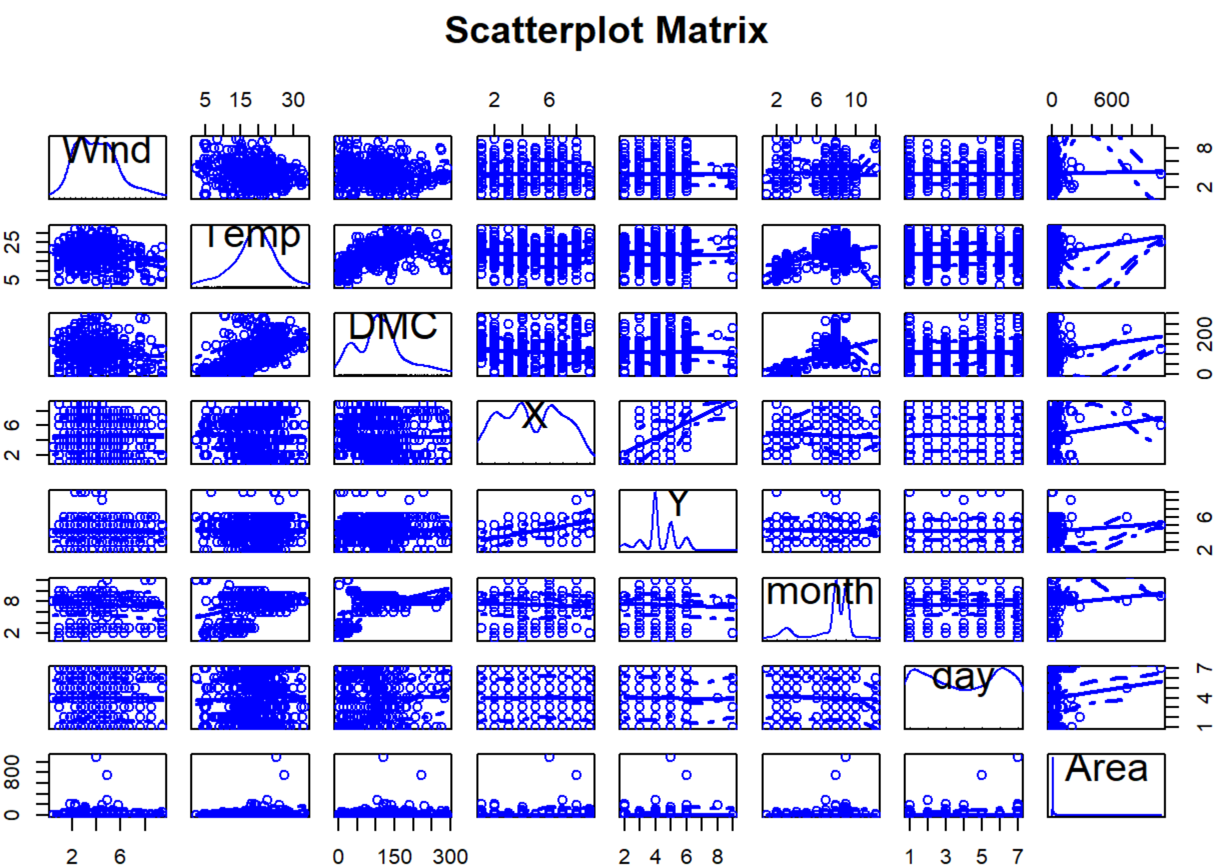
```
Forest_Fires$month <- dplyr::recode(Forest_Fires$Month,
      "jan" = 1, "feb" = 2, "mar" = 3,
      "apr" = 4, "may" = 5, "jun" = 6, "jul" = 7, "aug" = 8, "sep" = 9,
      "oct" = 10, "nov" = 11, "dec" = 12)

Forest_Fires$day <- dplyr::recode(Forest_Fires$Day,
      "sun" = 1, "mon" = 2, "tue" = 3,
      "wed" = 4, "thu" = 5, "fri" = 6, "sat" = 7)


Forest_Fires <- Forest_Fires[, -c(3, 4)]
```

```
<!-- Plot the scatterplot matrix -->
```

```r
scatterplotMatrix(Forest_Fires, main = "Scatterplot Matrix")
```




```r
scatterplotMatrix(~ Wind + Temp + DMC + X + Y + month + day + `Area`,
  data = Forest_Fires,
  main = "Scatterplot Matrix"
)
```




```
mod1 <- lm(Area ~ Wind + Temp + DMC + X + Y + month + day, data = Forest_Fires)

summary(mod1)
```

```
##
## Call:
## lm(formula = Area ~ Wind + Temp + DMC + X + Y + month + day,
##     data = Forest_Fires)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
##  -36.18  -15.54   -8.90   -0.21 1064.00
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -34.76405   18.74342  -1.855   0.0642 .
## Wind          1.28947    1.60279   0.805   0.4215
## Temp          0.95851    0.56714   1.690   0.0916 .
## DMC           0.02709    0.05307   0.510   0.6100
## X             1.65288    1.43855   1.149   0.2511
## Y             0.88505    2.70931   0.327   0.7441
## month         0.59834    1.42140   0.421   0.6740
## day           1.34050    1.30759   1.025   0.3058
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 63.48 on 509 degrees of freedom
## Multiple R-squared:  0.01896,    Adjusted R-squared:  0.005467
## F-statistic: 1.405 on 7 and 509 DF,  p-value: 0.2008
```

```
mod2 <- lm(Area ~ Wind + Temp + X + day, data = Forest_Fires)

summary(mod2)
```

```
##
## Call:
## lm(formula = Area ~ Wind + Temp + X + day, data = Forest_Fires)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
##  -33.51  -15.46   -9.23   -0.76 1064.22
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -28.523     14.901  -1.914   0.0562 .
