A1- Regresión no lineal

Ricardo Salinas

1. Parte 1: Análisis de normalidad

```
#Accede a los datos de cars en R (data = cars)
d = cars
print(d)
```

```
##    speed dist
## 1      4    2
## 2      4   10
## 3      7    4
## 4      7   22
## 5      8   16
## 6      9   10
## 7     10   18
## 8     10   26
## 9     10   34
## 10    11   17
## 11    11   28
## 12    12   14
## 13    12   20
## 14    12   24
## 15    12   28
## 16    13   26
## 17    13   34
## 18    13   34
## 19    13   46
## 20    14   26
## 21    14   36
## 22    14   60
## 23    14   80
## 24    15   20
## 25    15   26
## 26    15   54
## 27    16   32
## 28    16   40
## 29    17   32
## 30    17   40
## 31    17   50
## 32    18   42
## 33    18   56
## 34    18   76
## 35    18   84
## 36    19   36
## 37    19   46
```

```
## 38    19   68
## 39    20   32
## 40    20   48
## 41    20   52
## 42    20   56
## 43    20   64
## 44    22   66
## 45    23   54
## 46    24   70
## 47    24   92
## 48    24   93
## 49    24  120
## 50    25   85
```

```r
#Prueba normalidad univariada de la velocidad y distancia (prueba con dos de las pruebas vistas en clase)
library(nortest)
lillie.test(d$speed)
```

```
##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  d$speed
## D = 0.068539, p-value = 0.8068
```

```r
lillie.test(d$dist)
```

```
##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  d$dist
## D = 0.12675, p-value = 0.04335
```

```r
shapiro.test(d$speed)
```

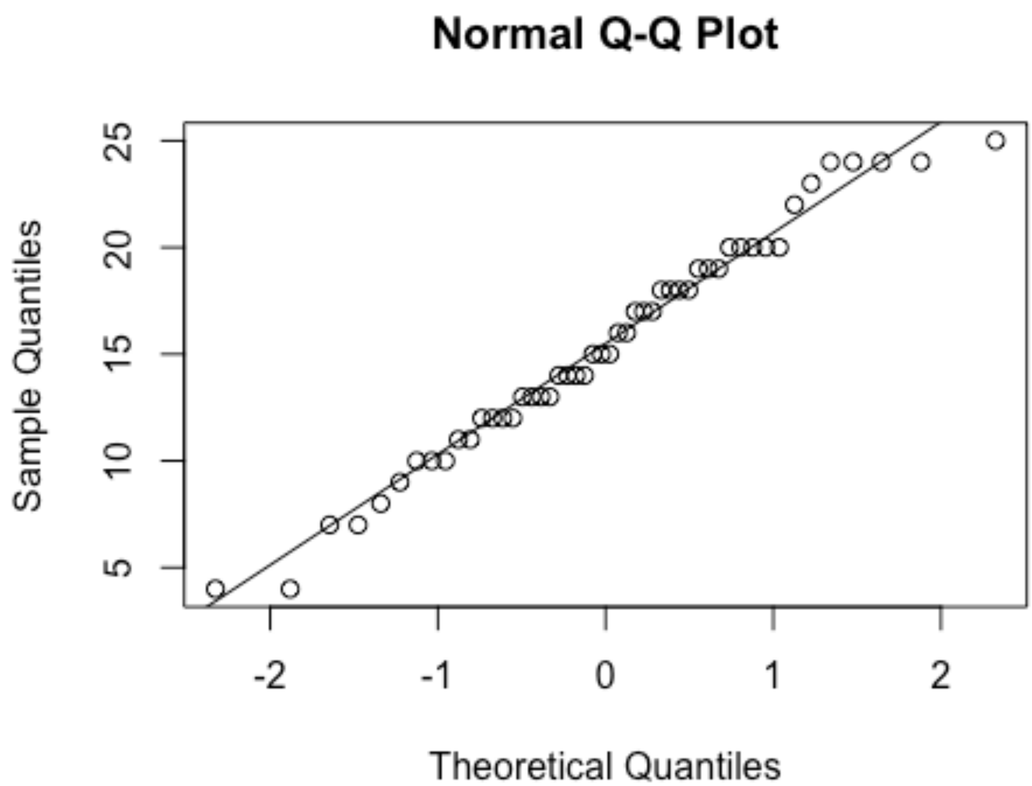
```
##
##  Shapiro-Wilk normality test
##
## data:  d$speed
## W = 0.97765, p-value = 0.4576
```

```r
shapiro.test(d$dist)
```

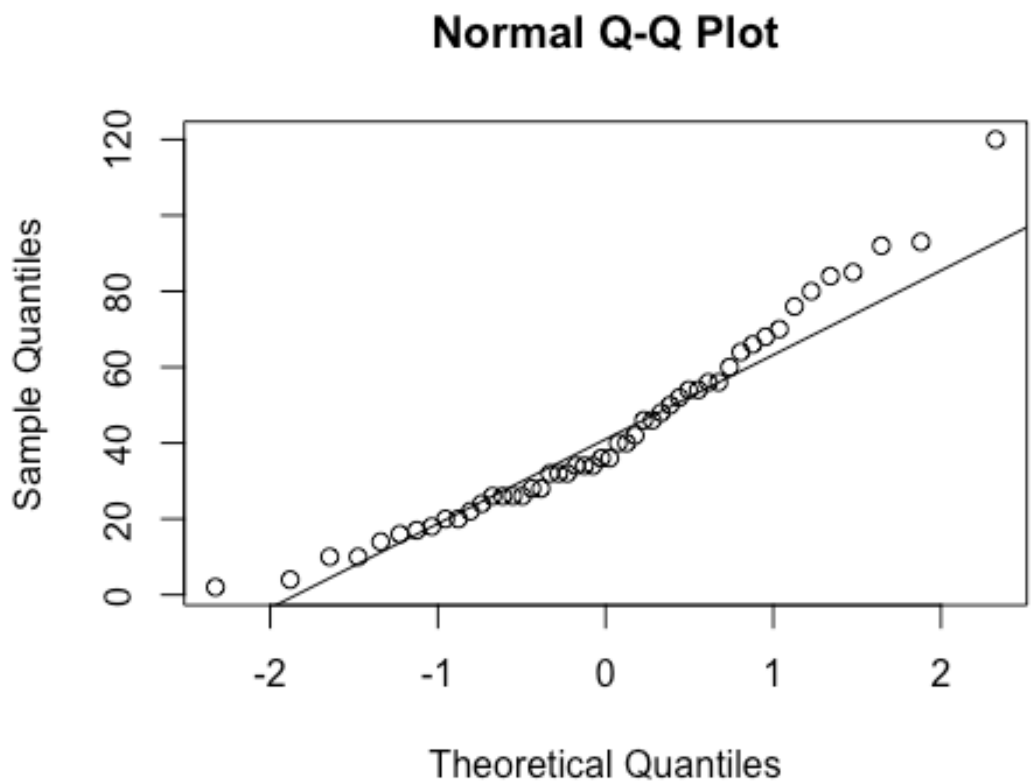
```
##
##  Shapiro-Wilk normality test
##
## data:  d$dist
## W = 0.95144, p-value = 0.0391
```

```r
#Realiza gráficos que te ayuden a identificar posibles alejamientos de normalidad:
```

