Regresion Multiple

2024-09-24

```r
# Cargar el archivo CSV
datos <- read.csv("documents/AlCorte.csv", header = TRUE)
```

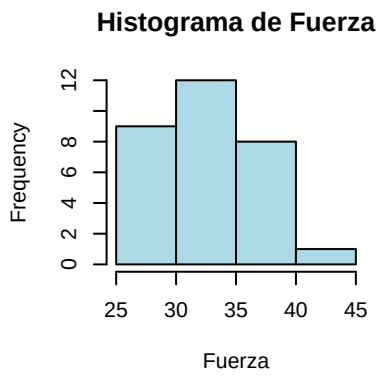
```r
# Medidas descriptivas
summary(datos)
```

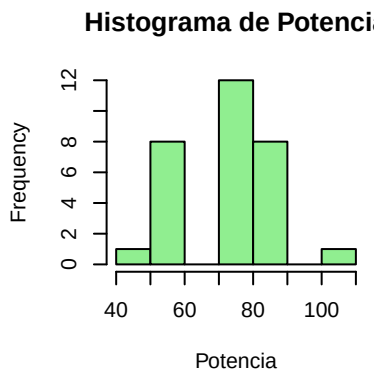
```
##      Fuerza      Potencia    Temperatura      Tiempo    Resistencia
##  Min.   :25   Min.   : 45   Min.   :150   Min.   :10   Min.   :22.70
##  1st Qu.:30   1st Qu.: 60   1st Qu.:175   1st Qu.:15   1st Qu.:34.67
##  Median :35   Median : 75   Median :200   Median :20   Median :38.60
##  Mean   :35   Mean   : 75   Mean   :200   Mean   :20   Mean   :38.41
##  3rd Qu.:40   3rd Qu.: 90   3rd Qu.:225   3rd Qu.:25   3rd Qu.:42.70
##  Max.   :45   Max.   :105   Max.   :250   Max.   :30   Max.   :58.70
```

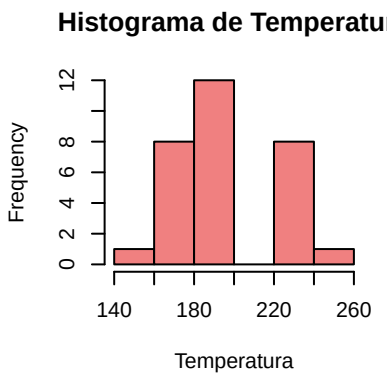
```r
# Desviación estándar para cada variable
sapply(datos, sd)
```

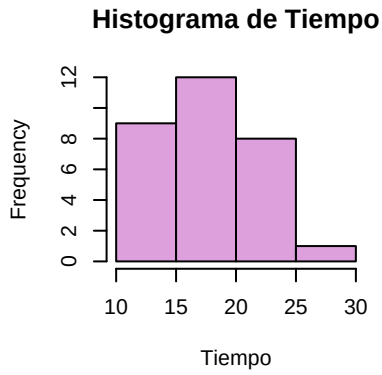
```
##      Fuerza    Potencia Temperatura      Tiempo Resistencia
##    4.548588   13.645765   22.742941    4.548588    8.954403
```

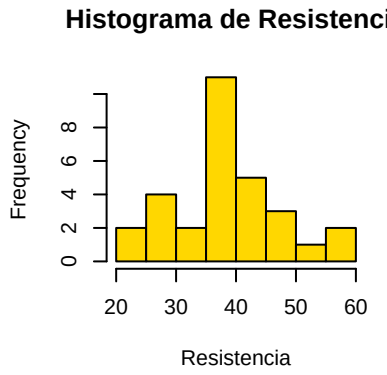
```r
# Histograma de cada variable
par(mfrow=c(2,3))  # Organiza los gráficos en una cuadrícula de 2 filas y 3 columnas

# Histograma de Fuerza
hist(datos$Fuerza, main = "Histograma de Fuerza", xlab = "Fuerza", col = "lightblue", border = "black")

# Histograma de Potencia
hist(datos$Potencia, main = "Histograma de Potencia", xlab = "Potencia", col = "lightgreen", border = "

# Histograma de Temperatura
hist(datos$Temperatura, main = "Histograma de Temperatura", xlab = "Temperatura", col = "lightcoral", bo

# Histograma de Tiempo
hist(datos$Tiempo, main = "Histograma de Tiempo", xlab = "Tiempo", col = "plum", border = "black")

# Histograma de Resistencia
hist(datos$Resistencia, main = "Histograma de Resistencia", xlab = "Resistencia", col = "gold", border =
```