## Wind           1.270      1.598   0.795   0.4271
## Temp           1.184      0.494   2.397   0.0169 *
## X              1.862      1.207   1.543   0.1236
## day            1.311      1.302   1.007   0.3144
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 63.34 on 512 degrees of freedom
## Multiple R-squared:  0.01743,    Adjusted R-squared:  0.009756
## F-statistic: 2.271 on 4 and 512 DF,  p-value: 0.06056
```

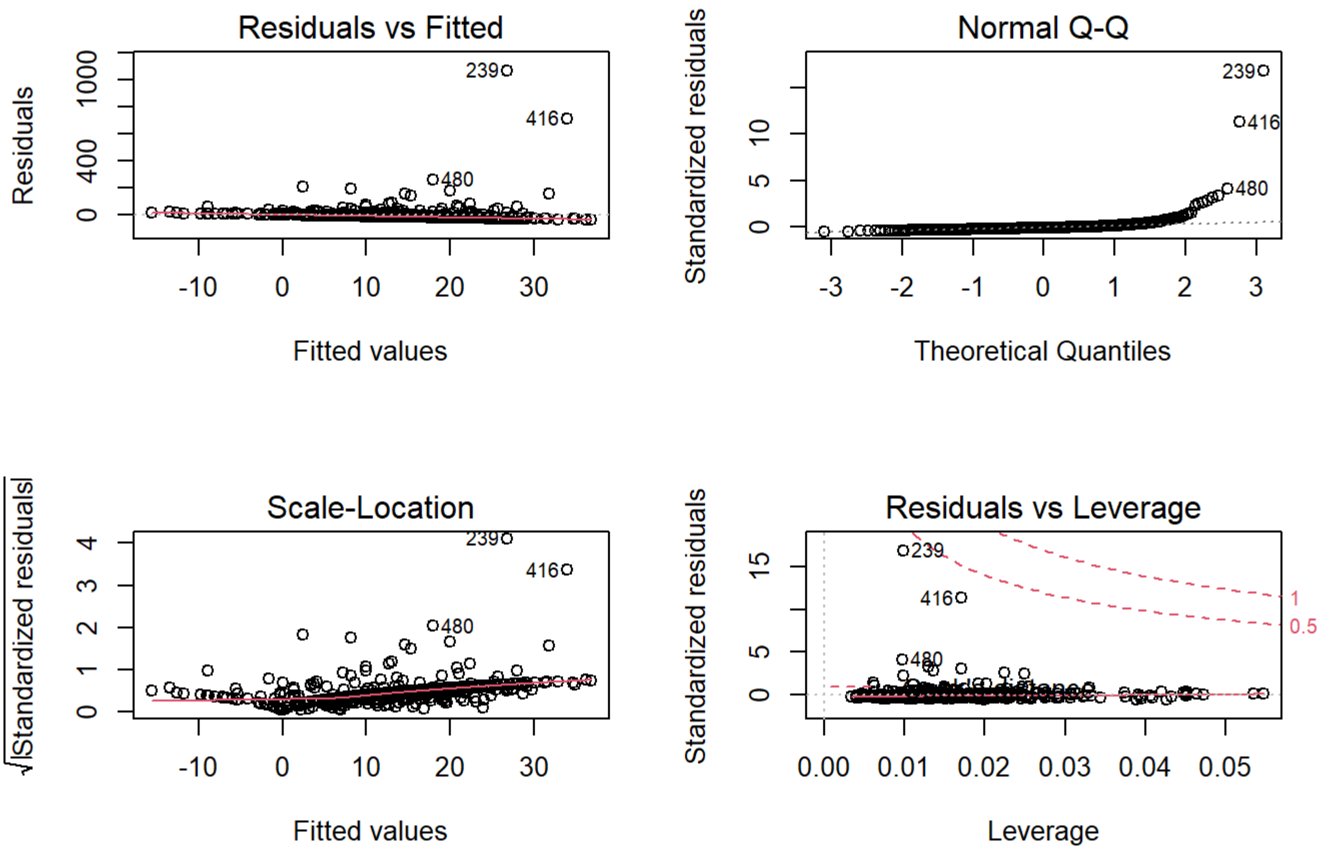
```
mod3 <- lm(Area ~ Wind + I(Temp^2) + X + day, data = Forest_Fires)

summary(mod3)
```

```
##
## Call:
## lm(formula = Area ~ Wind + I(Temp^2) + X + day, data = Forest_Fires)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
##  -37.80  -15.28   -8.86   -0.59 1063.32
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -19.31930   12.14241  -1.591   0.1122
## Wind          1.21374    1.58797   0.764   0.4450
## I(Temp^2)     0.03391    0.01333   2.544   0.0112 *
## X             1.85533    1.20584   1.539   0.1245
## day           1.35564    1.30013   1.043   0.2976
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 63.3 on 512 degrees of freedom
## Multiple R-squared:  0.01881,    Adjusted R-squared:  0.01114
## F-statistic: 2.454 on 4 and 512 DF,  p-value: 0.04502
```

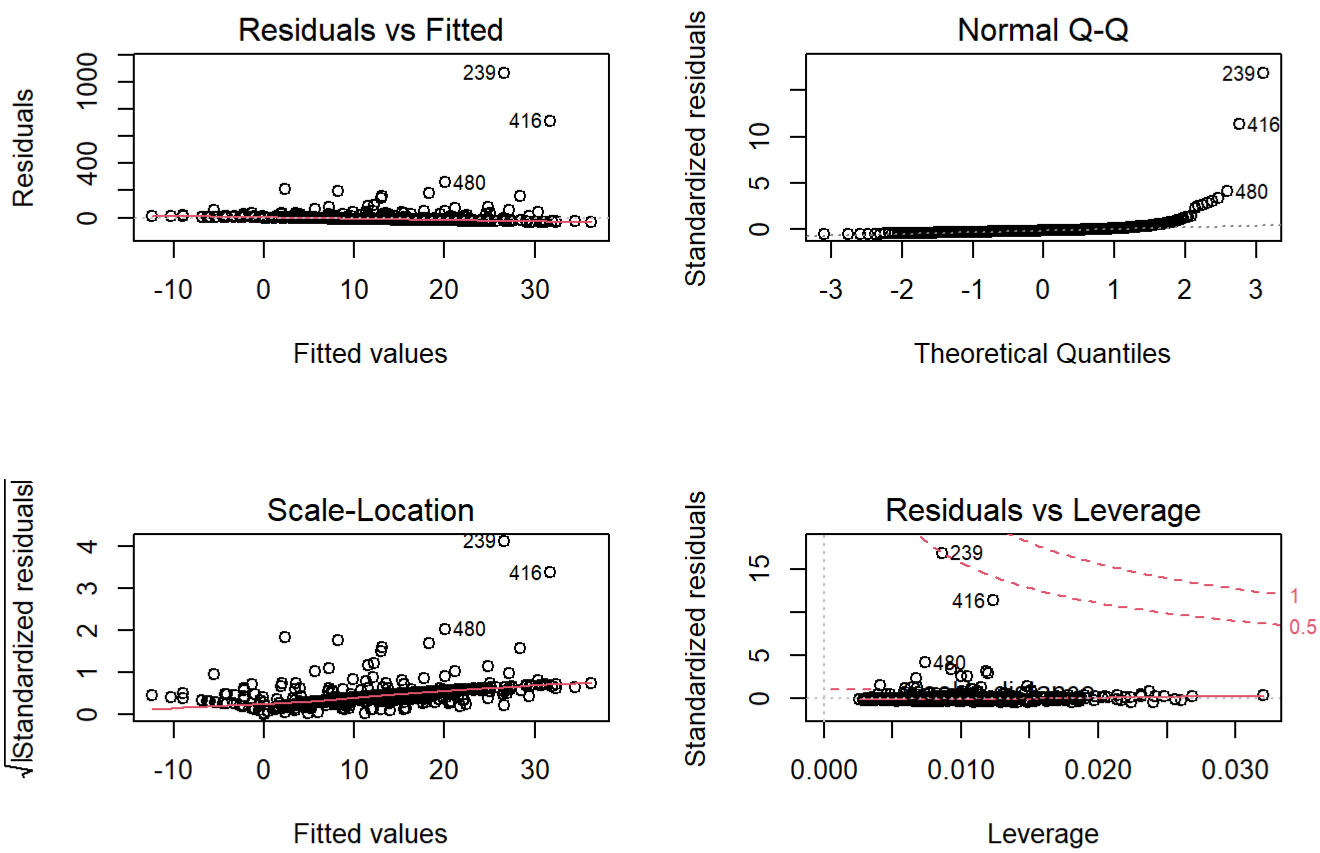
```
mod4 <- lm(Area ~ log(Wind) + I(Temp^2) + X + day, data = Forest_Fires)

summary(mod4)
```

```
##
## Call:
## lm(formula = Area ~ log(Wind) + I(Temp^2) + X + day, data = Forest_Fires)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
##  -38.02  -15.23   -8.80   -0.03 1062.92
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -20.66331   12.33956  -1.675   0.0946 .