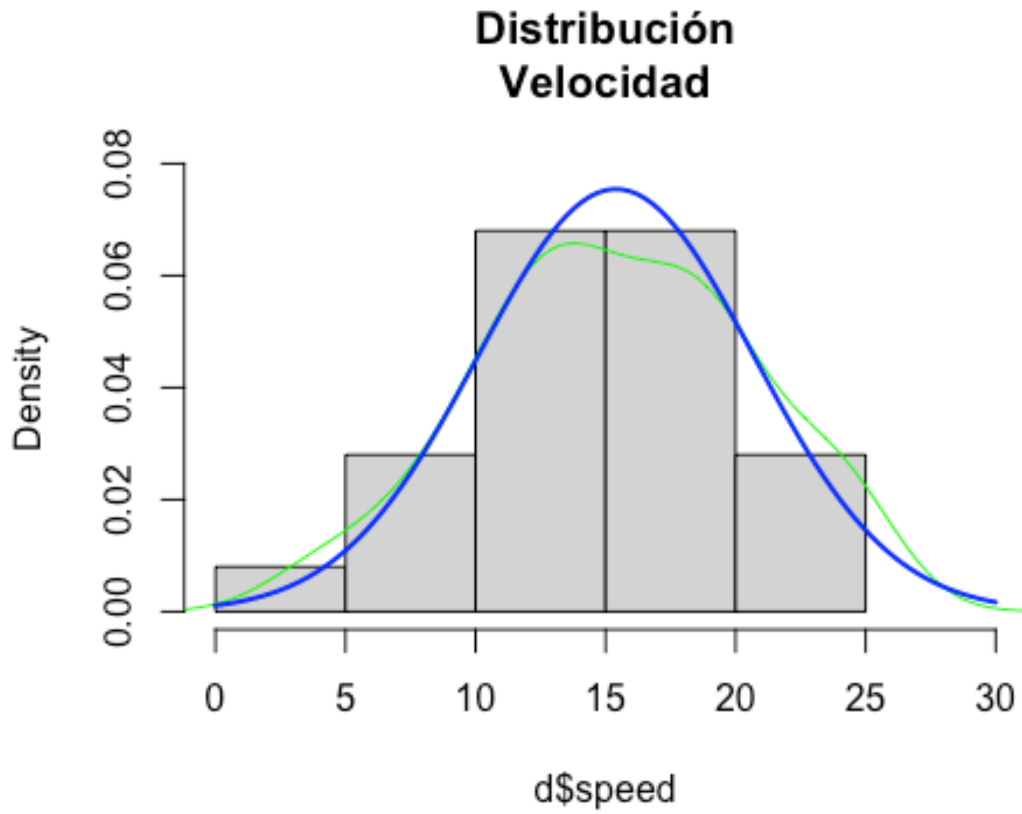
```
#Los datos y su respectivo QQPlot: qqnorm(datos) y qqline(datos) para cada variable
qqnorm(d$speed)
qqline(d$speed)
```



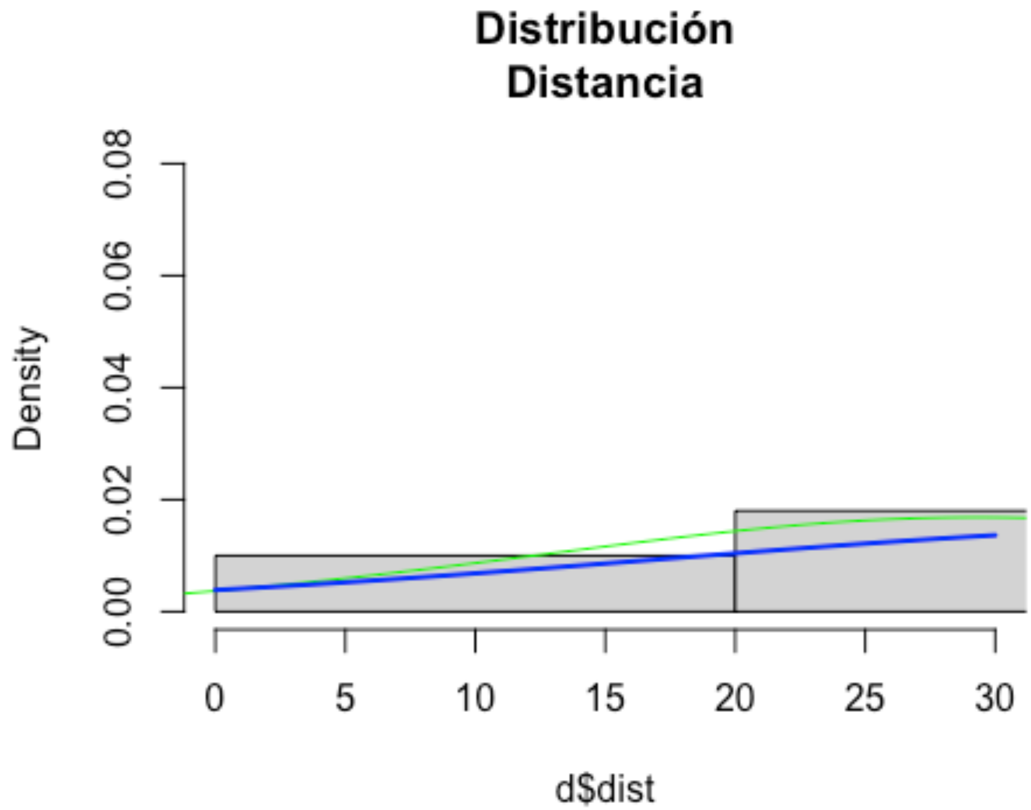

```
qqnorm(d$dist)
qqline(d$dist)
```




```
#Realiza el histograma y su distribución teórica de probabilidad (sugerencia, adapta el código:

hist(d$speed,freq=FALSE, xlim=c(0,30), ylim=c(0,0.08),main="Distribución Velocidad")
lines(density(d$speed),col="green")
curve(dnorm(x,mean=mean(d$speed),sd=sd(d$speed)), from=0, to=30, add=TRUE, col="blue",lwd=2)
```




```
hist(d$dist,freq=FALSE, xlim=c(0,30), ylim=c(0,0.08),main="Distribución
Distancia")
lines(density(d$dist),col="green")
curve(dnorm(x,mean=mean(d$dist),sd=sd(d$dist)), from=0, to=30,
add=TRUE, col="blue",lwd=2)
```




```
#Calcula el coeficiente de sesgo y el coeficiente de curtosis (sugerencia:
usar la librería e1071, usar: skeness y kurtosis) para cada variable.

library(e1071)
print("El sesgo de velocidad es :")

## [1] "El sesgo de velocidad es :"

vs = skewness(d$speed)
print(vs)

## [1] -0.1105533

print("La curtosis de velocidad es:")

## [1] "La curtosis de velocidad es:"

vc = kurtosis(d$speed)
print(vc)

## [1] -0.6730924

print("El sesgo de distancia es :")

## [1] "El sesgo de distancia es :"
```

```
ds = skewness(d$dist)
print(ds)

## [1] 0.7591268

print("La curtosis de distancia es:")

## [1] "La curtosis de distancia es:"

dc = kurtosis(d$dist)
print(dc)

## [1] 0.1193971

print(" Como conlusion se puede ver que kla variable de velocidad presenta
mucha mejor normalidad de los datos y mejores resultados dentro de las
pruebas, en cambio la variable distancia si tiene una inclinacion dentro de
sus datos tieniendo sesgo a la derecha.")

## [1] " Como conlusion se puede ver que kla variable de velocidad presenta
mucha mejor normalidad de los datos y mejores resultados dentro de las
pruebas, en cambio la variable distancia si tiene una inclinacion dentro de
sus datos tieniendo sesgo a la derecha."
```

Parte 2: Regresión lineal

```
#Prueba regresión lineal simple entre distancia y velocidad. Usa lm(y~x).
#Escribe el modelo lineal obtenido.
x = d$speed
y = d$dist

modelo = lm(y~x)

#Analiza significancia del modelo: individual, conjunta y coeficiente de
determinación. Usa summary(Modelo)

summary(modelo)

##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -29.069  -9.525  -2.272   9.215  43.201
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -17.5791     6.7584  -2.601   0.0123 *
## x             3.9324     0.4155   9.464 1.49e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

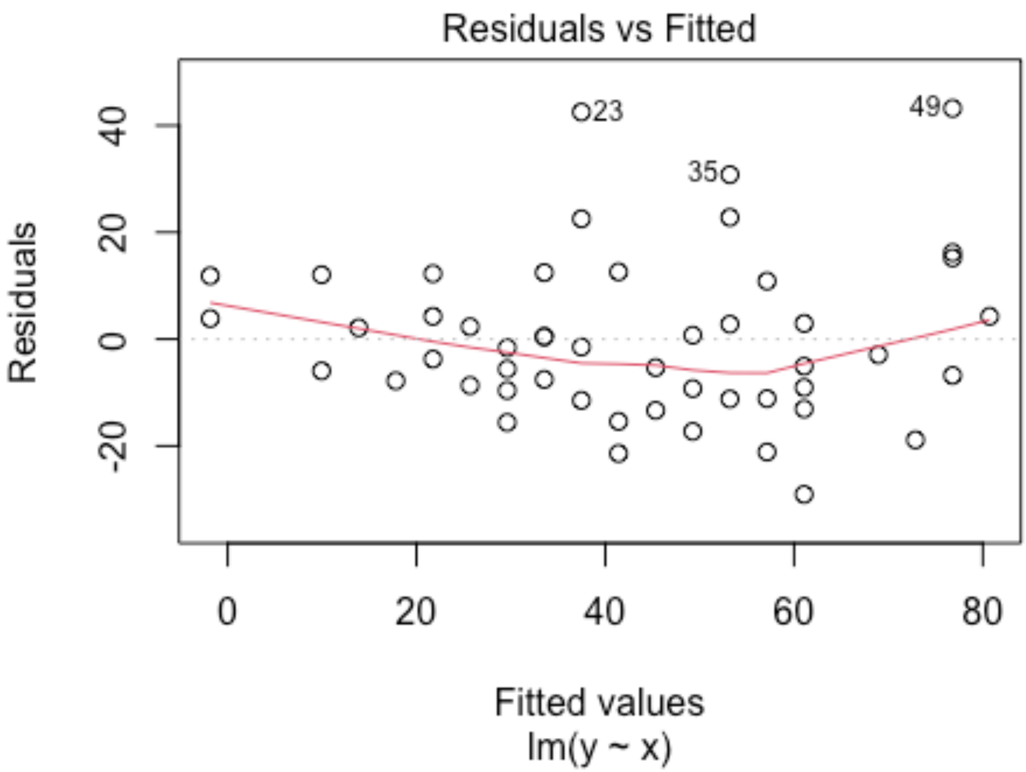
```
##
## Residual standard error: 15.38 on 48 degrees of freedom
## Multiple R-squared:  0.6511, Adjusted R-squared:  0.6438
## F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12