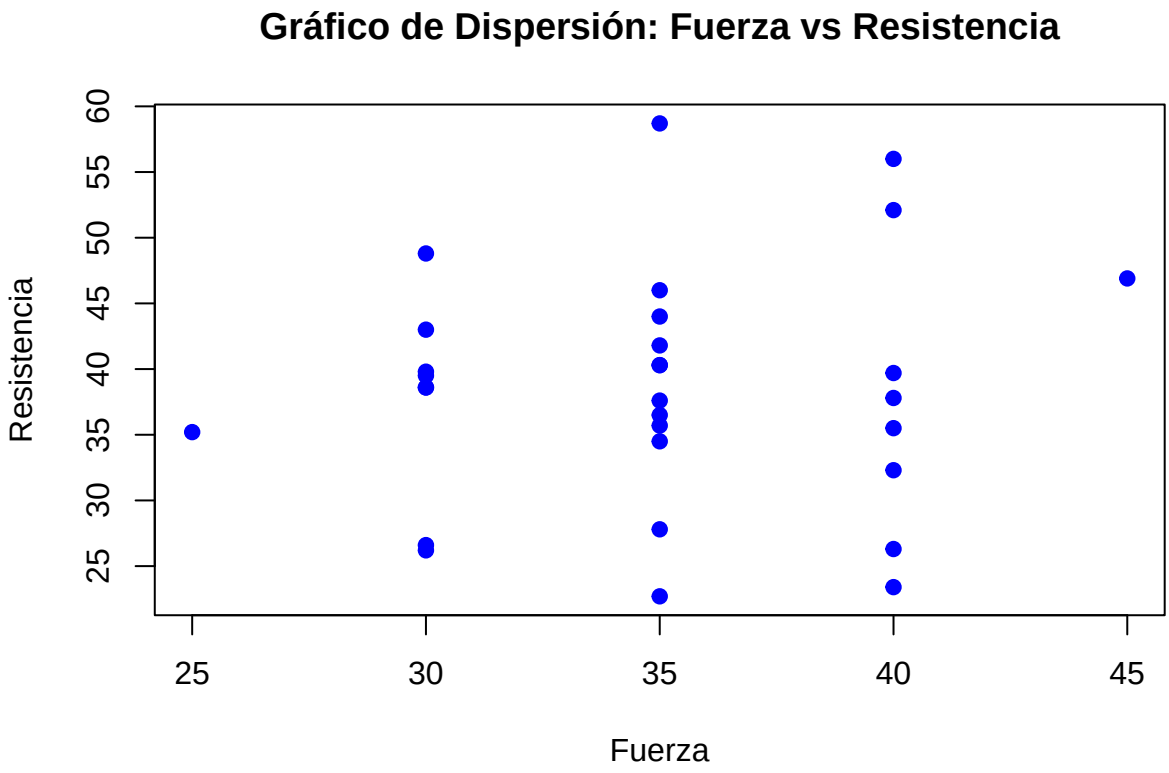




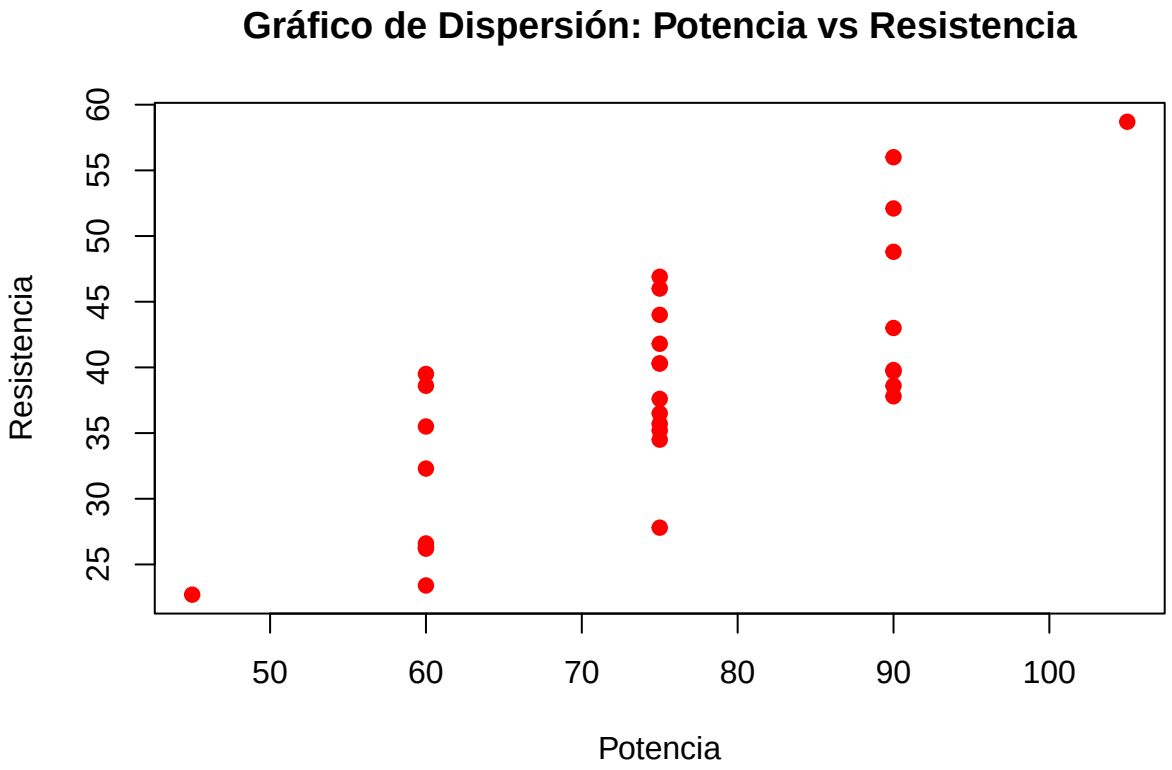





```r
# Gráfico de dispersión entre Fuerza y Resistencia
plot(datos$Fuerza, datos$Resistencia, main = "Gráfico de Dispersión: Fuerza vs Resistencia",
     xlab = "Fuerza", ylab = "Resistencia", col = "blue", pch = 19)
```



```r
# Gráfico de dispersión entre Potencia y Resistencia
plot(datos$Potencia, datos$Resistencia, main = "Gráfico de Dispersión: Potencia vs Resistencia",
     xlab = "Potencia", ylab = "Resistencia", col = "red", pch = 19)
```




```r
# Matriz de correlación
correlacion <- cor(datos)
print(correlacion)
```

```
##                Fuerza  Potencia Temperatura    Tiempo Resistencia
## Fuerza      1.0000000 0.0000000   0.0000000 0.0000000   0.1075208
## Potencia    0.0000000 1.0000000   0.0000000 0.0000000   0.7594185
## Temperatura 0.0000000 0.0000000   1.0000000 0.0000000   0.3293353
## Tiempo      0.0000000 0.0000000   0.0000000 1.0000000   0.1312262
## Resistencia 0.1075208 0.7594185   0.3293353 0.1312262   1.0000000
```

```r
# Modelo de regresión lineal múltiple usando todas las variables predictoras
modelo_completo <- lm(Resistencia ~ Fuerza + Potencia + Temperatura + Tiempo, data = datos)

# Resumen del modelo completo
summary(modelo_completo)
```

```
##
## Call:
## lm(formula = Resistencia ~ Fuerza + Potencia + Temperatura +
##     Tiempo, data = datos)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.0900  -1.7608  -0.3067   2.4392   7.5933
##
## Coefficients:
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -37.47667   13.09964  -2.861  0.00841 **
## Fuerza        0.21167    0.21057   1.005  0.32444
## Potencia      0.49833    0.07019   7.100 1.93e-07 ***
## Temperatura   0.12967    0.04211   3.079  0.00499 **
## Tiempo        0.25833    0.21057   1.227  0.23132
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.158 on 25 degrees of freedom
## Multiple R-squared:  0.714,  Adjusted R-squared:  0.6682
## F-statistic:  15.6 on 4 and 25 DF,  p-value: 1.592e-06
```

```
# Selección de modelo paso a paso (Stepwise)
modelo <- step(modelo_completo, direction = "both")
```

```
## Start:  AIC=102.96
## Resistencia ~ Fuerza + Potencia + Temperatura + Tiempo
##
##               Df Sum of Sq     RSS    AIC
## - Fuerza       1     26.88  692.00 102.15
## - Tiempo       1     40.04  705.16 102.72
## <none>                      665.12 102.96
## - Temperatura  1    252.20  917.32 110.61
## - Potencia     1   1341.01 2006.13 134.08
##
## Step:  AIC=102.15
## Resistencia ~ Potencia + Temperatura + Tiempo
##
##               Df Sum of Sq     RSS    AIC
## - Tiempo       1     40.04  732.04 101.84
## <none>                      692.00 102.15
## + Fuerza       1     26.88  665.12 102.96
## - Temperatura  1    252.20  944.20 109.47
## - Potencia     1   1341.02 2033.02 132.48
##
## Step:  AIC=101.84
## Resistencia ~ Potencia + Temperatura
##
##               Df Sum of Sq     RSS    AIC
## <none>                      732.04 101.84
## + Tiempo       1     40.04  692.00 102.15
## + Fuerza       1     26.88  705.16 102.72
## - Temperatura  1    252.20  984.24 108.72
## - Potencia     1   1341.01 2073.06 131.07
```

```
# Resumen del mejor modelo encontrado
summary(modelo)
```

```
##
## Call:
## lm(formula = Resistencia ~ Potencia + Temperatura, data = datos)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -11.3233  -2.8067  -0.8483   3.1892   9.4600
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -24.90167   10.07207  -2.472  0.02001 *
## Potencia      0.49833    0.07086   7.033 1.47e-07 ***
## Temperatura   0.12967    0.04251   3.050  0.00508 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.207 on 27 degrees of freedom
## Multiple R-squared:  0.6852, Adjusted R-squared:  0.6619
## F-statistic: 29.38 on 2 and 27 DF,  p-value: 1.674e-07
```

```
# Comparación de modelos por AIC
AIC(modelo_completo, modelo)
```

```
##                  df      AIC
## modelo_completo  6 190.0994
## modelo           4 188.9755
```