## log(Wind)     5.10875    5.63346   0.907   0.3649
## I(Temp^2)     0.03347    0.01317   2.540   0.0114 *
## X             1.83387    1.20597   1.521   0.1290
## day           1.34536    1.29988   1.035   0.3012
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 63.29 on 512 degrees of freedom
## Multiple R-squared:  0.01927,    Adjusted R-squared:  0.0116
## F-statistic: 2.514 on 4 and 512 DF,  p-value: 0.04077
```

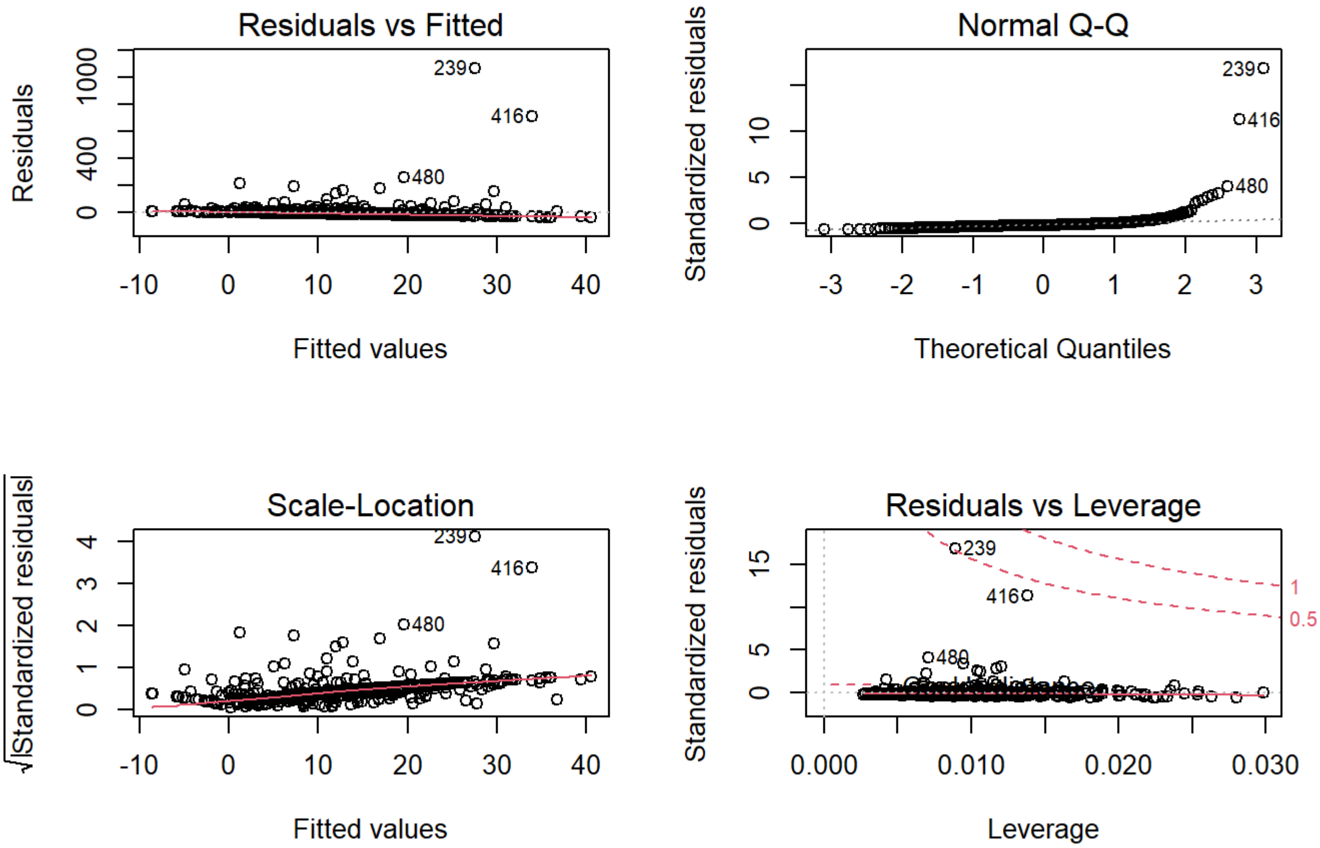
```
# Model 1
par(mfrow = c(2, 2))
plot(mod1)
```



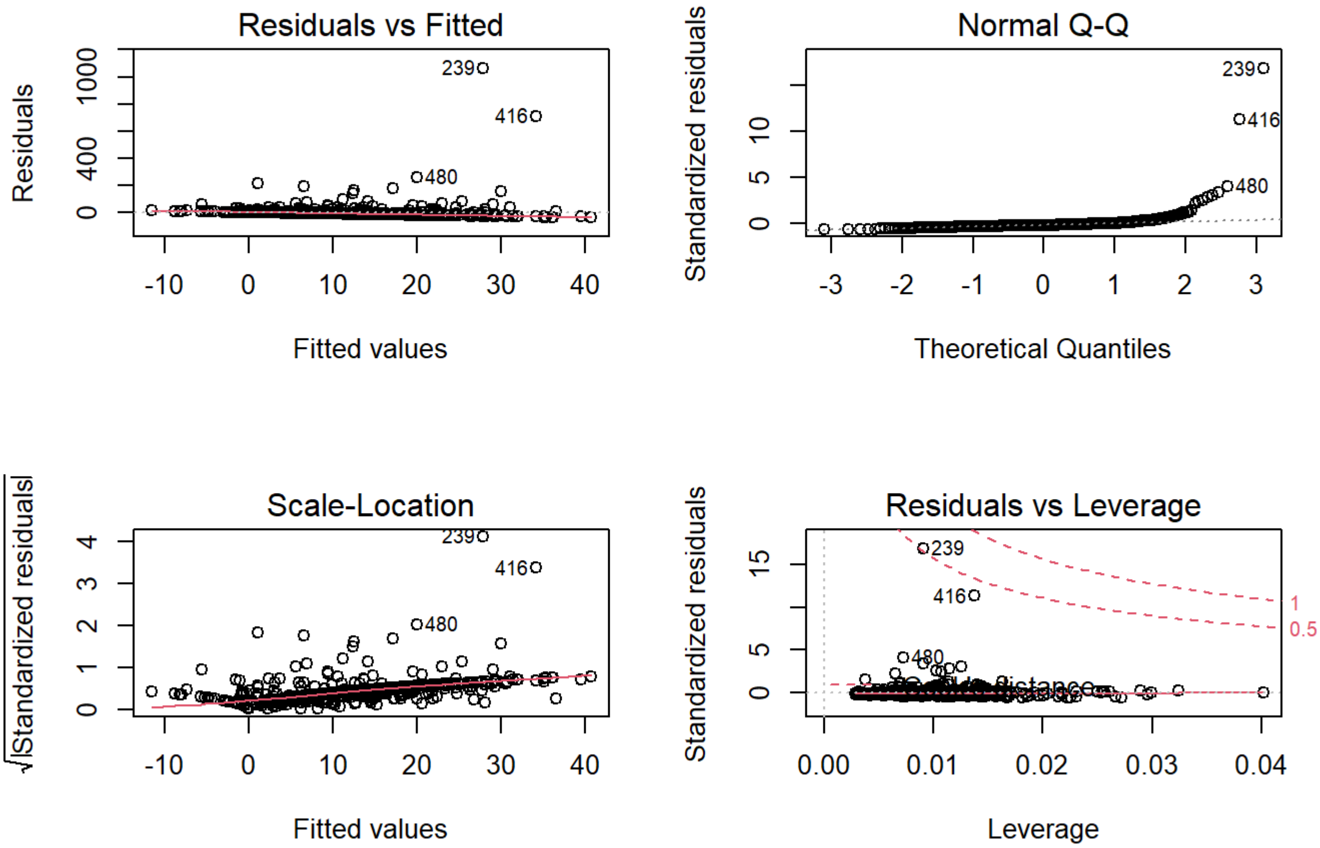

```
# Model 2
par(mfrow = c(2, 2))
plot(mod2)
```



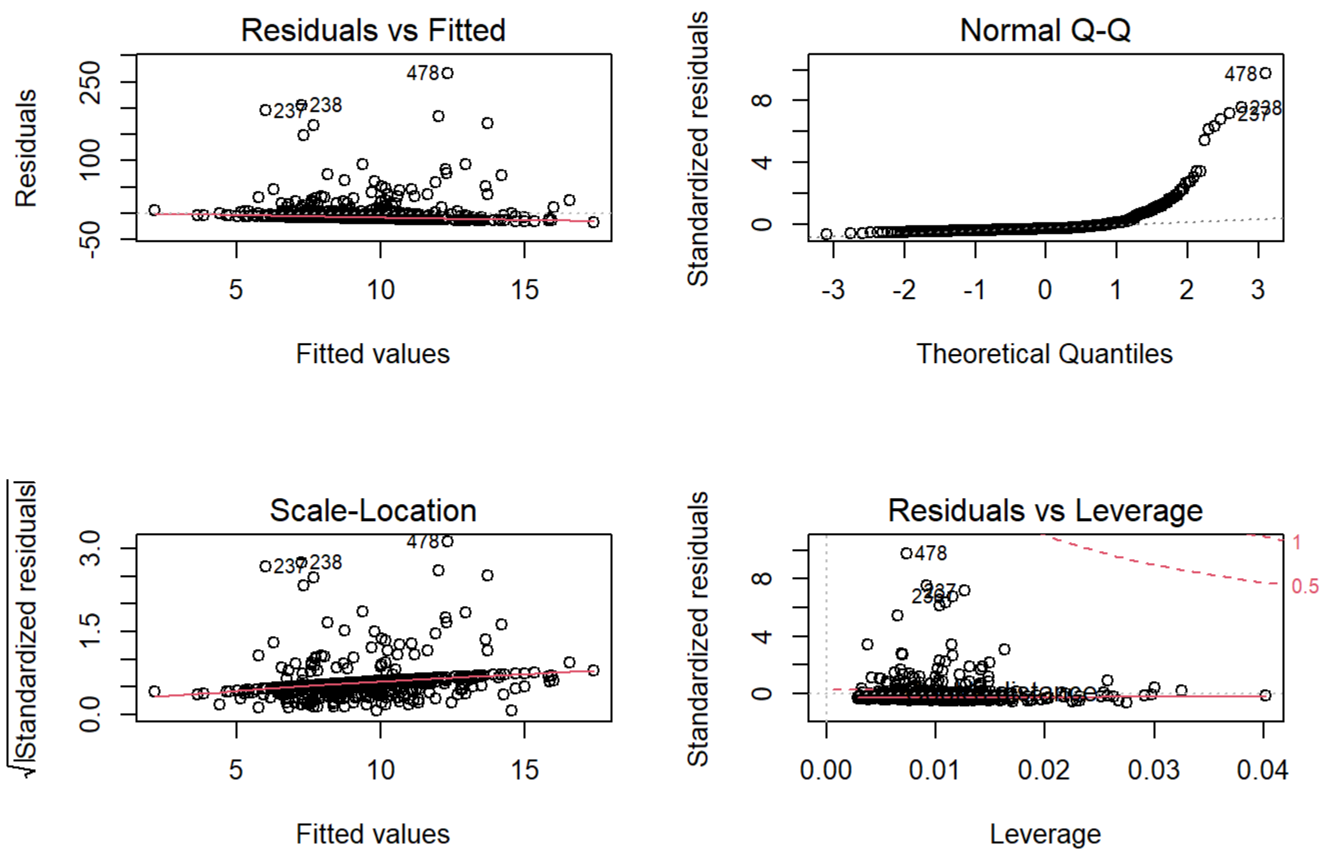

```
# Model 3
par(mfrow = c(2, 2))
plot(mod3)
```



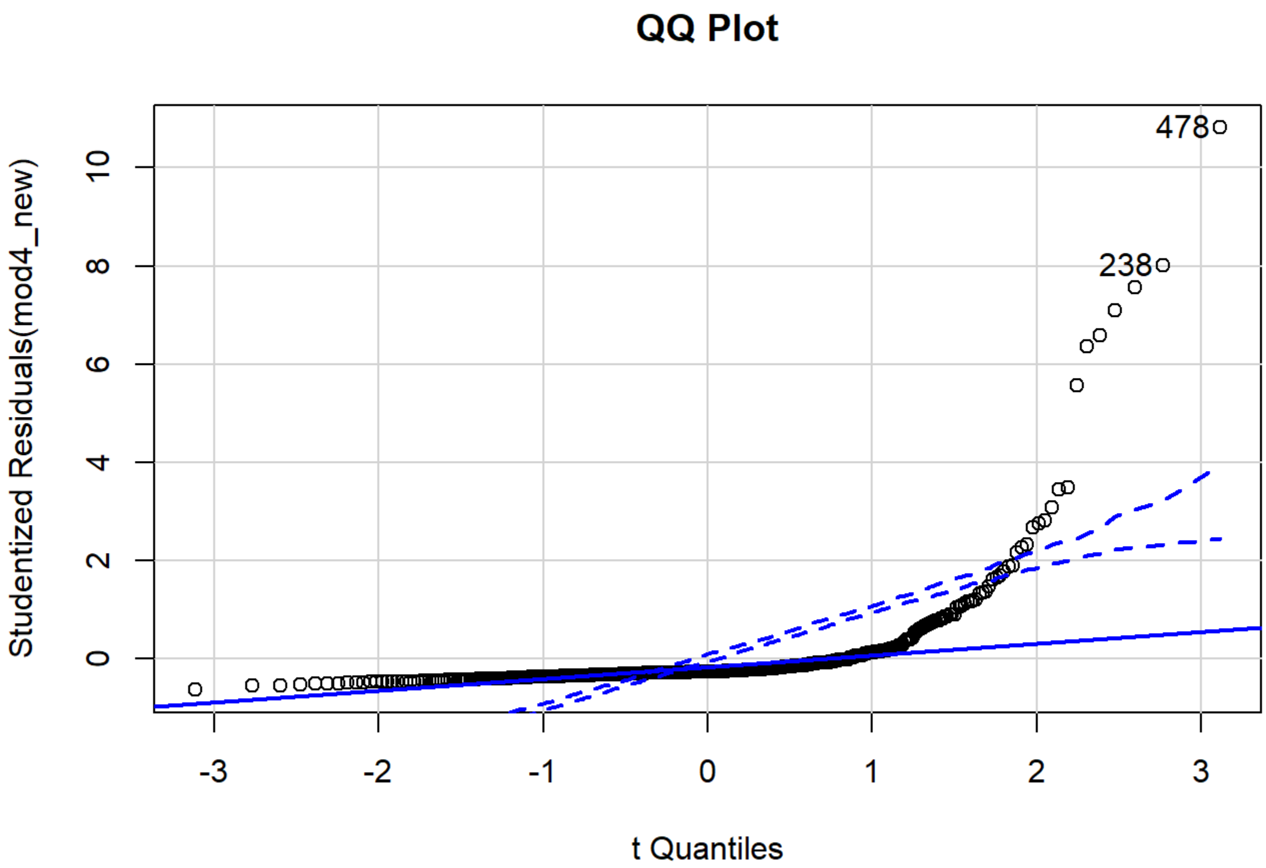

```
# Model 4
par(mfrow = c(2, 2))
plot(mod4)
```




```
mod4_new <- lm(Area ~ log(Wind) + I(Temp^2) + X + day, data = Forest_Fires[-c(416, 239), ])