#Analiza validez del modelo.
#Residuos con media cero
#Normalidad de los residuos
#Homocedasticidad, independencia y linealidad.
library(lmtest)

## Loading required package: zoo

##
## Attaching package: 'zoo'

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

bptest(modelo)

##
##  studentized Breusch-Pagan test
##
## data:  modelo
## BP = 3.2149, df = 1, p-value = 0.07297

gqtest(modelo)

##
##  Goldfeld-Quandt test
##
## data:  modelo
## GQ = 1.5512, df1 = 23, df2 = 23, p-value = 0.1498
## alternative hypothesis: variance increases from segment 1 to 2

dwtest(modelo)

##
##  Durbin-Watson test
##
## data:  modelo
## DW = 1.6762, p-value = 0.09522
## alternative hypothesis: true autocorrelation is greater than 0

bgtest(modelo)

##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
```

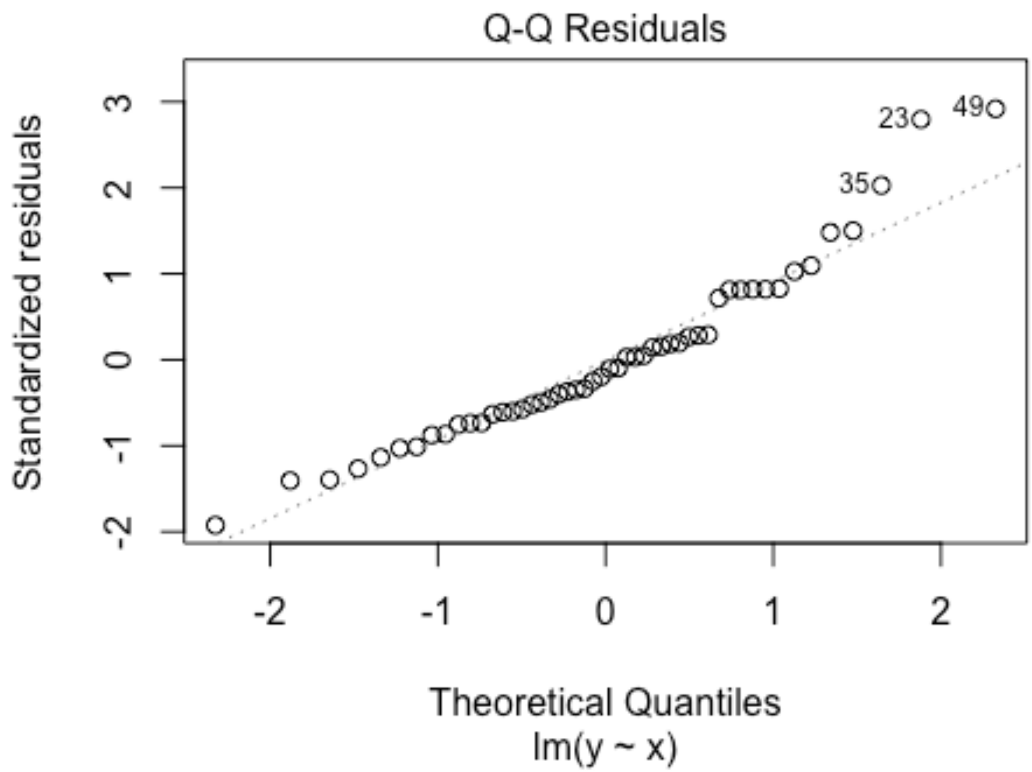
```
## data:  modelo
## LM test = 1.2908, df = 1, p-value = 0.2559
```

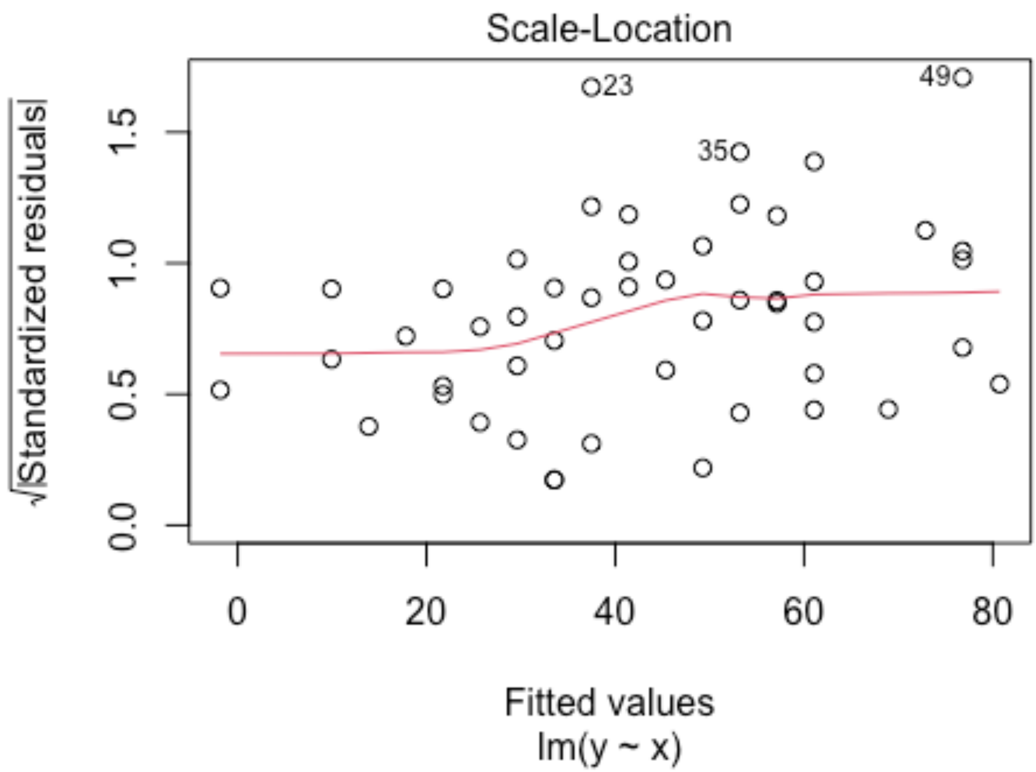
```r
resettest(modelo)
```

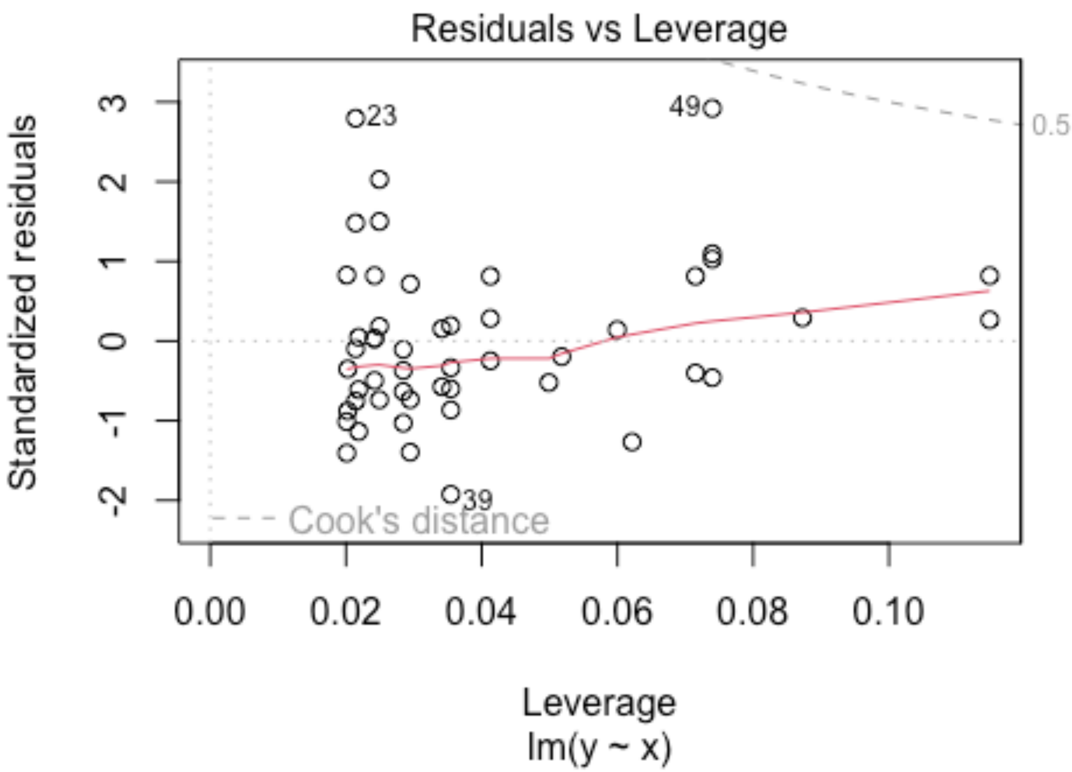
```
##
##  RESET test
##
## data:  modelo
## RESET = 1.5554, df1 = 2, df2 = 46, p-value = 0.222
```

