

# Práctica Tema 6

Álvaro Miranda García

2023-03-27

**1. Instale y cargue las siguientes librerías: “MASS”, “caret”, “stat”, “olsrr”, “kable”, “kableExtra”, “knitr” y “rmarkdown”**

Han sido instaladas.

**2. Cree 2 variables almacenadas como vector: “y\_cuentas” y “x\_distancia” a partir de los siguientes valores numéricos:**

```
y_cuentas = c(110,2,6,98,40,94,31,5,8,10)
x_distancia = c(1.1,100.2,90.3,5.4,57.5,6.6,34.7,65.8,57.9,86.1)
```

**3. Verifique el supuesto de linealidad de la variable explicativa incluyendo un contraste de hipótesis.**

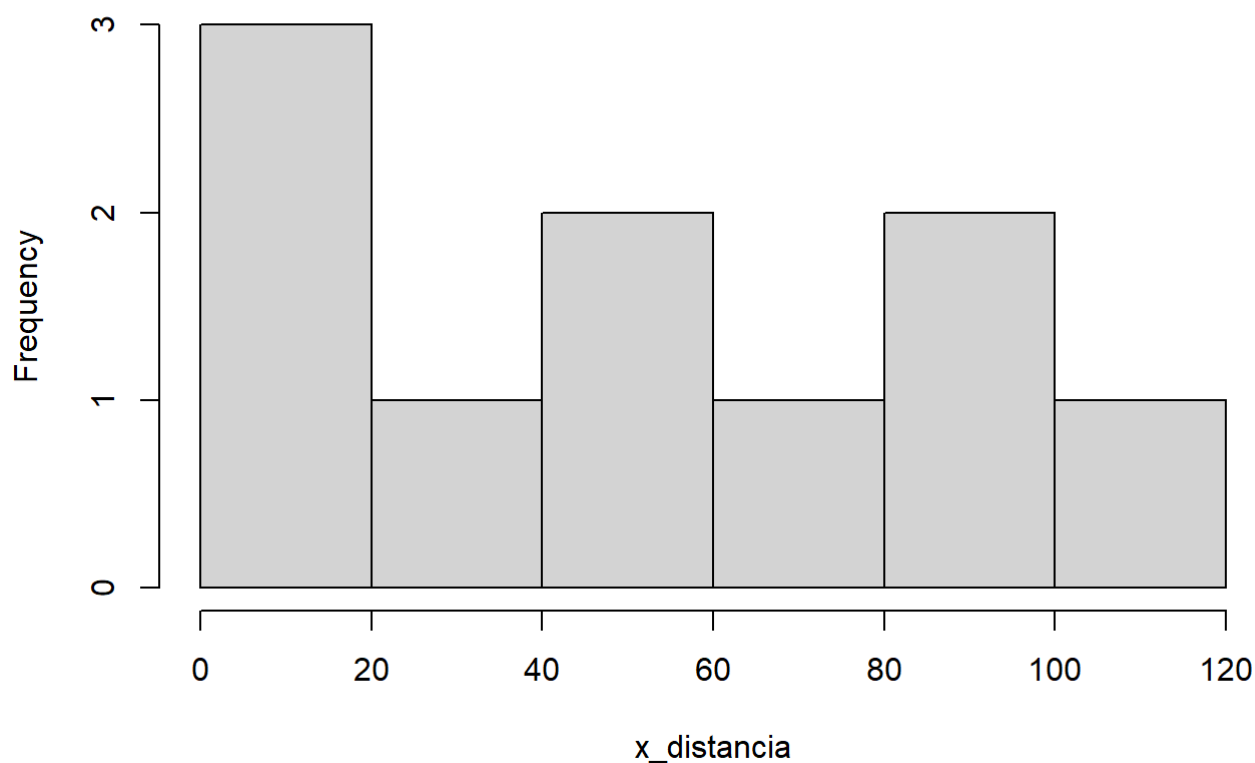
```
modelo <- lm(y_cuentas ~ x_distancia)
anova(modelo)
```

```
## Analysis of Variance Table
##
## Response: y_cuentas
##           Df Sum Sq Mean Sq F value    Pr(>F)
## x_distancia  1 14552.3   14552   47.399 0.0001265 ***
## Residuals    8   2456.1     307
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

**4. Verifique el supuesto de normalidad de la variable explicativa mediante su visualización en un histograma y un test de normalidad.**