# new Model 4
par(mfrow = c(2, 2))
plot(mod4_new)
```

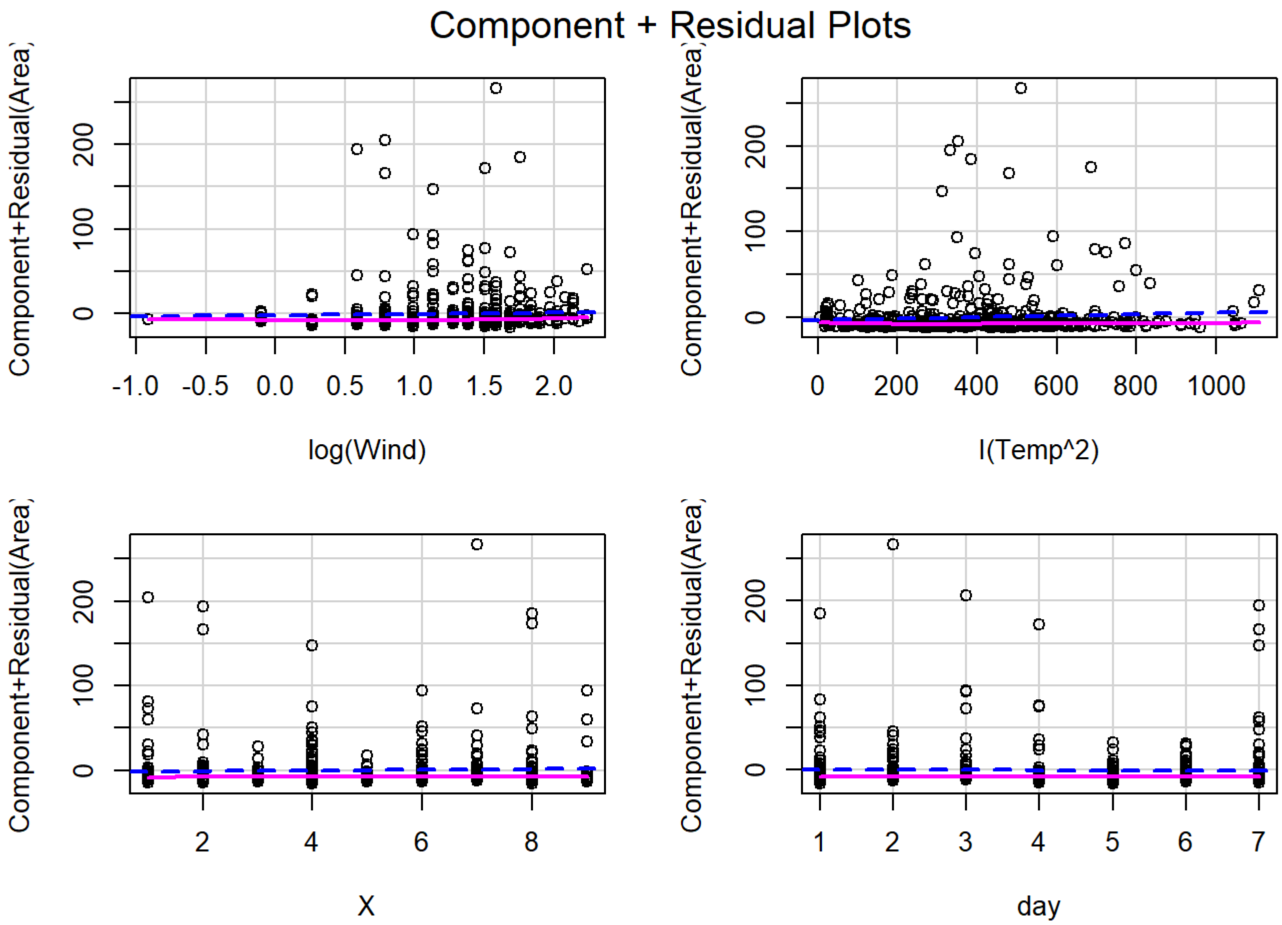



```
qqPlot(mod4_new, labels = rownames(Forest_Fires), id.method = "identify", simulate = TRUE, main = "QQ Plot")
```




```
## [1] 238 478
```

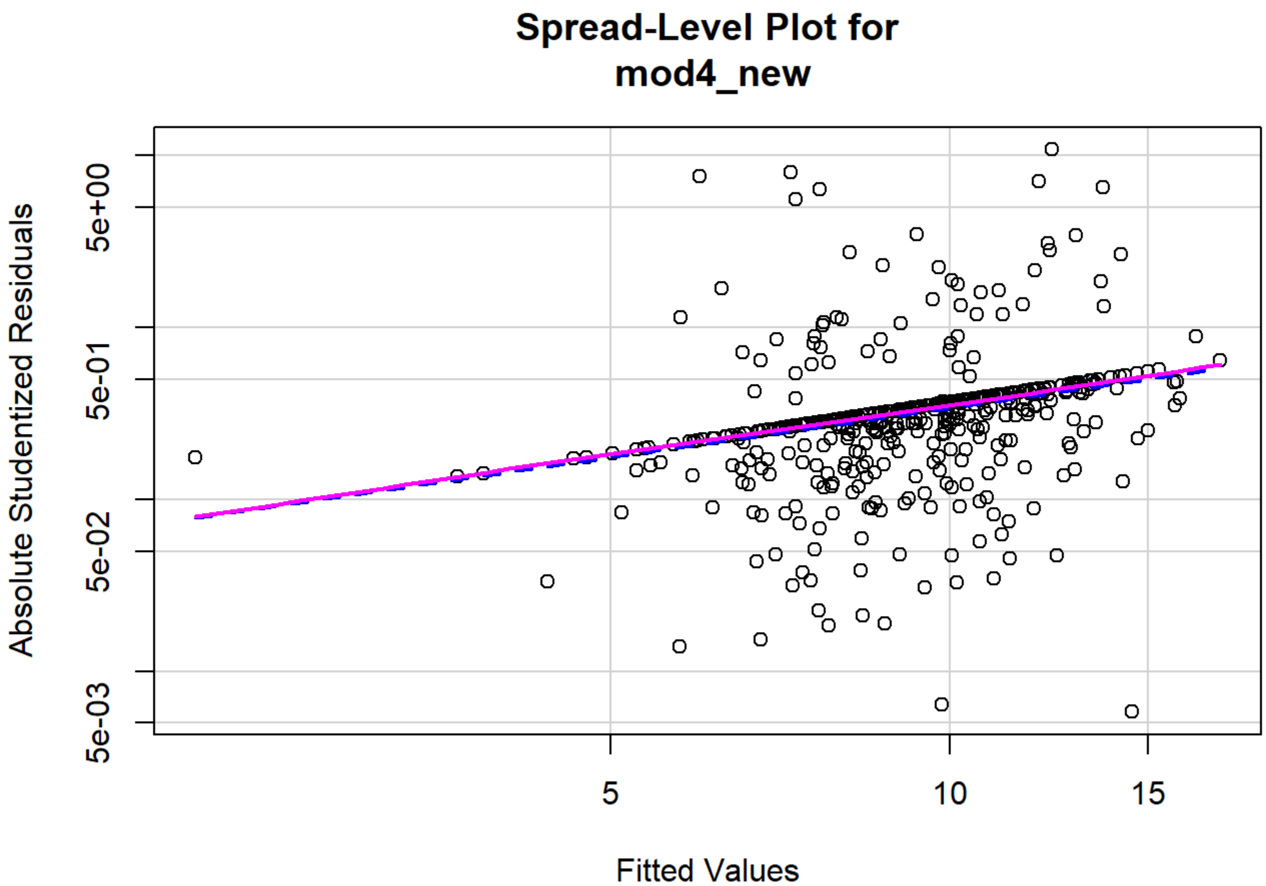
```
crPlots(mod4_new)
```




```
ncvTest(mod4_new)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 25.40227, Df = 1, p = 4.6537e-07
```

```
spreadLevelPlot(mod4_new)
```

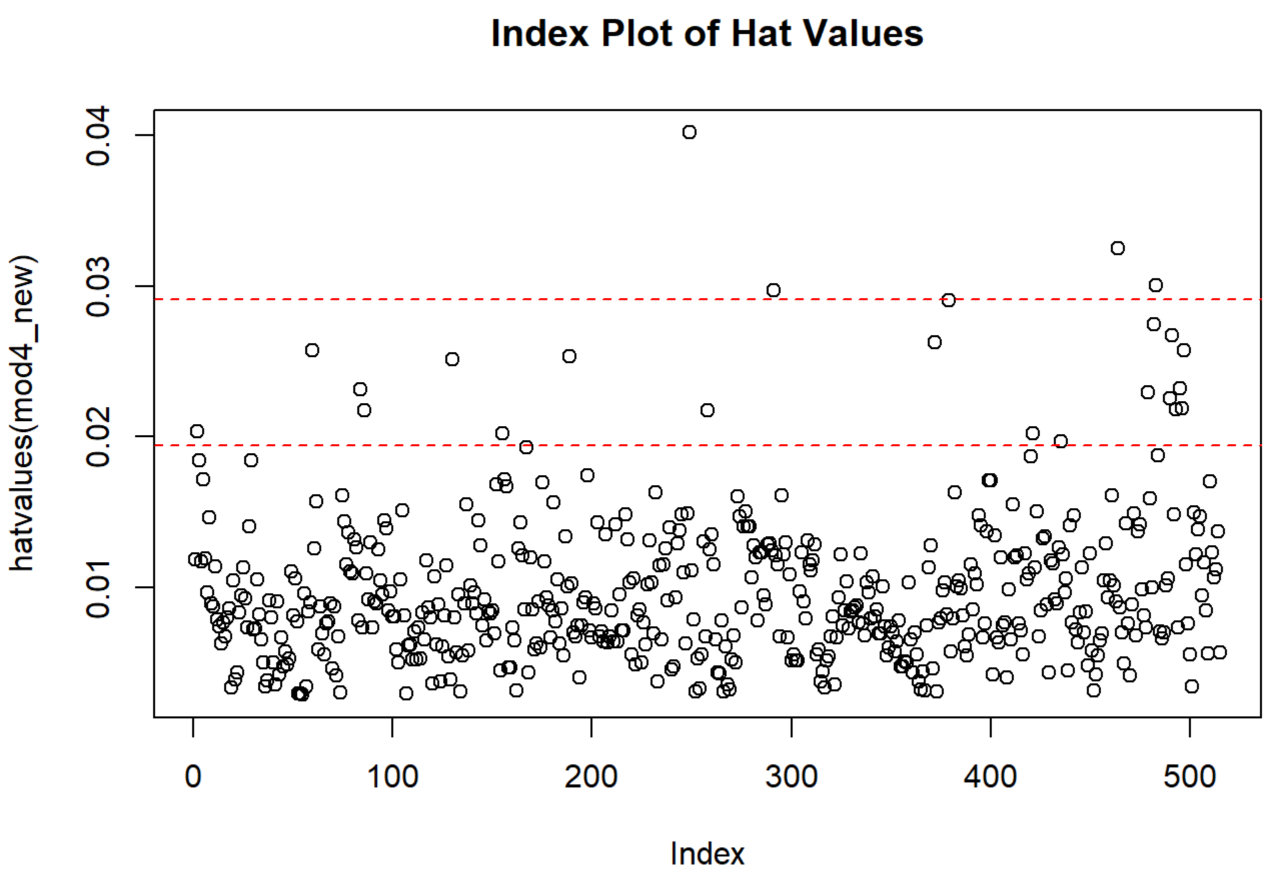



```
##
## Suggested power transformation:  0.04966621
```

```
outlierTest(mod4_new)
```

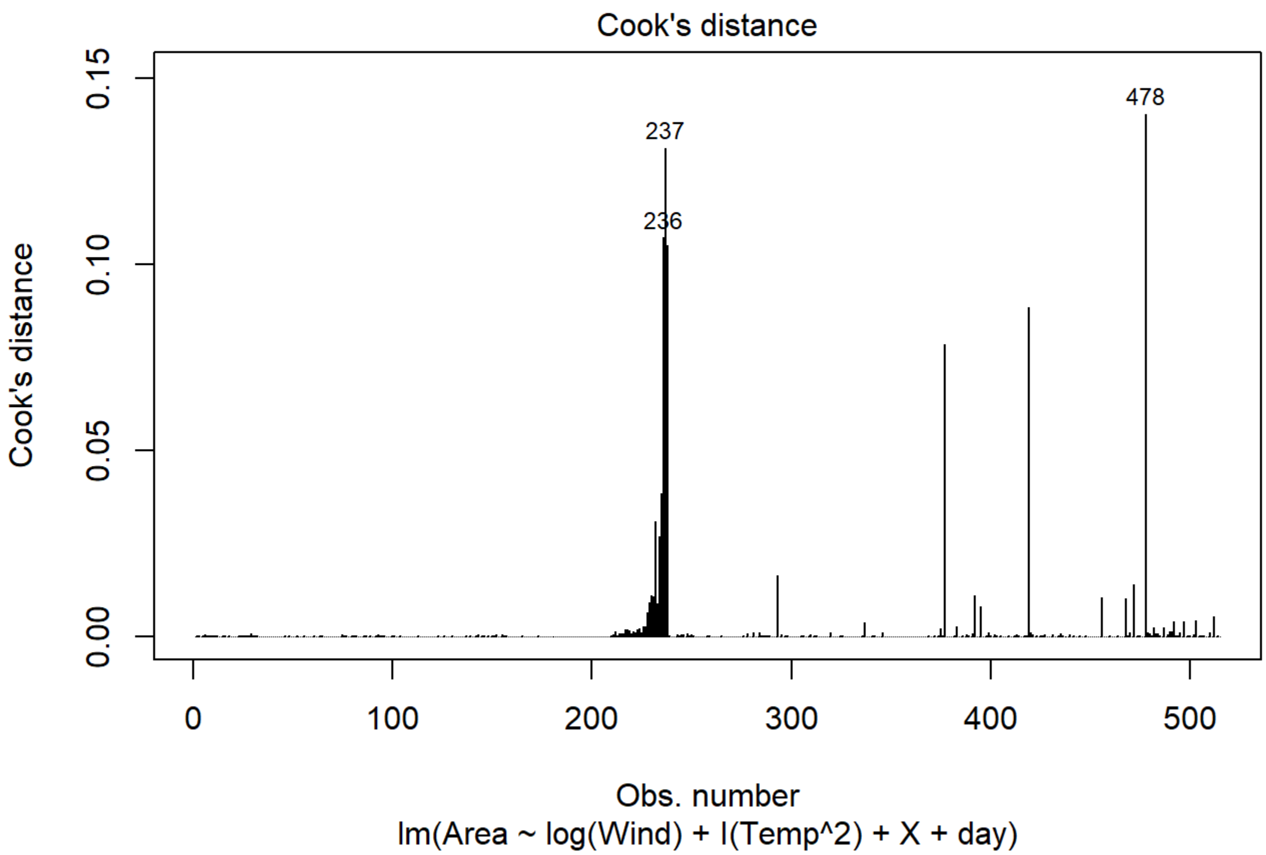
```
##      rstudent unadjusted p-value Bonferroni p
## 478 10.817526         1.0892e-24   5.6092e-22
## 238  8.002656         8.2955e-15   4.2722e-12
## 237  7.552753         1.9893e-13   