```r
#Usa plot(Modelo) para los gráficos y añade pruebas de hipótesis.
plot(modelo)
```

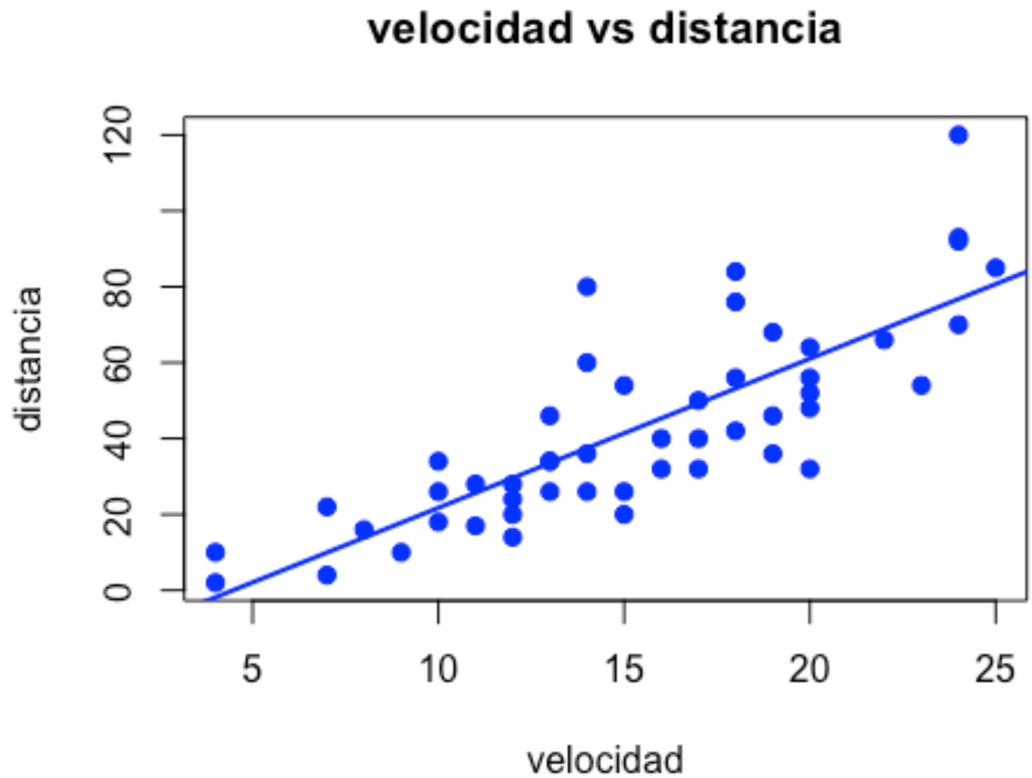

Residuals vs Fitted
Residuals
40
20
0
-20
23
35
49
0
20
40
60
80
Fitted values
lm(y ~ x)


Q-Q Residuals
Standardized residuals
Theoretical Quantiles
lm(y ~ x)
23
49
35
3
2
1
0
-1
-2
-2
-1
0
1
2


Scale-Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
0
20
40
60
80
23
49
35
Fitted values
lm(y ~ x)



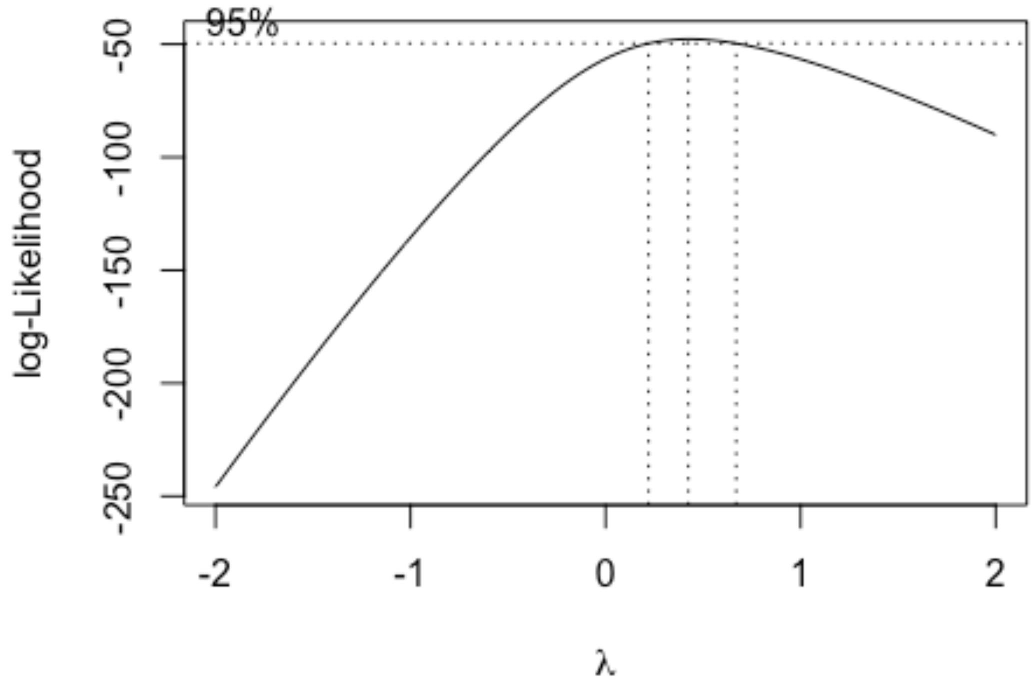

```
#Grafica los datos y el modelo de la distancia en función de la velocidad.
plot(x,y, col="blue", main="velocidad vs distancia", ylab="distancia", xlab
= "velocidad", pch=19)
abline(lm(y~x), col="blue", lwd=2)
```




```
#Comenta sobre la idoneidad del modelo en función de su significancia y
validez.

print("Nuestro modelo cuenta con datos atipicos, lo cual no es conveniente,
tiene buenos resultados en si.")

## [1] "Nuestro modelo cuenta con datos atipicos, lo cual no es conveniente,
tiene buenos resultados en si."
```

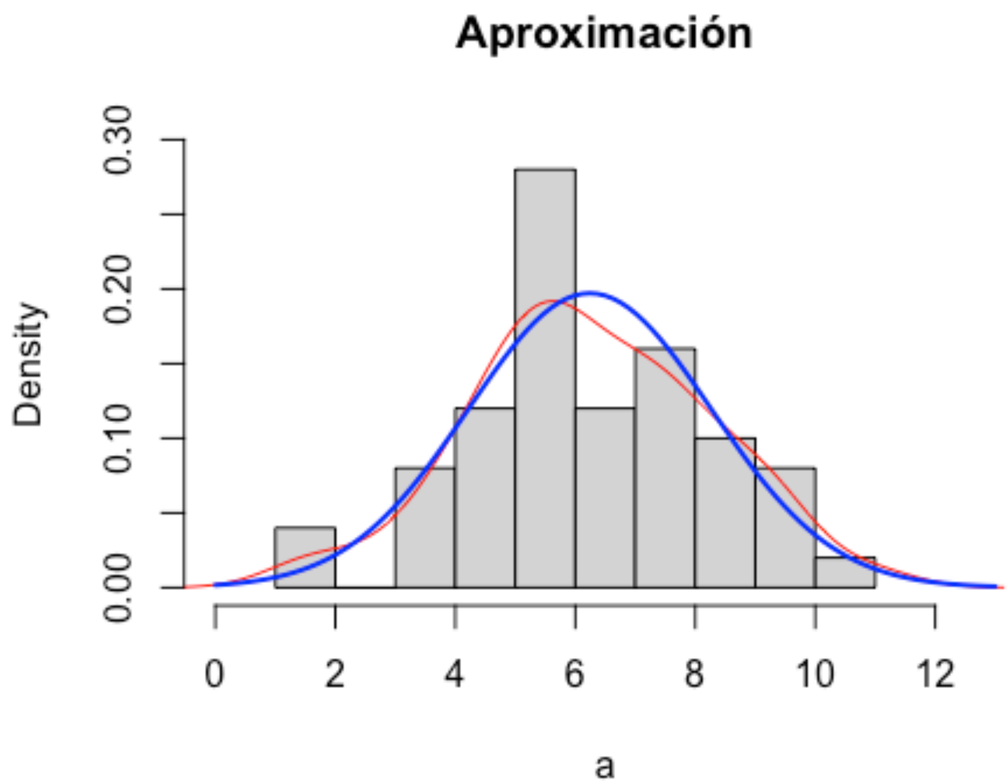
Parte 3: Regresión no lineal