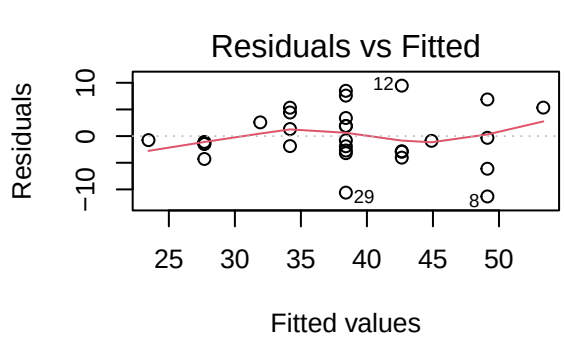
```
# Comparar R² ajustado
summary(modelo_completo)$adj.r.squared
```

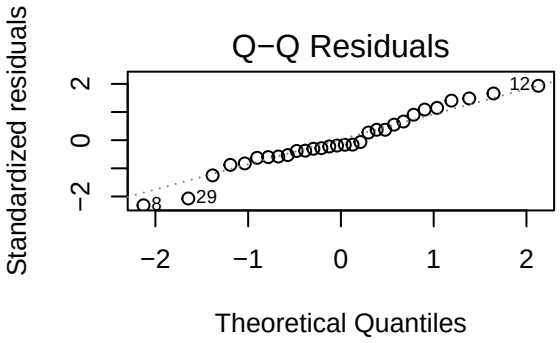
```
## [1] 0.6681928
```

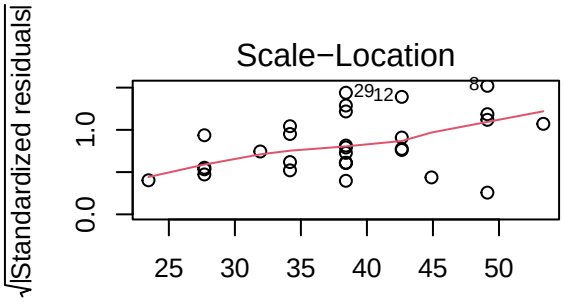
```
summary(modelo)$adj.r.squared
```

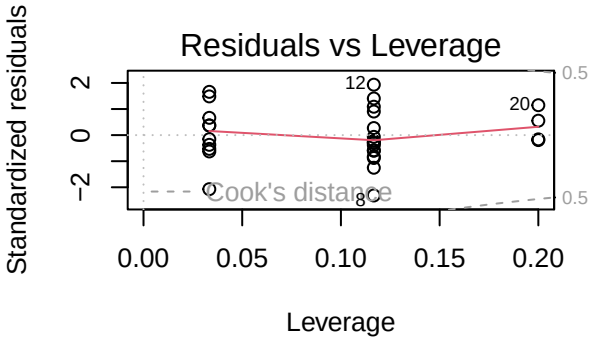
```
## [1] 0.6618581
```

```
# Diagnóstico del modelo: gráficos de diagnóstico
par(mfrow = c(2, 2))
plot(modelo)
```