```
hist(x_distancia)
```

## Histogram of x\_distancia



```
shapiro.test(x_distancia)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  x_distancia
## W = 0.90687, p-value = 0.2602
```

**5. Multiplique las variable de respuesta por la variable explicativa. Llama al objeto “xy”.**

```
xy = y_cuentas * x_distancia
xy
```

```
## [1] 121.0 200.4 541.8 529.2 2300.0 620.4 1075.7 329.0 463.2 861.0
```

**6. Eleve al cuadrado la variable explicativa. Llama al objeto “x\_cuadrado”.**

```
x_cuadrado <- x_distancia^2
x_cuadrado
```

```
## [1] 1.21 10040.04 8154.09 29.16 3306.25 43.56 1204.09 4329.64
## [9] 3352.41 7413.21
```

**7. A continuación, almacena las variables: “y\_cuentas”, “x\_distancia”, “xy” y “x\_cuadrado” en un data frame llamado “tabla\_datos”**

```
tabla_datos <- data.frame(y_cuentas, x_distancia, xy, x_cuadrado)
```

8. Visualice el objeto “tabla\_datos” en una tabla en la consola a través de alguna de las funciones ofrecidas por la librería “kableExtra”.

```
library(kableExtra)
```

```
## Warning: package 'kableExtra' was built under R version 4.2.3
```

```
kable(tabla_datos, caption = "Tabla de Datos") %>%  
  kable_styling(full_width = F)
```

Tabla de Datos

y_cuentas	x_distancia	xy	x_cuadrado
110	1.1	121.0	1.21
2	100.2	200.4	10040.04
6	90.3	541.8	8154.09
98	5.4	529.2	29.16
40	57.5	2300.0	3306.25
94	6.6	620.4	43.56
31	34.7	1075.7	1204.09
5	65.8	329.0	4329.64
8	57.9	463.2	3352.41
10	86.1	861.0	7413.21

9. Realice el sumatorio de los valores almacenados en las 4 columnas del data frame “tabla\_datos”.

```
#primer sumatorio

sum1 = sum (tabla_datos$y_cuentas)

#segundo sumatorio

sum2 = (tabla_datos$x_distancia)

#tercer sumatorio

sum3 = (tabla_datos$xy)

#cuarto sumatorio

sum4 = (tabla_datos$x_cuadrado)
```

**10. Añade el sumatorio de las 4 columnas como un último registro en el data frame “tabla\_datos” de modo que tengamos en un solo objeto los valores junto con el sumatorio.**

```
sum.y_cuentas <- sum(tabla_datos$y_cuentas)
sum.x_distancia <- sum(tabla_datos$x_distancia)
sum.xy <- sum(tabla_datos$xy)
sum.x_cuadrado <- sum(tabla_datos$x_cuadrado)

row.sums <- c(sum.y_cuentas,sum.x_distancia,sum.xy,sum.x_cuadrado)

tabla_datos_final <- rbind(tabla_datos, row.sums)