```
#Encuentra el valor de  en la transformación Box-Cox para el modelo lineal:
donde Y sea la distancia y X la velocidad. Aprovecha que el comando de boxcox
en R te da la oportunidad de trabajar con el modelo líneal:



#Utiliza: boxcox(lm(Distancia~Velocidad)) si la variable con más alejamiento
de normalidad es la distancia

library(MASS)
bx<-boxcox(lm(y~x))
```

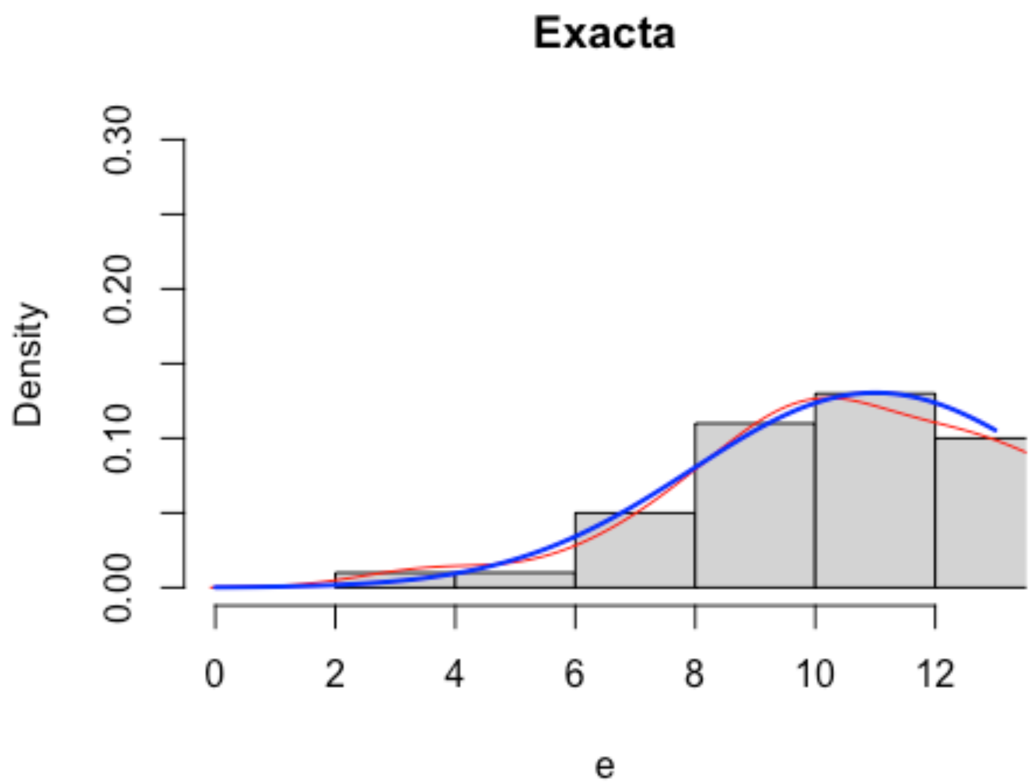



```
l=bx$x[which.max(bx$y)]
print(paste("lambda: ", l))

## [1] "lambda:  0.424242424242424"

a = sqrt(y)
e = (((y)^0.42))/0.42

#Analiza la normalidad de las transformaciones obtenidas. Utiliza como
argumento de normalidad:

library(e1071)
print("El sesgo del modelo exacto es: ")

## [1] "El sesgo del modelo exacto es: "

se = skewness(e)
print(se)

## [1] -0.1790006

print("La curtosis del modelo exacto es: ")

## [1] "La curtosis del modelo exacto es: "
```

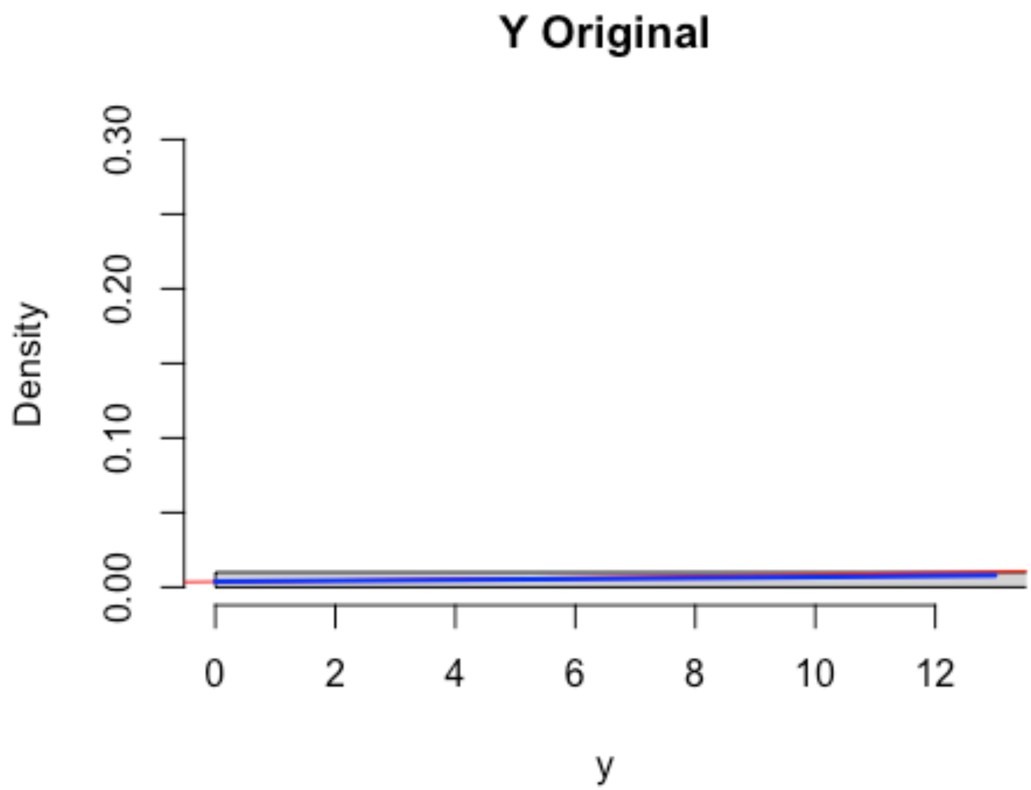
```r
ce = kurtosis(e)
print(ce)

## [1] -0.1774661

print("El sesgo del modelo aproximado es: ")

## [1] "El sesgo del modelo aproximado es: "

sa = skewness(a)
print(sa)

## [1] -0.01902765

print("La curtosis del modelo aproximado es: ")

## [1] "La curtosis del modelo aproximado es: "

ca = kurtosis(a)
print(ca)

## [1] -0.3144682

print("El sesgo de y es: ")

## [1] "El sesgo de y es: "

sy = skewness(y)
print(sy)

## [1] 0.7591268

print("La curtosis de y es: ")

## [1] "La curtosis de y es: "

cy = kurtosis(y)
print(cy)