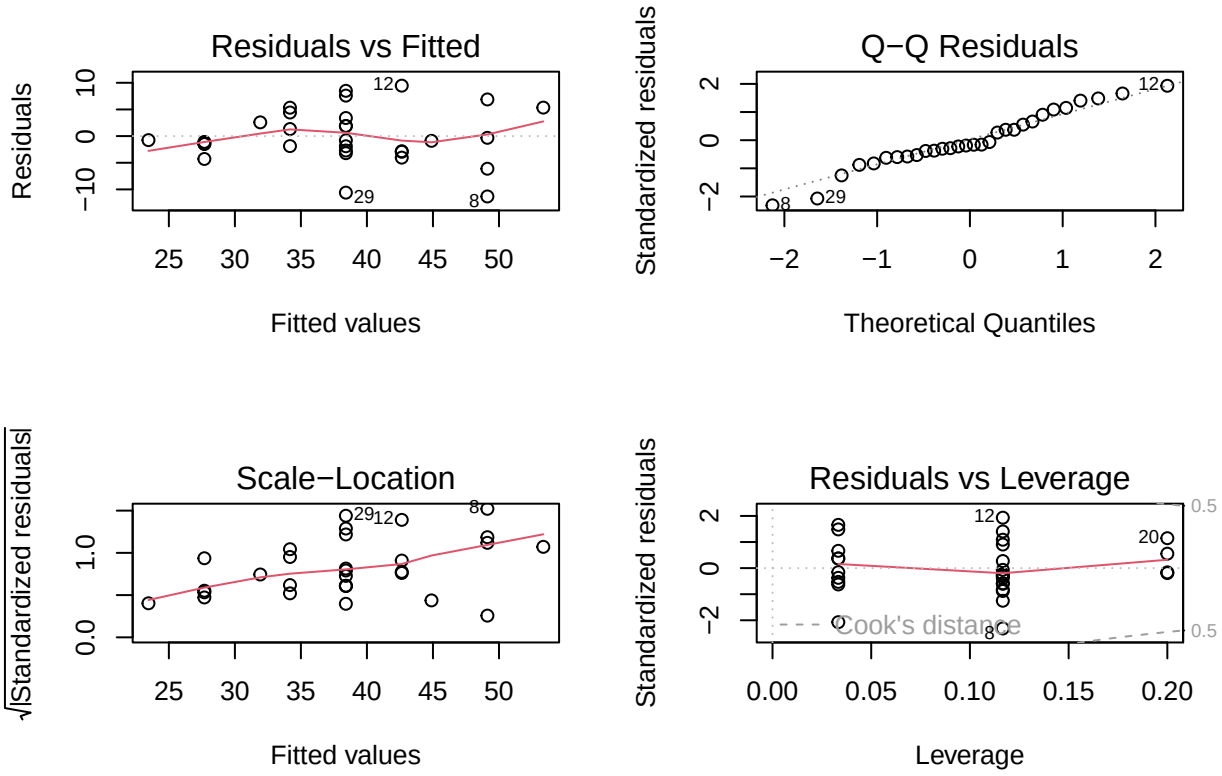





```
# Predicciones con el mejor modelo
predicciones <- predict(modelo, newdata = datos)
predicciones
```

```
##        1        2        3        4        5        6        7        8
## 27.69000 27.69000 42.64000 42.64000 34.17333 34.17333 49.12333 49.12333
##        9       10       11       12       13       14       15       16
## 27.69000 27.69000 42.64000 42.64000 34.17333 34.17333 49.12333 49.12333
##       17       18       19       20       21       22       23       24
## 38.40667 38.40667 23.45667 53.35667 31.92333 44.89000 38.40667 38.40667
##       25       26       27       28       29       30
## 38.40667 38.40667 38.40667 38.40667 38.40667 38.40667
```

```
# Gráficos de diagnóstico del modelo
par(mfrow = c(2, 2))
plot(modelo)
```




```
# Test de normalidad de Shapiro-Wilk para los residuos
shapiro.test(residuals(modelo))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(modelo)
## W = 0.96588, p-value = 0.4333
```

```
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
# Prueba de Breusch-Pagan para homocedasticidad
bptest(modelo)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  modelo
## BP = 4.0043, df = 2, p-value = 0.135
```