tabla_datos_final
```

```
##      y_cuentas x_distancia      xy x_cuadrado
## 1         110          1.1  121.0         1.21
## 2           2        100.2  200.4       10040.04
## 3           6         90.3  541.8       8154.09
## 4          98          5.4  529.2         29.16
## 5          40         57.5 2300.0       3306.25
## 6          94          6.6  620.4         43.56
## 7          31         34.7 1075.7       1204.09
## 8           5         65.8  329.0       4329.64
## 9           8         57.9  463.2       3352.41
## 10          10         86.1  861.0       7413.21
## 11         404        505.6 7041.7      37873.66
```

$$\sigma^2 = \frac{\sum_{i=1}^n (y_i - \bar{y})^2}{n-1}$$

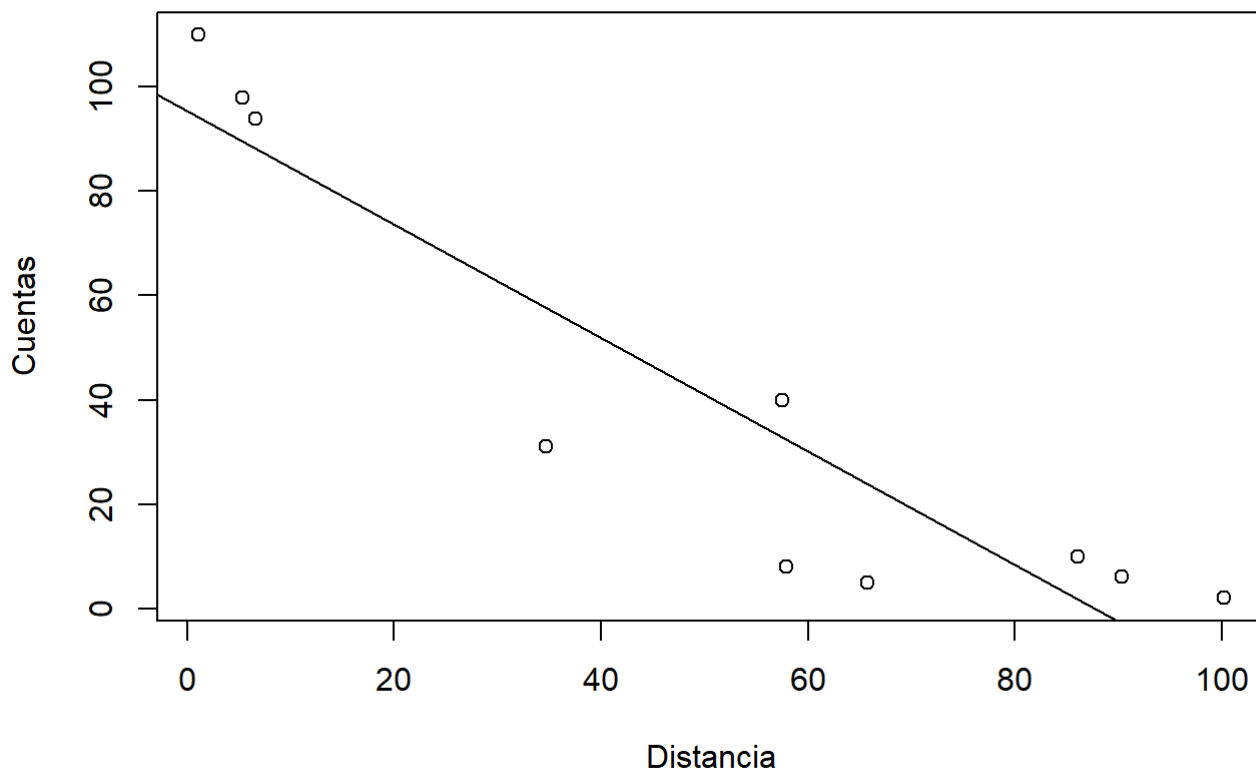
**11. Calcule la recta de regresión por el método de mínimos cuadrados (ordinario) a través de los datos incluidos en el data frame “tabla\_datos”.**

```
modelo <- lm(y_cuentas ~ x_distancia)
```

**12. Visualice en un gráfico de dispersión la recta de regresión, nube de puntos. Indique en el título la ecuación resultante y edite los nombre de los ejes.**

