Actividad 13: Regresión No Lineal

Daniela Jiménez Téllez

2024-09-11

Importación de librerías

```
library(nortest)
library(e1071)
library(lmtest)
```

```
## Cargando paquete requerido: zoo
```

```
##
## Adjuntando el paquete: 'zoo'
```

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

```
library(MASS)
```

El objetivo es encontrar el mejor modelo que relacione la velocidad de los automóviles y las distancias necesarias para detenerse en autos de modelos existentes en 1920 (base de datos car). La ecuación encontrada no sólo deberá ser el mejor modelo obtenido sino también deberá ser el más económico en terminos de la complejidad del modelo.

Importación de datos

```
datos <- cars

velocidad <- datos$speed
distancia <- datos$dist
```

Parte 1: Análisis de Normalidad

1. Accede a los datos de cars en R (data = cars)

- Prueba normalidad univariada de la velocidad y distancia (prueba con dos de las pruebas vistas en clase).

```
# Test de Shapiro - Wilk

shap_vel <- shapiro.test(velocidad)
shap_dist <- shapiro.test(distancia)

cat("Resultados del test de Shapiro-Wilk para velocidad: \n\n")
```

```
## Resultados del test de Shapiro-Wilk para velocidad:
```

```
print(shap_vel)
```

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

```
cat("\n\n")
```

```
cat("Resultados del test de Shapiro-Wilk para distancia: \n\n")
```

```
## Resultados del test de Shapiro-Wilk para distancia:
```

```
print(shap_dist)
```

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

```
cat("\n\n")
```

```
# Test de Anderson-Darling

ad_vel <- ad.test(velocidad)
ad_dist <- ad.test(distancia)

cat("Resultados del test de Anderson-Darling para velocidad: \n\n")
```

```
## Resultados del test de Anderson-Darling para velocidad:
```

```
print(ad_vel)
```

```
##
##  Anderson-Darling normality test
##
## data:  velocidad
## A = 0.26143, p-value = 0.6927
```

```
cat("\n\n")
```

```
cat("Resultados del test de Anderson-Darling para distancia: \n\n")
```

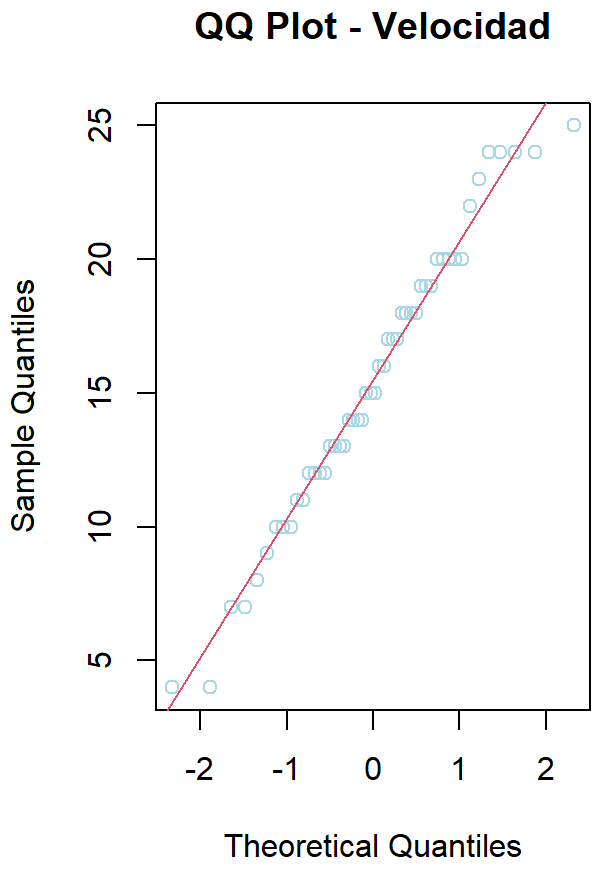
```
## Resultados del test de Anderson-Darling para distancia:
```