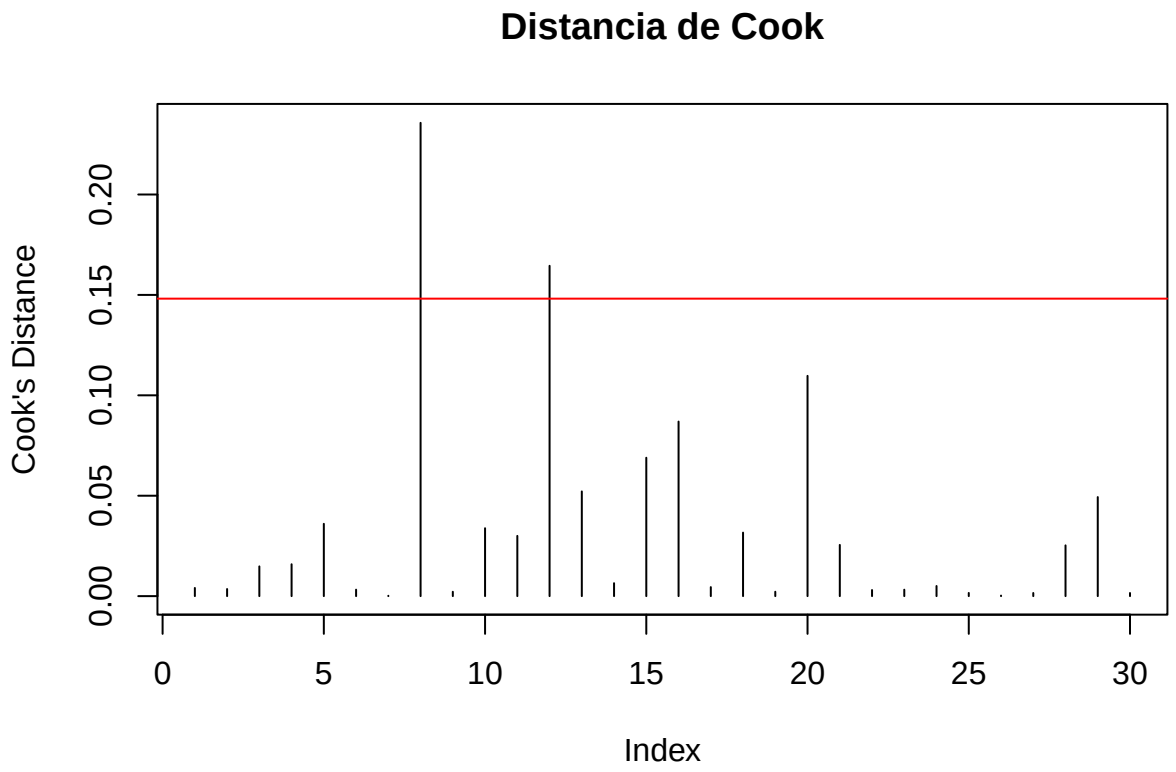
```
library(car)
```

```
## Loading required package: carData
```

```r
# Calcular el VIF (Variance Inflation Factor) para cada predictor
vif(modelo)
```

```
##    Potencia Temperatura
##           1           1
```

```r
# Distancia de Cook
plot(cooks.distance(modelo), main = "Distancia de Cook", ylab = "Cook's Distance", type = "h")

# Líneas para identificar observaciones influyentes
abline(h = 4/(nrow(datos)-length(coef(modelo))), col = "red")
```




```r
# Resumen del modelo para validar coeficientes y significancia
summary(modelo)
```

```
##
## Call:
## lm(formula = Resistencia ~ Potencia + Temperatura, data = datos)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.3233  -2.8067  -0.8483   3.1892   9.4600
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -24.90167   10.07207  -2.472  0.02001 *
## Potencia      0.49833    0.07086   7.033 1.47e-07 ***
## Temperatura   0.12967    0.04251   3.050  0.00508 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 5.207 on 27 degrees of freedom
## Multiple R-squared:  0.6852, Adjusted R-squared:  0.6619
## F-statistic: 29.38 on 2 and 27 DF,  p-value: 1.674e-07
```

```
# Calcular AIC y BIC del modelo
AIC(modelo)
```

```
## [1] 188.9755
```

```
BIC(modelo)
```

```
## [1] 194.5803
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

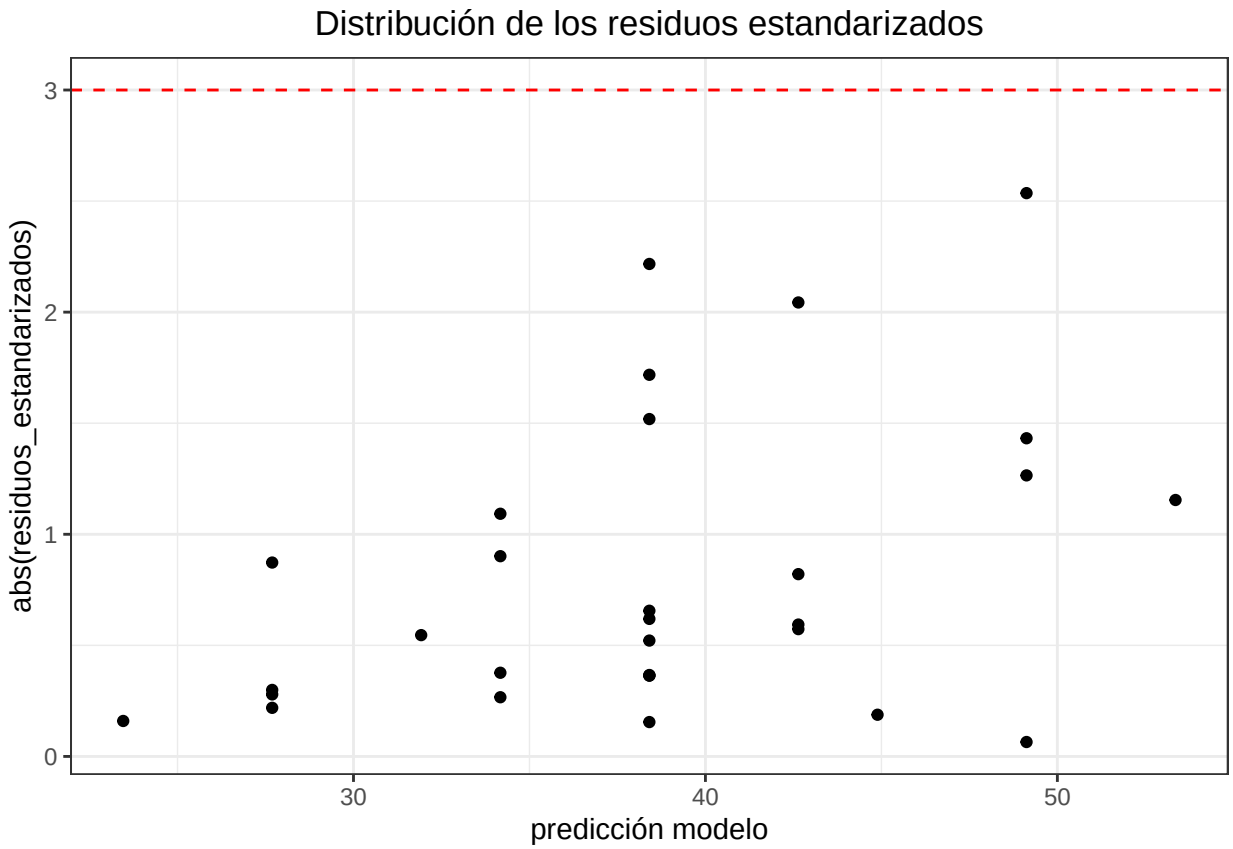
```
## The following object is masked from 'package:car':
##
##     recode
```

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

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