```
plot(x_distancia, y_cuentas, main = paste("Y =", round(modelo$coefficients[2],4),"* X + ", round(modelo$coefficients[1],4)), xlab = "Distancia", ylab = "Cuentas")
abline(modelo)
```

$$Y = -1.0872 * X + 95.371$$



**13. Calcule los residuos, residuos estandarizados y residuos estudentizados del modelo recién ajustado.**

```
residuos <- resid(modelo)
residuos
```

```
##          1          2          3          4          5          6          7
## 15.824925 15.570779  8.807066  8.500073  7.145471  5.804766 -26.643686
##          8          9         10
## -18.830406 -24.419631  8.240642
```

```
residuos_estandarizados <- rstandard(modelo)
residuos_estandarizados
```

```
##          1          2          3          4          5          6          7
## 1.0784816 1.0622593  0.5721649  0.5661008  0.4307982  0.3843280 -1.6213545
##          8          9         10
## -1.1448722 -1.4726329  0.5266739
```

```
residuos_estudentizados <- rstudent(modelo)
residuos_estudentizados
```

```
##           1           2           3           4           5           6           7
## 1.0912716 1.0721374 0.5465101 0.5404749 0.4077319 0.3628715 -1.8509341
##           8           9          10
## -1.1711614 -1.6134626 0.5014281
```

**14. Calcula el pronóstico o estimación del modelo para una observación que registra una distancia de 6.6km con respecto a la mina.**

```
predict(modelo, newdata = data.frame(x_distancia = 6.6))
```

```
##           1
## 88.19523
```

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$$

**15. Genera dos conjuntos aleatorios de datos: “entrenamiento” y “validación”**

```
set.seed(12345)
entrenamiento <- sample(1:nrow(tabla_datos), 0.7*nrow(tabla_datos))
validacion <- setdiff(1:nrow(tabla_datos), entrenamiento)
```

**16. Ajusta nuevamente el modelo con el conjunto de “entrenamiento”.**

```
modelo_entrenamiento <- lm(y_cuentas ~ x_distancia, data = tabla_datos, subset = entrenamiento)
```

**17. Interprete el valor asociado a los coeficientes de regresión y a R2. ¿Qué significan los asteriscos inmediatamente a la derecha de los valores arrojados tras ajustar el modelo?**

```
summary(modelo_entrenamiento)
```

```
##
## Call:
## lm(formula = y_cuentas ~ x_distancia, data = tabla_datos, subset = entrenamiento)
##
## Residuals:
##      3      8      2      5     10      9      7
## 0.3291 -11.6053  0.7475 19.6904  2.4546 -12.1311  0.5148
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  45.9719    15.2326   3.018  0.0295 *
## x_distancia  -0.4463     0.2073  -2.153  0.0839 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.63 on 5 degrees of freedom
## Multiple R-squared:  0.4811, Adjusted R-squared:  0.3773
## F-statistic: 4.636 on 1 and 5 DF, p-value: 0.08392
```

Los asteriscos significan que el coeficiente de regresión es significativo estadísticamente para un nivel de significación del 5%.

### 18. ¿Cómo se ha realizado el cálculo para los grados de libertad del modelo?

Los grados de libertad se calculan restando el número de observaciones menos el número de parámetros estimados. En este caso, es 8.

### 19. Especifique el total de varianza explicada y no explicada por el modelo.