## [1] 0.1193971

#Obten el histograma de los 2 modelos obtenidos (exacto y aproximado) y los
datos originales.
hist(a,freq=FALSE, xlim=c(0,13), ylim=c(0,0.30),main="Aproximación")
lines(density(a),col="red")
curve(dnorm(x,mean=mean(a),sd=sd(a)), from=0, to=13, add=TRUE,
col="blue",lwd=2)
```



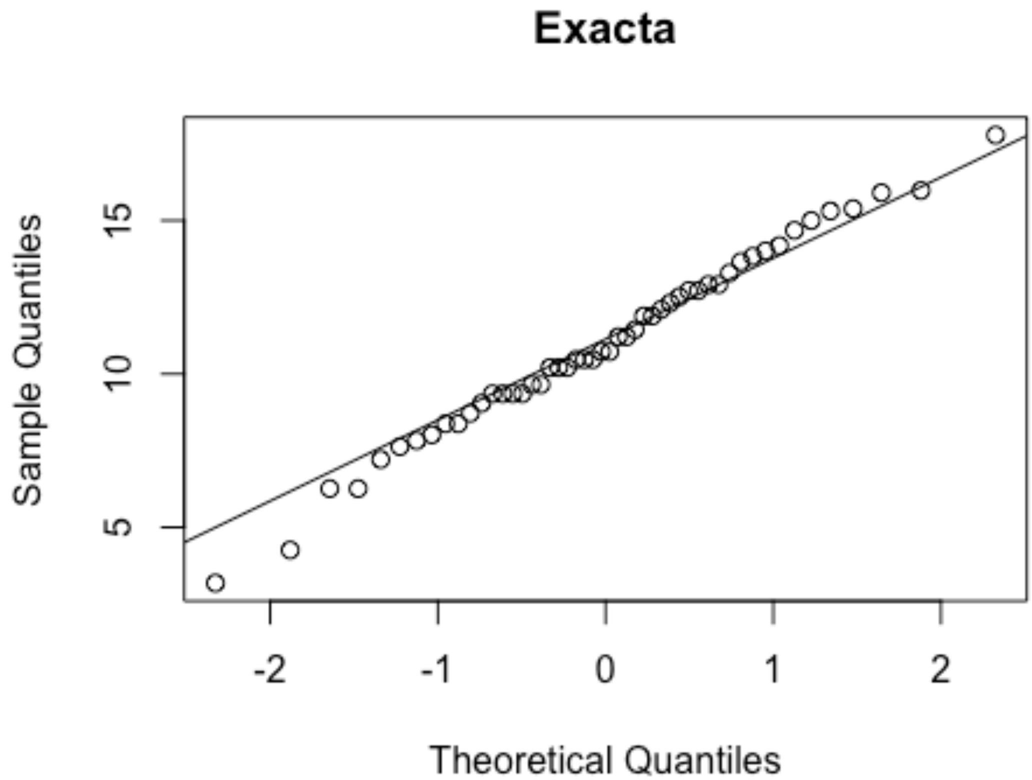

```
hist(e,freq=FALSE, xlim=c(0,13), ylim=c(0,0.30),main="Exacta")
lines(density(e),col="red")
curve(dnorm(x,mean=mean(e),sd=sd(e)), from=0, to=13, add=TRUE,
col="blue",lwd=2)
```



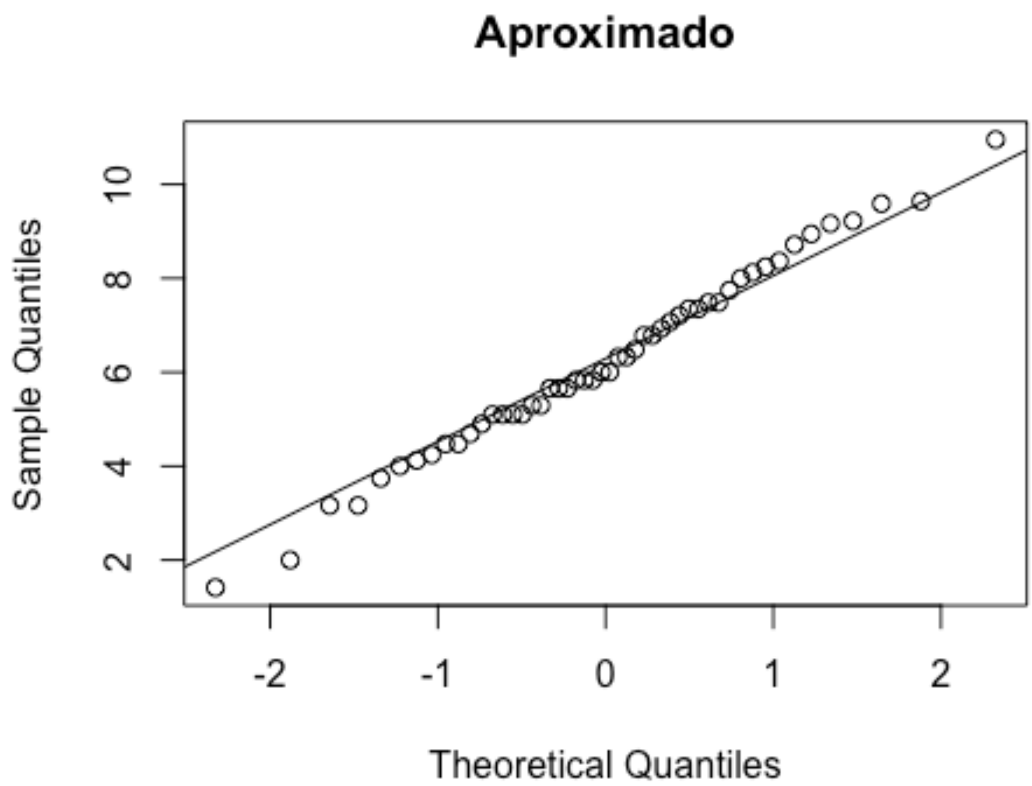

```
hist(y,freq=FALSE, xlim=c(0,13), ylim=c(0,0.30),main="Y Original")
lines(density(y),col="red")
curve(dnorm(x,mean=mean(y),sd=sd(y)), from=0, to=13, add=TRUE,
col="blue",lwd=2)
```




```
#Realiza algunas pruebas de normalidad para los datos transformados.