```
datos$residuos_estandarizados <- rstudent(modelo)
#Introduce una columma en Datos con los residuos estandarizados de los n datos
```

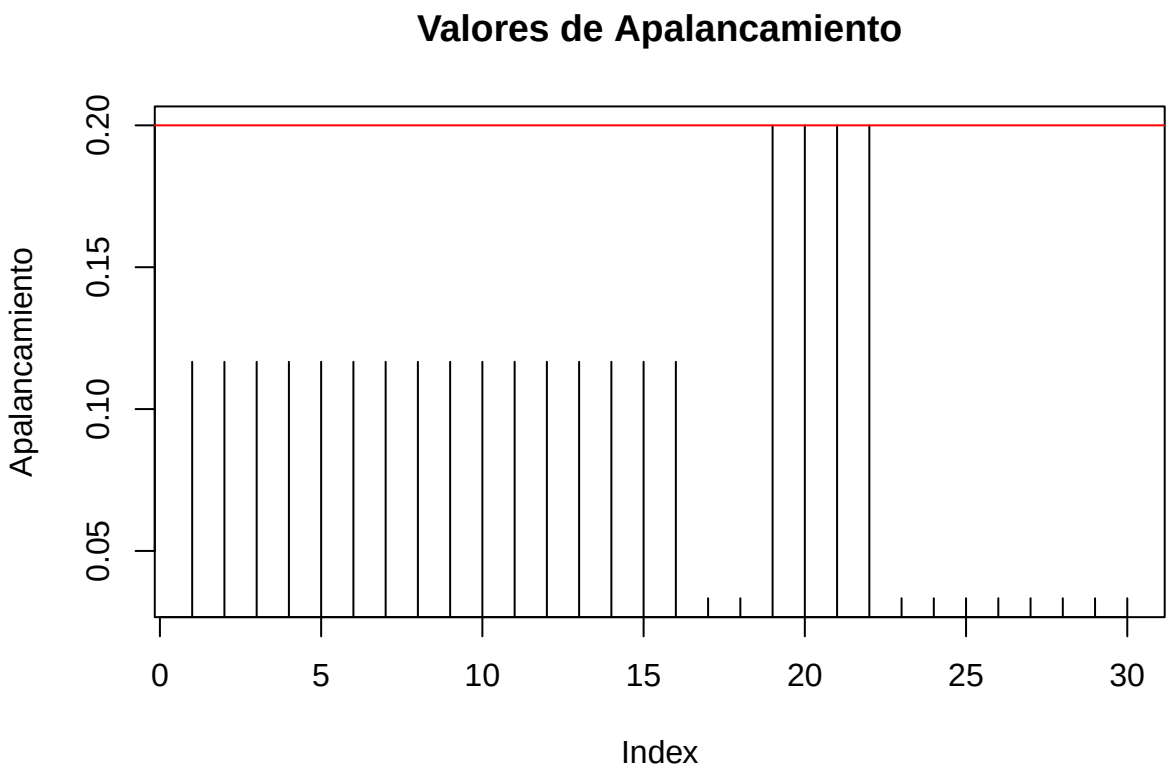
```
library(ggplot2)
ggplot(data = datos, aes(x = predict(modelo), y = abs(residuos_estandarizados))) +
geom_hline(yintercept = 3, color = "red", linetype = "dashed") +
# se identifican en rojo observaciones con residuos estandarizados absolutos > 3
geom_point(aes(color = ifelse(abs(residuos_estandarizados) > 3, 'red', 'black'))) +
scale_color_identity() +
labs(title = "Distribución de los residuos estandarizados", x = "predicción modelo") +
theme_bw() + theme(plot.title = element_text(hjust = 0.5))
```




```
Atipicos = which(abs(datos$residuos_estandarizados)>3)
datos[Atipicos, ]
```

```
## [1] Fuerza                  Potencia              Temperatura
## [4] Tiempo                  Resistencia           residuos_estandarizados
## <0 rows> (or 0-length row.names)
```

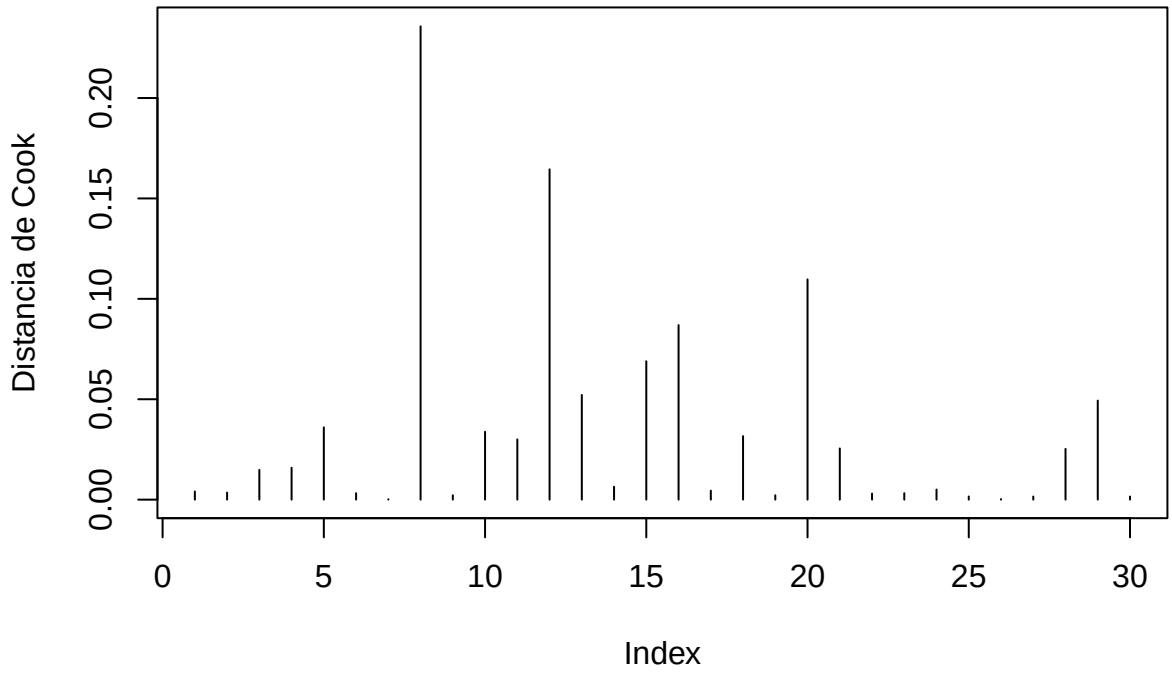
```
leverage = hatvalues(modelo)
#Calcula el leverage de los n datos
plot(leverage, type="h", main="Valores de Apalancamiento", ylab="Apalancamiento")
abline(h = 2*mean(leverage), col="red") # Límite comúnmente usado
```