1.0245e-10
## 236  7.101350         4.1807e-12   2.1531e-09
## 419  6.579747         1.1730e-10   6.0407e-08
## 377  6.366131         4.3325e-10   2.2312e-07
## 235  5.567427         4.1894e-08   2.1576e-05
```

```
hat.plot <- function(mod4_new) {
  p <- length(coefficients(mod4_new))
  n <- length(fitted(mod4_new))
  plot(hatvalues(mod4_new),
    main = "Index Plot of Hat Values"
  )
  abline(h = c(2, 3) * p / n, col = "red", lty = 2)
  identify(1:n, hatvalues(mod4_new), names(hatvalues(mod4_new)))
}
hat.plot(mod4_new)
```

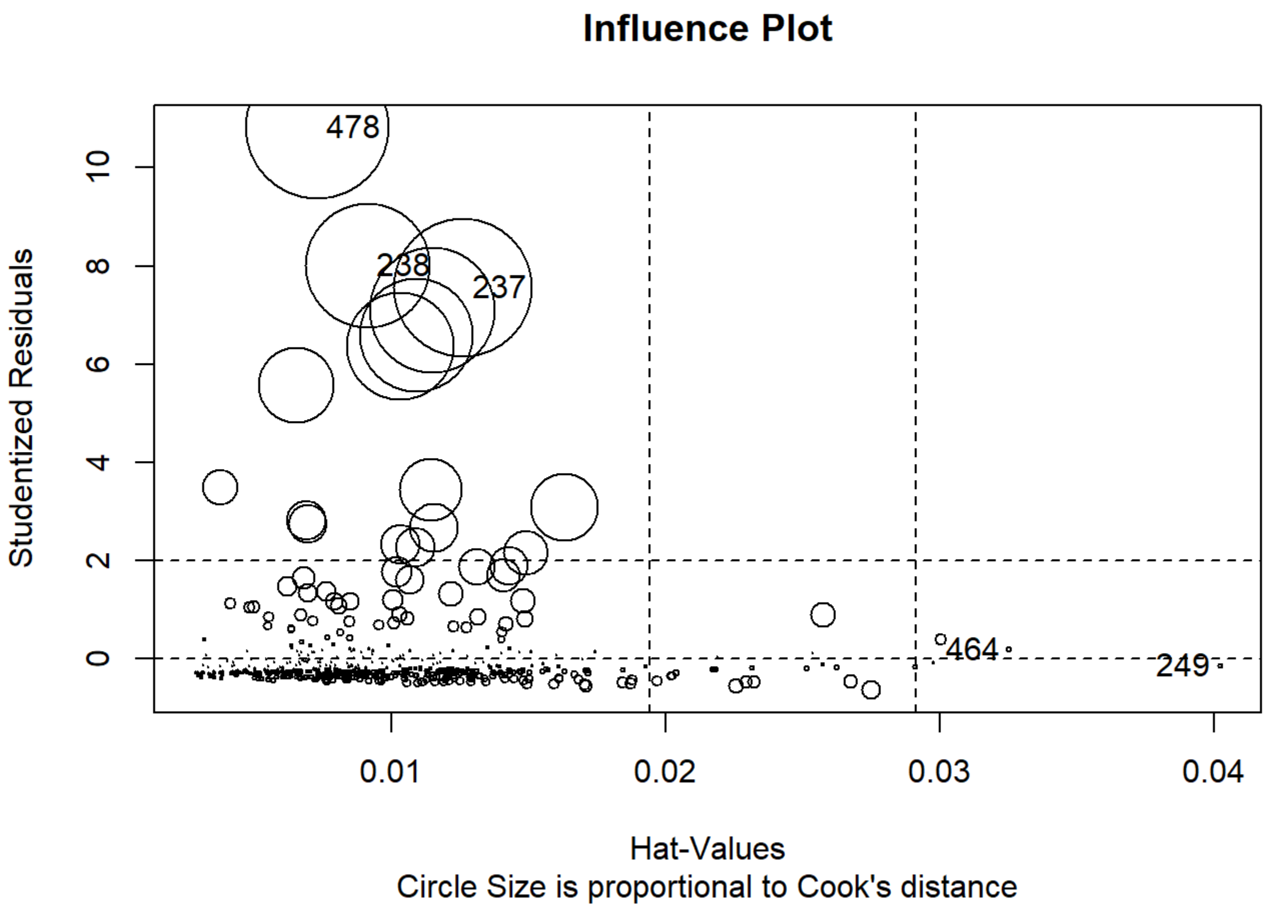



```
## integer(0)
```

```
cutoff <- 4 / (nrow(df) - length(mod4_new$coefficients) - 2)
plot(mod4_new, which = 4, cook.levels = cutoff)
abline(h = cutoff, lty = 2, col = "red")
```




```
    influencePlot(mod4_new,
      main = "Influence Plot",
      sub = "Circle Size is proportional to Cook's distance"