qqnorm(e, main="Exacta")
qqline(e)
```

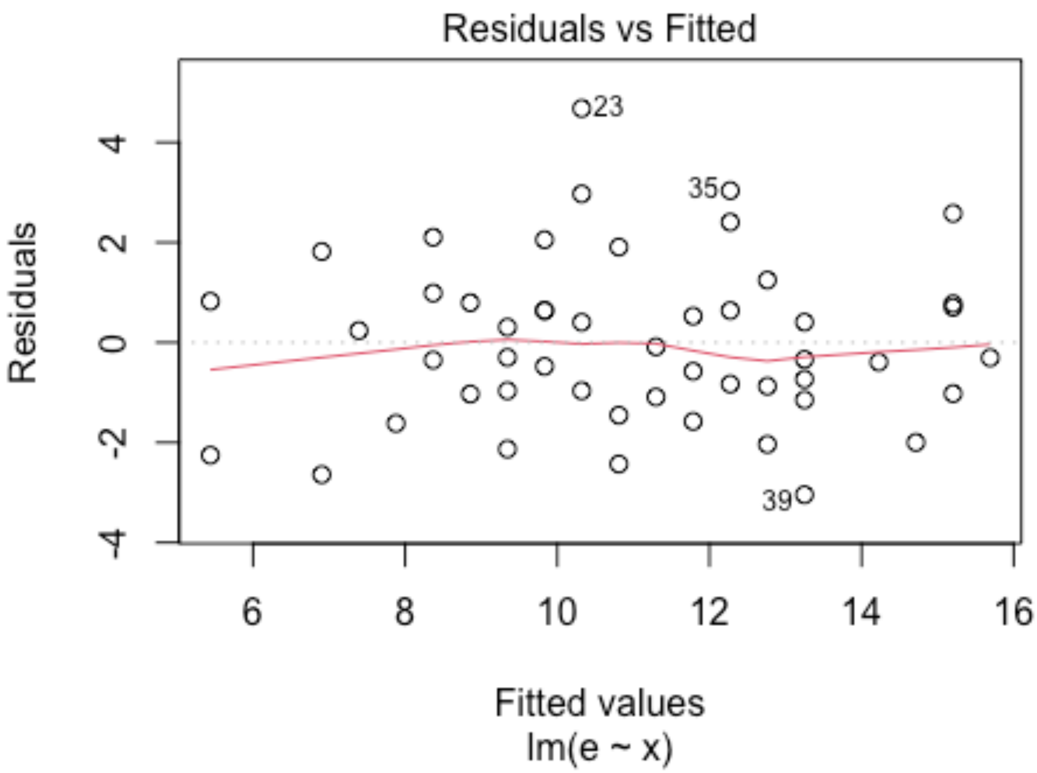



```r
qqnorm(a, main="Aproximado")
qqline(a)
```

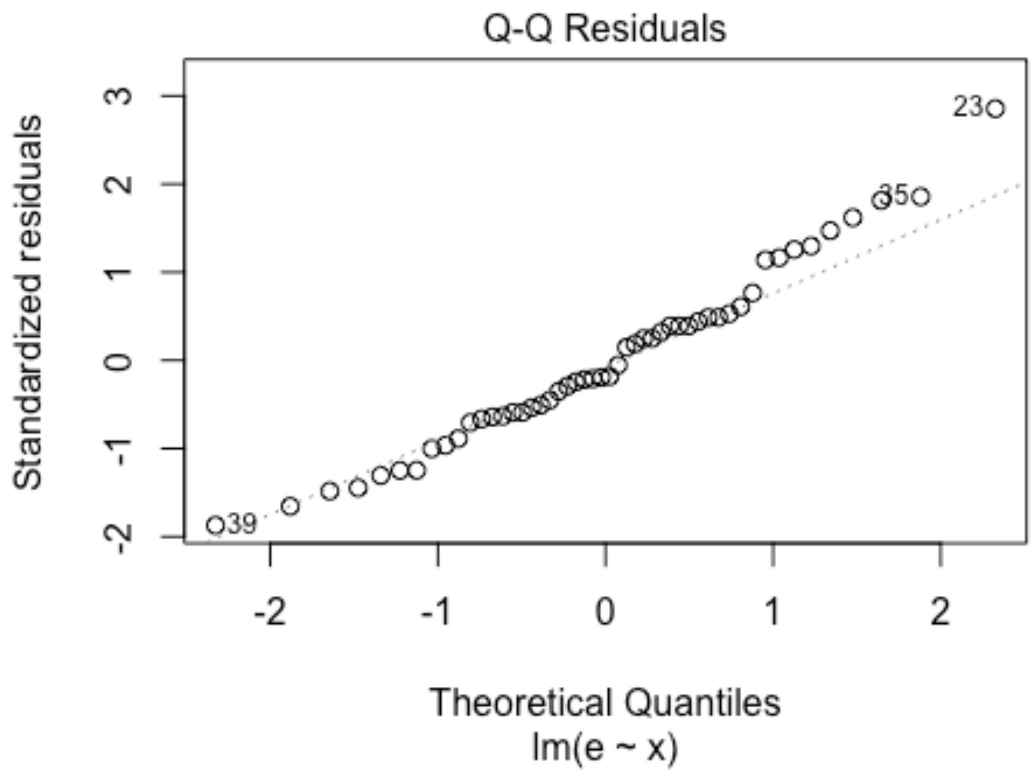


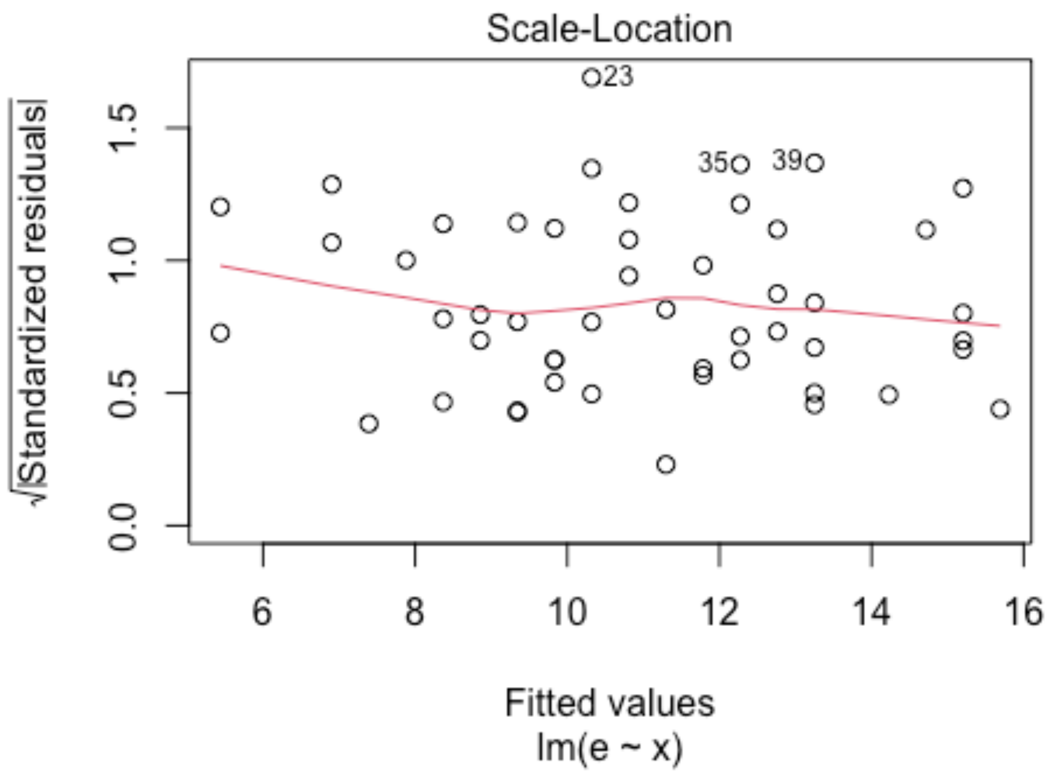

```
lillie.test(y)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  y
## D = 0.12675, p-value = 0.04335

shapiro.test(y)

##
##  Shapiro-Wilk normality test
##
## data:  y
## W = 0.95144, p-value = 0.0391

lillie.test(a)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  a
## D = 0.067624, p-value = 0.8218
```

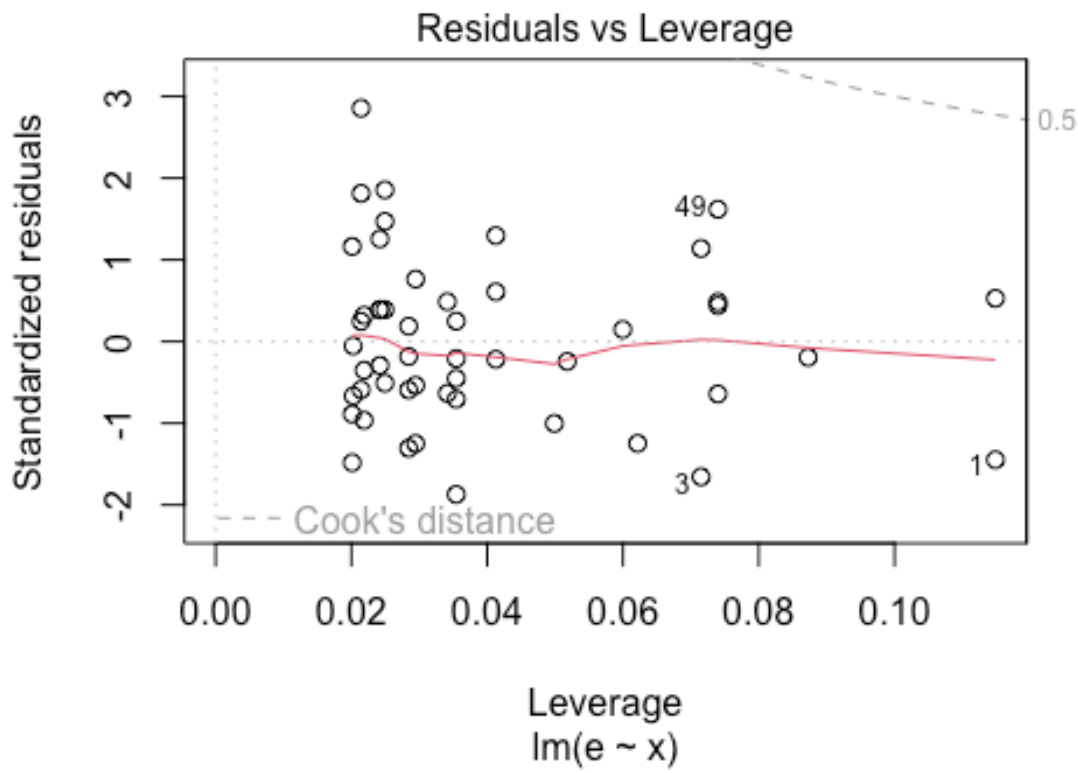
```r
shapiro.test(a)

##
##  Shapiro-Wilk normality test
##
## data:  a
## W = 0.99347, p-value = 0.9941

lillie.test(e)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  e
## D = 0.056406, p-value = 0.9567

shapiro.test(e)

##
##  Shapiro-Wilk normality test
##
## data:  e
## W = 0.99148, p-value = 0.9745

#Concluye sobre las dos transformaciones realizadas: Define la mejor
transformación de los datos de acuerdo a las características de las dos
transformaciones encontradas (exacta o aproximada). Toman en cuenta la
normalidad de los datos y la economía del modelo.

print("El modelo exacto demuestra mejores resultados y una distribucion de
datos mas normal, por lo cual nos deberiamos inclinar a este.")