```
high_leverage_points = which(leverage > 2*mean(leverage))
datos[high_leverage_points, ]
```

```
##    Fuerza Potencia Temperatura Tiempo Resistencia residuos_estandarizados
## 19     35       45         200     20        22.7                -0.159511
## 20     35      105         200     20        58.7                 1.154355
```

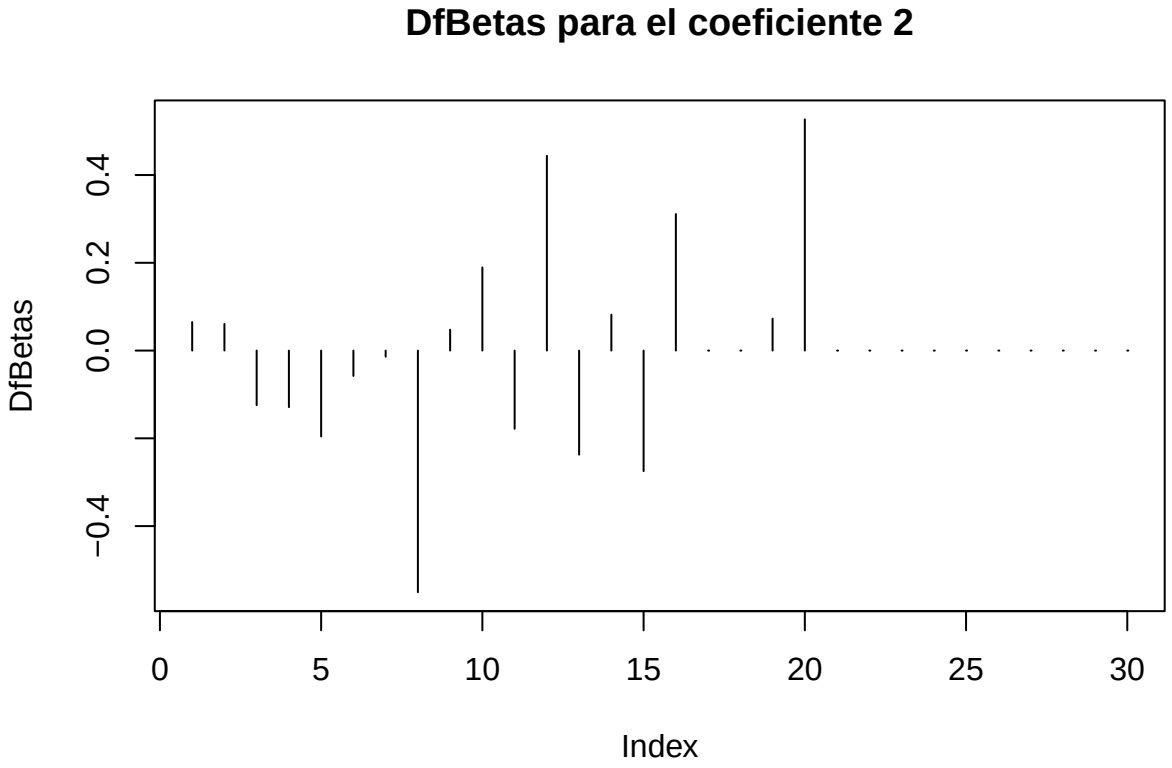
```
cooksd <- cooks.distance(modelo)
#Calcula la distancia de Cook de los n datos
plot(cooksd, type="h", main="Distancia de Cook", ylab="Distancia de Cook")
abline(h = 1, col="red") # Límite comúnmente usado
```




```
puntos_influyentes = which(cooksd > 1)
datos[puntos_influyentes, ]
```

```
## [1] Fuerza                  Potencia                Temperatura
## [4] Tiempo                  Resistencia             residuos_estandarizados
## <0 rows> (or 0-length row.names)
```

```
dfbetas_values = dfbetas(modelo)
#Calcula la DfBeta de los n datos para cada
plot(dfbetas_values[, 2], type="h", main="DfBetas para el coeficiente 2",
ylab="DfBetas")
abline(h = c(-1, 1), col="red") # Límites comunes
```



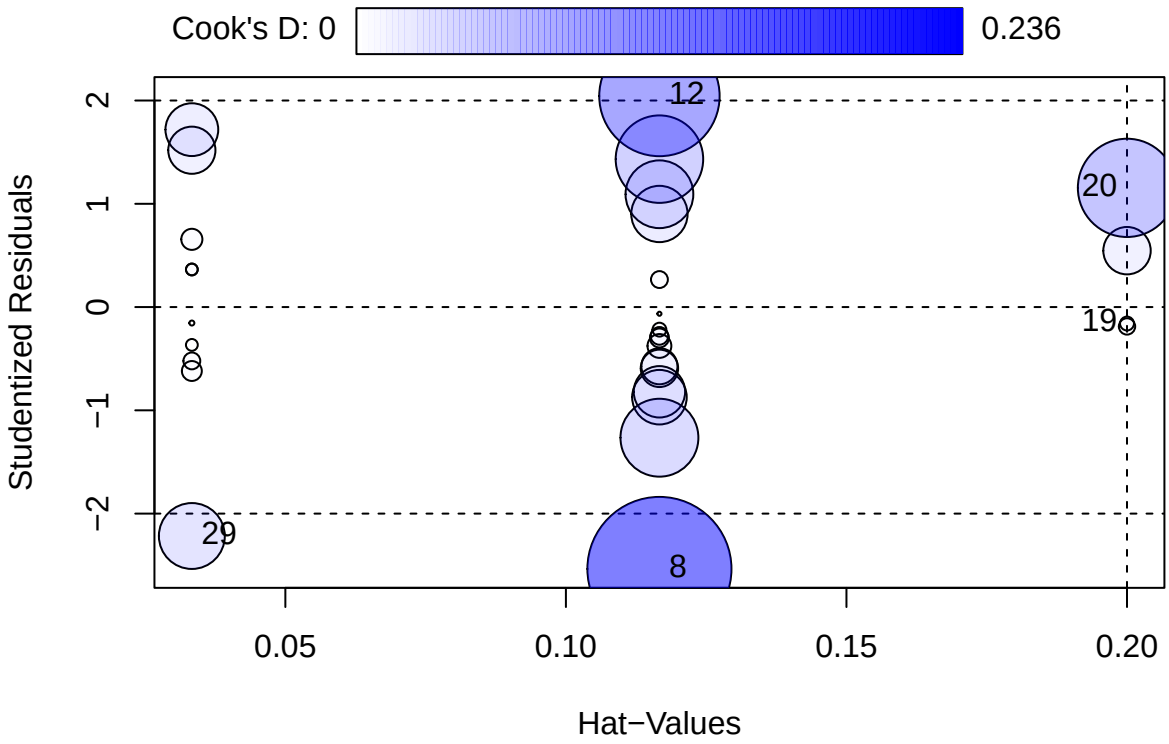