)
```




```
##        StudRes         Hat        CookD
## 237  7.5527527 0.012609015 0.1312660582
## 238  8.0026561 0.009132413 0.1050634255
## 249 -0.1517836 0.040197655 0.0001933443
## 464  0.1744308 0.032526536 0.0002049752
## 478 10.8175256 0.007311625 0.1404331420
```

```
mod1 <- lm(Area ~ Wind + Temp + DMC + X + Y + month + day, data = Forest_Fires[-c(237, 238, 478), ])


mod2 <- lm(Area ~ Wind + Temp + X + day, data = Forest_Fires[-c(237, 238, 478), ])


mod3 <- lm(Area ~ Wind + I(Temp^2) + X + day, data = Forest_Fires[-c(237, 238, 478), ])


mod4 <- lm(Area ~ log(Wind) + I(Temp^2) + X + day, data = Forest_Fires[-c(237, 238, 478), ])


mod5 <- lm(Area ~ I(Temp^2) + I(X^1.3) + I(day^10), data = Forest_Fires[-c(237, 238, 478), ])
```

```
anova(mod2, mod1)
```

```
## Analysis of Variance Table
##
## Model 1: Area ~ Wind + Temp + X + day
## Model 2: Area ~ Wind + Temp + DMC + X + Y + month + day
##   Res.Df     RSS Df Sum of Sq      F Pr(>F)
## 1    509 1971428
## 2    506 1967546  3    3881.6 0.3327 0.8017
```

```
AIC(mod1, mod2, mod3, mod4)
```

```
##      df      AIC
## mod1  9 5717.207
## mod2  6 5712.220
## mod3  6 5711.240
## mod4  6 5710.882
```

```
<!-- As only model 1 and 2 are nested models the anova funcion is used on them. -->
<!-- The insignificant P value indicated that the excess predicitors in  the pairs  -->
<!-- dont add to the linear predictions so we are better off dropping them -->
<!-- i.e model 2 is the best among the pair -->


<!-- The AIC test indicated that model 4 is the best among them all. -->


<!-- Let's interpret the results -->
```

```
summary(mod4)
```

```
##
## Call:
## lm(formula = Area ~ log(Wind) + I(Temp^2) + X + day, data = Forest_Fires[-c(237,
##     238, 478), ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
##  -40.49  -14.70   -8.00    0.99 1063.00
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -26.08092   12.17605  -2.142  0.03267 *
## log(Wind)     7.04371    5.54970   1.269  0.20495
## I(Temp^2)     0.03532    0.01297   2.723  0.00669 **
## X             2.30496    1.18883   1.939  0.05307 .
## day           1.15423    1.28195   0.900  0.36835
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 62.15 on 509 degrees of freedom
## Multiple R-squared:  0.02424,    Adjusted R-squared:  0.01658
## F-statistic: 3.162 on 4 and 509 DF,  p-value: 0.0139
```

```
prediction <- predict(mod4, Forest_Fires)
head(prediction)
```

```
##         1         2         3         4         5         6
## 12.751744  4.216900  7.509481 11.481738  2.242917 22.796874
```

```
rm(list = ls())
```