## [1] "El modelo exacto demuestra mejores resultados y una distribucion de
datos mas normal, por lo cual nos deberiamos inclinar a este."

#Con la mejor transformación (punto 2), realiza la regresión lineal simple
entre la mejor transformación (exacta o aproximada) y la variable velocidad:

#Escribe el modelo lineal para la transformación.
modelon = lm(e~x)

#Grafica los datos y el modelo lineal (ecuación) de la transformación elegida
vs velocidad.
plot(modelon)
```



Residuals vs Fitted
Residuals
Fitted values
lm(e ~ x)
23
35
39
4
2
0
-2
-4
6
8
10
12
14
16


Q-Q Residuals
Standardized residuals
Theoretical Quantiles
lm(e ~ x)
23
35
39
3
2
1
0
-1
-2
-2
-1
0
1
2


Scale-Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
6
8
10
12
14
16
23
35
39
Fitted values
lm(e ~ x)




```
#Analiza validez del modelo: normalidad de los residuos, homocedasticidad e
independencia. Indica si hay candidatos a datos atípicos o influyentes en la
regresión. Usa plot(Modelo) para los gráficos y añade pruebas de hipótesis.
library(lmtest)
bptest(modelon)

##
##  studentized Breusch-Pagan test
##
## data:  modelon
## BP = 0.16186, df = 1, p-value = 0.6874

gqtest(modelon)

##
##  Goldfeld-Quandt test
##
## data:  modelon
## GQ = 0.73256, df1 = 23, df2 = 23, p-value = 0.7694
## alternative hypothesis: variance increases from segment 1 to 2

dwtest(modelon)
```

```
##
##  Durbin-Watson test
##
## data:  modelon
## DW = 1.9613, p-value = 0.3874
## alternative hypothesis: true autocorrelation is greater than 0
```

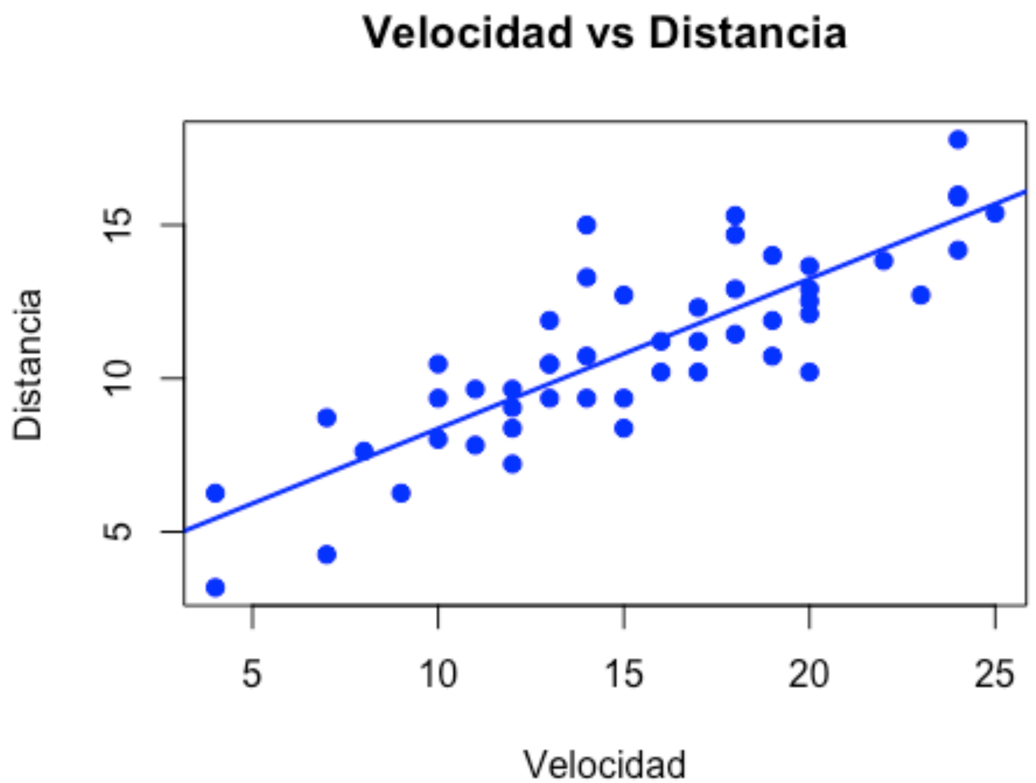
```r
bgtest(modelon)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  modelon
## LM test = 5.2401e-06, df = 1, p-value = 0.9982
```

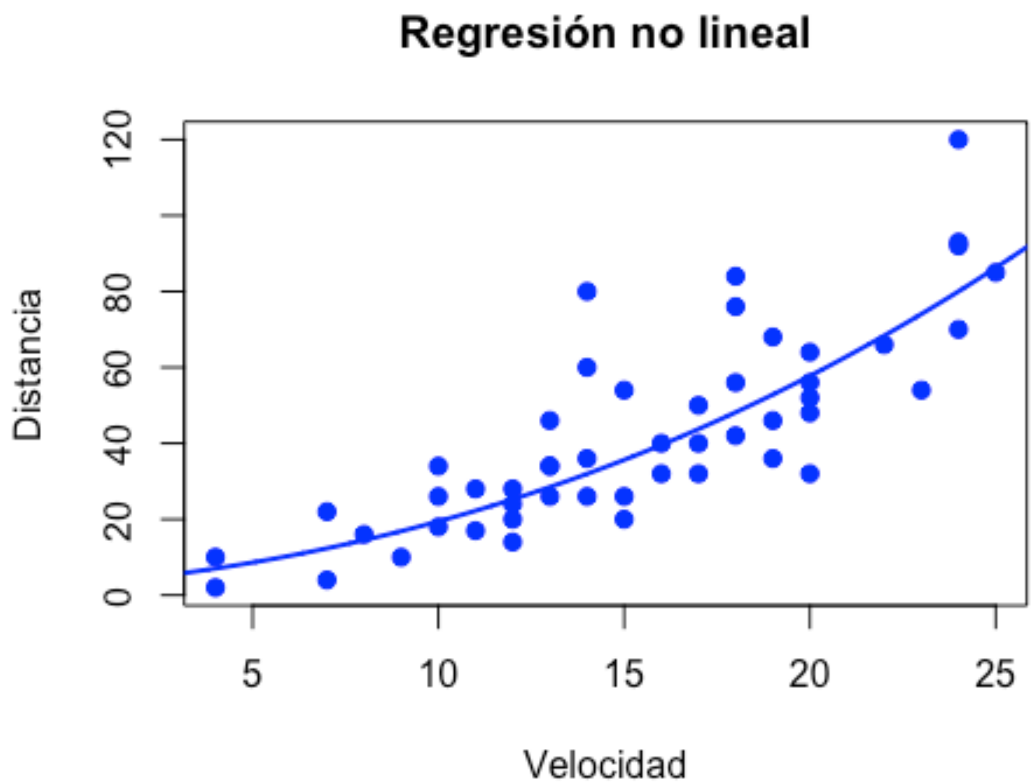
```r
resettest(modelon)
```

```
##
##  RESET test
##
## data:  modelon
## RESET = 0.70165, df1 = 2, df2 = 46, p-value = 0.501
```

```r
plot(x,e, col="blue", main="Velocidad vs Distancia", ylab="Distancia",
xlab = "Velocidad", pch=19)
abline(lm(e~x), col="blue", lwd=2)
```




```r
Y = function(x){(0.2016*x+1.4616)^(50/21)}
plot(x, y, col = "blue", pch=19, xlab = "Velocidad", ylab = "Distancia",
main="Regresión no lineal")
x = seq(0,26,0.01)
lines(x, Y(x), col="blue",lwd=2)
```



Parte 4: Conclusión

```
#Define cuál de los dos modelos analizados (Punto 1 o Punto 2) es el mejor
modelo para describir la relación entre la distancia y la velocidad.
#Comenta sobre posibles problemas del modelo elegido (datos atípicos,
alejamiento de los supuestos, dificultad de cálculo o interpretación)

print("El mejor modelo es el no lineal, ya que presenta una mejor
distribucion de los datos y tambien nos brina menos datos atipicos, en
realidad no presenta tantos problemas, mas que puede llegar a ser dificil de
calcular si no se logra un modelo optimo.")

## [1] "El mejor modelo es el no lineal, ya que presenta una mejor
distribucion de los datos y tambien nos brina menos datos atipicos, en
realidad no presenta tantos problemas, mas que puede llegar a ser dificil de
calcular si no se logra un modelo optimo."
```