```
puntos_influyentes = which(abs(dfbetas_values[, 2]) > 1)
```

```
influencia = influence.measures(modelo)
#Calcula las medidas de los n datos
summary(influencia)
```

```
## Potentially influential observations of
##   lm(formula = Resistencia ~ Potencia + Temperatura, data = datos) :
##
##    dfb.1_ dfb.Ptnc dfb.Tmpr dffit cov.r   cook.d hat
## 8   0.71  -0.55    -0.55    -0.92  0.65_*  0.24   0.12
## 19 -0.04   0.07     0.00    -0.08  1.40_*  0.00   0.20
## 21  0.22   0.00    -0.25     0.27  1.35_*  0.03   0.20
## 22  0.07   0.00    -0.09    -0.09  1.39_*  0.00   0.20
```

```
# Detecta los datos con posible influencia
```

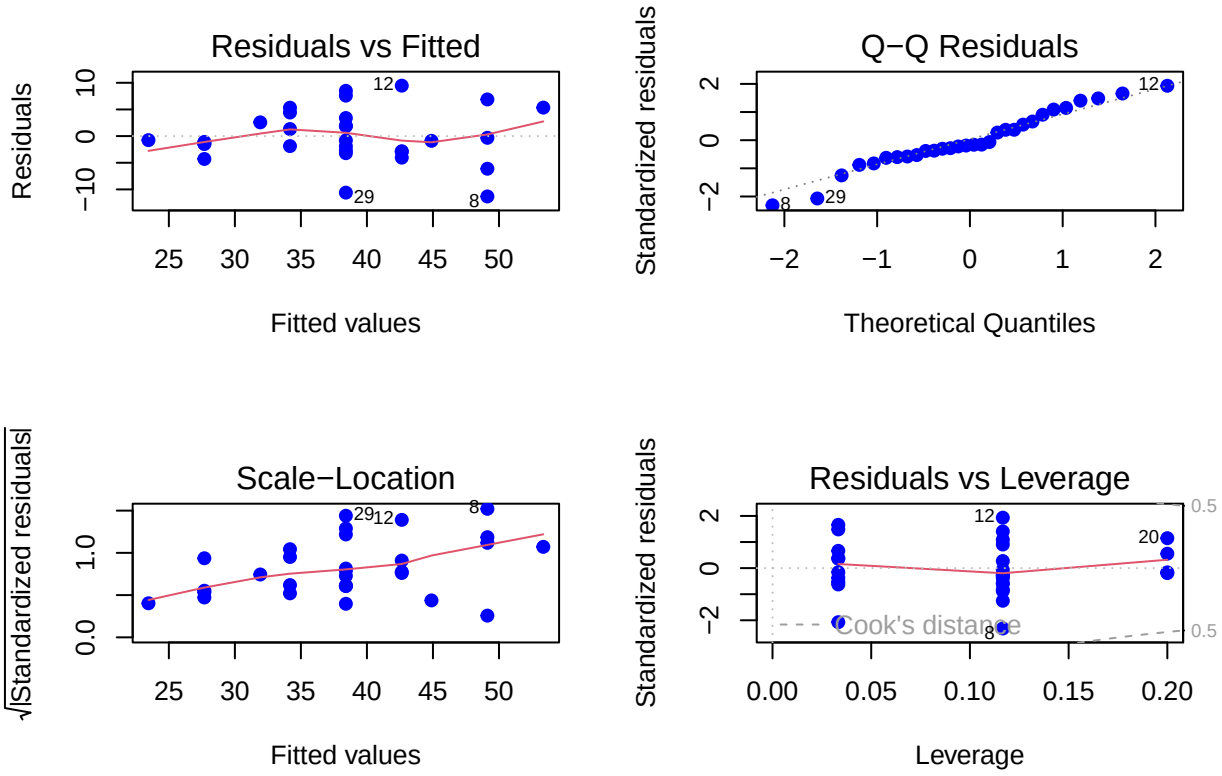
```
library(car)
influencePlot(modelo)
```




```
##      StudRes        Hat       CookD
## 8  -2.535832 0.11666667 0.235696235
## 12  2.043589 0.11666667 0.164507739
## 19 -0.159511 0.20000000 0.002199712
## 20  1.154355 0.20000000 0.109693544
## 29 -2.216952 0.03333333 0.049338917
```

```
# grafica los residuos con estandarización
```

```
par(mfrow=c(2, 2))
plot(modelo, col='blue', pch=19)
```


Residuals vs Fitted
Residuals
Fitted values
Q−Q Residuals
Standardized residuals
Theoretical Quantiles
Scale−Location
√|Standardized residuals|
Fitted values
Residuals vs Leverage
Standardized residuals
Cook's distance
Leverage