```
#Varianza explicada:
ssr = sum(residuos^2)

#Varianza no explicada:
sse = sum((y_cuentas - predict(modelo))^2)
```

### 20. Aplique la validación cruzada simple para evaluar la robustez y capacidad predictiva del modelo.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.2.3
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
cv <- trainControl(method = "cv", number = 10)
modelo_cruzado <- train(y_cuentas ~ x_distancia, tabla_datos, method = "lm", trControl = cv)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

### 21. Verifique que no existen observaciones influyentes.

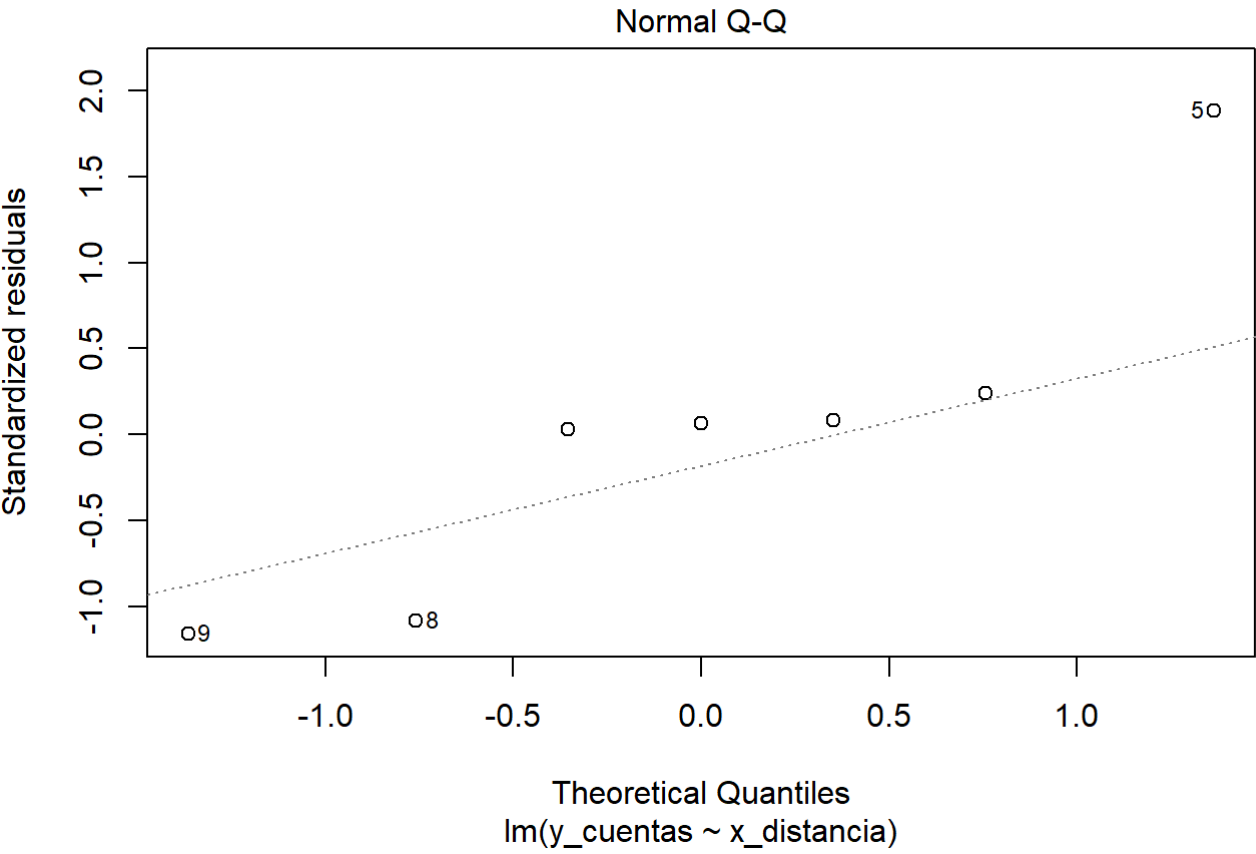
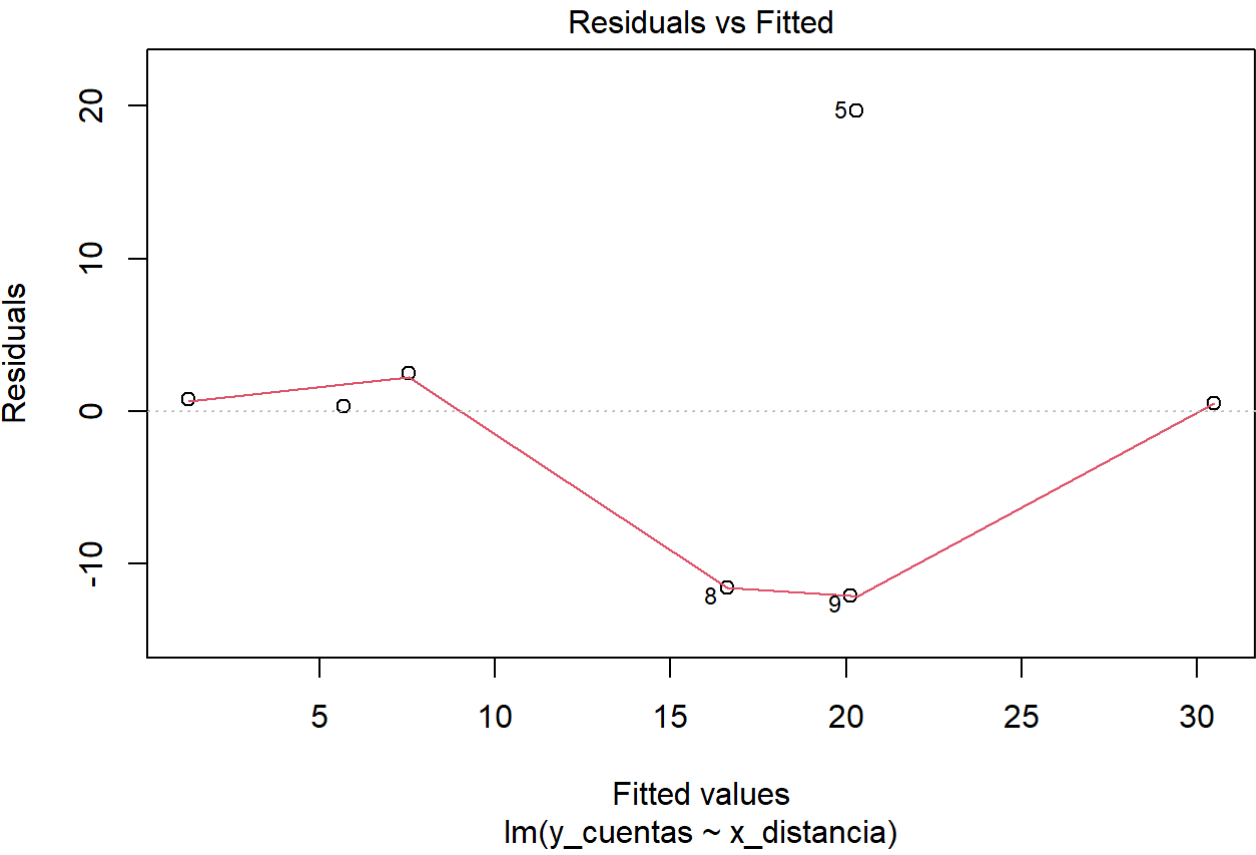
```
influence.measures(modelo_entrenamiento)
```

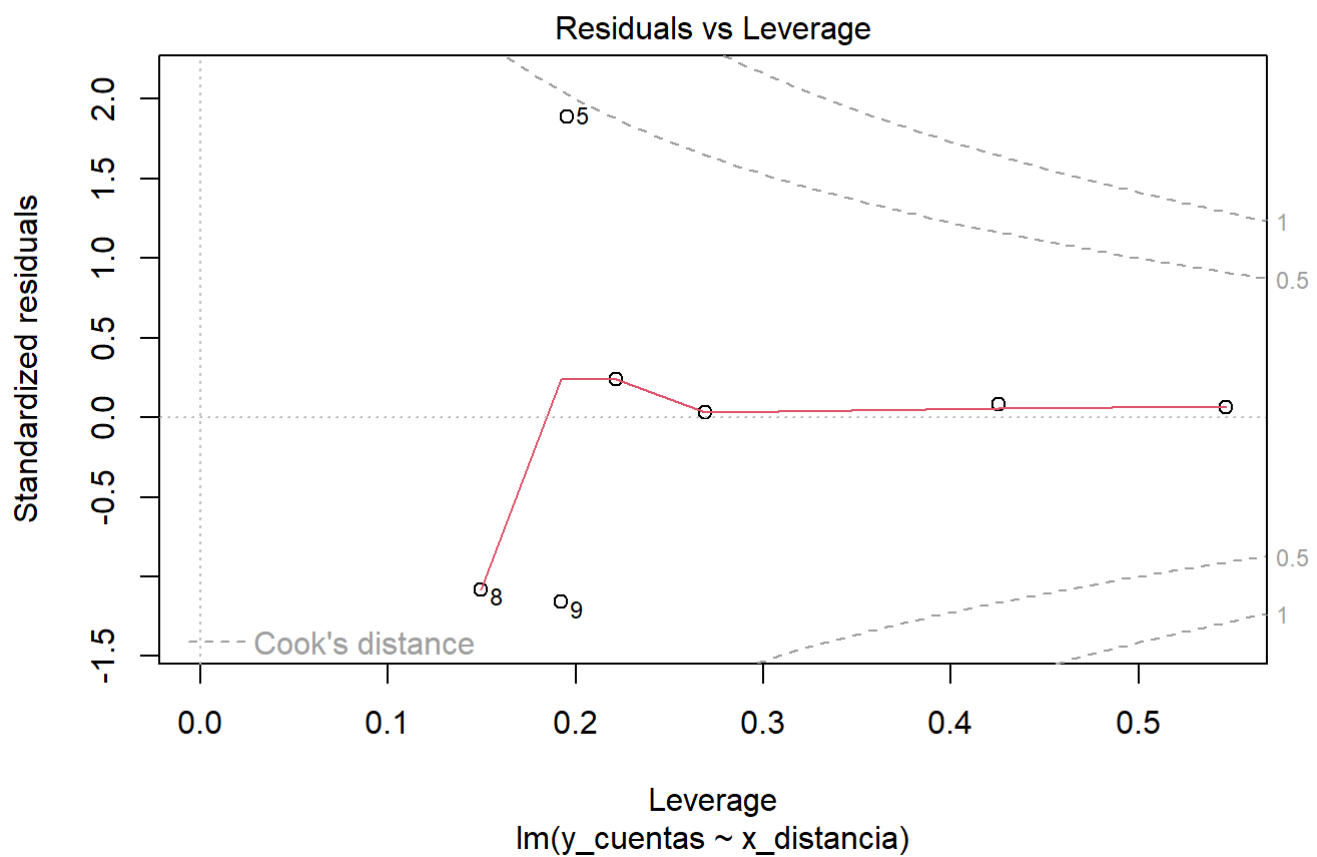
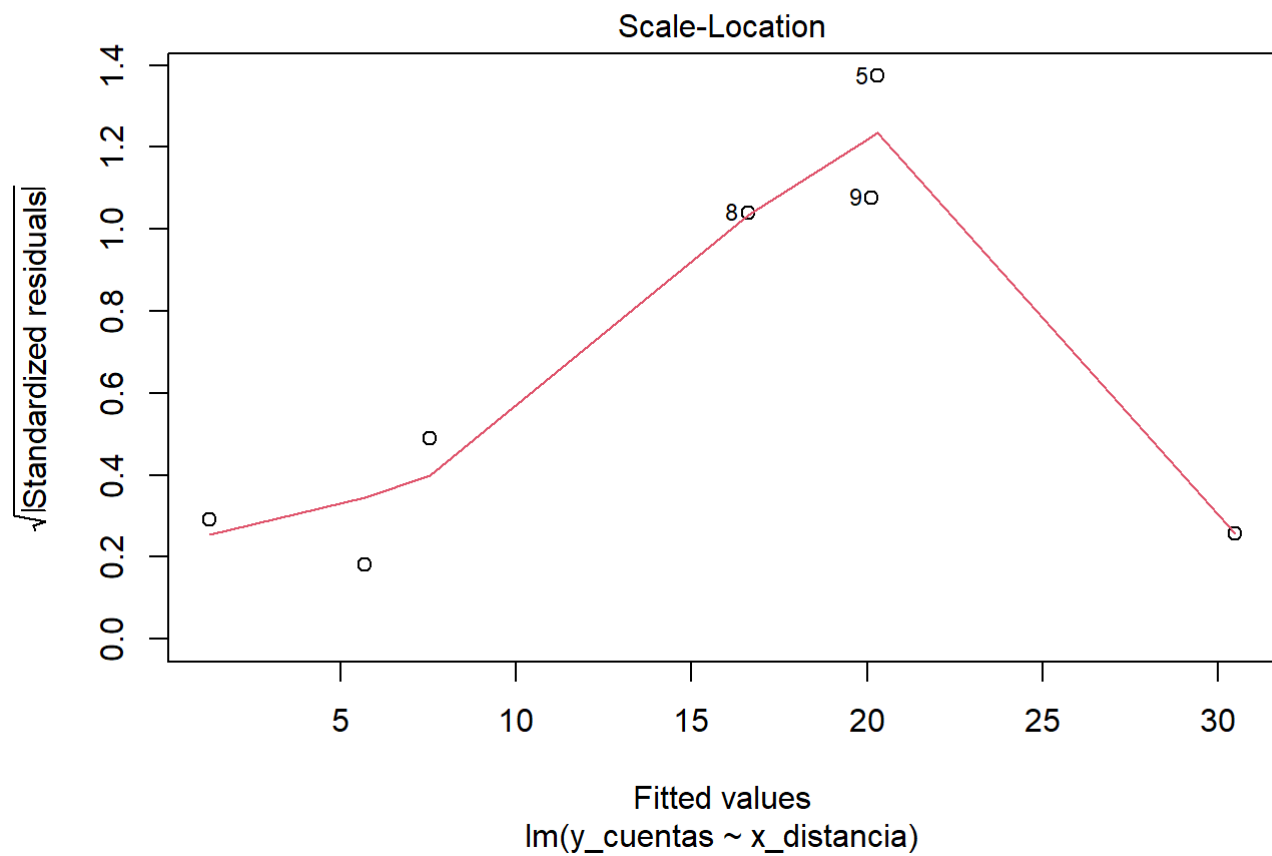
```
## Influence measures of
## lm(formula = y_cuentas ~ x_distancia, data = tabla_datos, subset = entrenamiento) :
##
##      dfb.1_ dfb.x_ds  dffit cov.r   cook.d   hat inf
## 3  -0.00801  0.0123  0.0180 2.137 0.000202 0.269
## 8  -0.22399  0.0974 -0.4635 1.078 0.102821 0.149
## 2  -0.04007  0.0533  0.0653 2.713 0.002665 0.426  *
## 5   1.15216 -0.8037  1.5504 0.161 0.432302 0.195  *
## 10 -0.03890  0.0684  0.1148 1.962 0.008141 0.222
## 9  -0.43445  0.2999 -0.5921 1.033 0.160104 0.192
## 7   0.06267 -0.0555  0.0646 3.441 0.002604 0.547  *
```

## 22. Verifique el supuesto de independencia de los residuos.

```
plot(modelo_entrenamiento)
```







23. Confirme que los errores del modelo permanecen constantes para todo el rango de estimaciones.

```
acf(resid(modelo_entrenamiento))
```

Series resid(modelo\_entrenamiento)