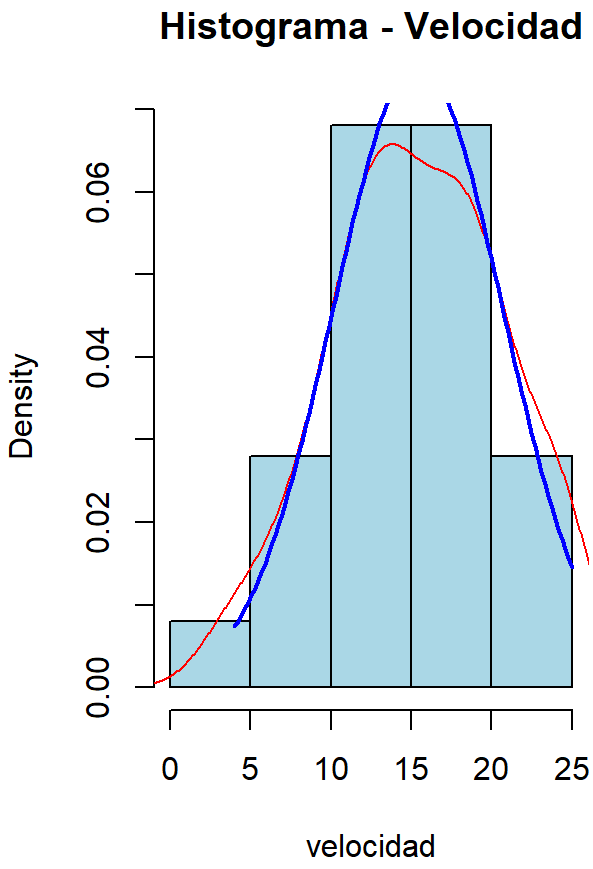
```
print(ad_dist)
```

```
##
##  Anderson-Darling normality test
##
## data:  distancia
## A = 0.74067, p-value = 0.05021
```

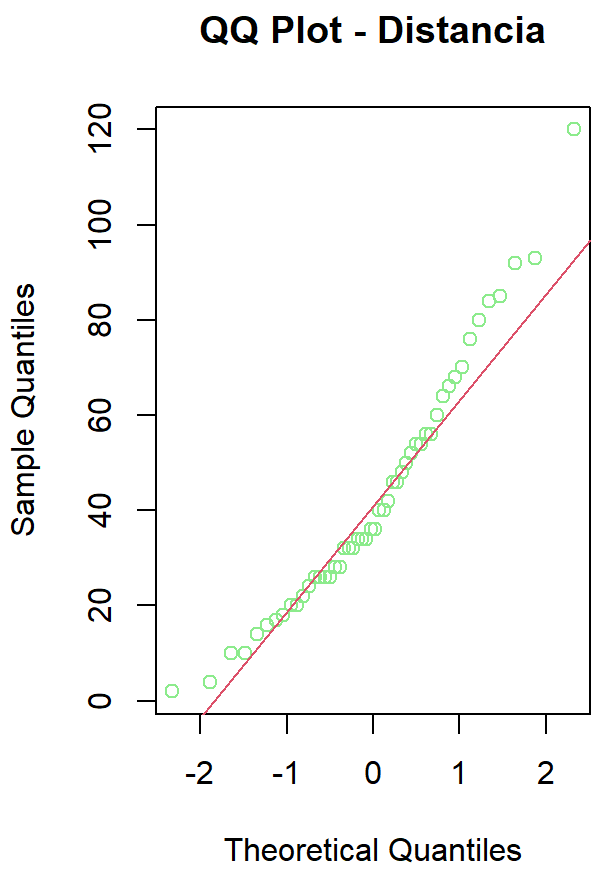
- Realiza gráficos que te ayuden a identificar posibles alejamientos de normalidad:
    - Los datos y su respectivo QQPlot: qqnorm(datos) y qqline(datos) para cada variable
    - Realiza el histograma y su distribución teórica de probabilidad (sugerencia, adapta el código:
        - hist(datos,freq=FALSE) lines(density(datos),col="red") curve(dnorm(x,mean=mean(datos),sd=sd(datos)), from=min(datos), to=max(datos), add=TRUE, col="blue",lwd=2)
    - Se te sugiere usar par(mfrow=c(1,2)) para graficar el QQ plot y el histograma de una variable en un mismo espacio.

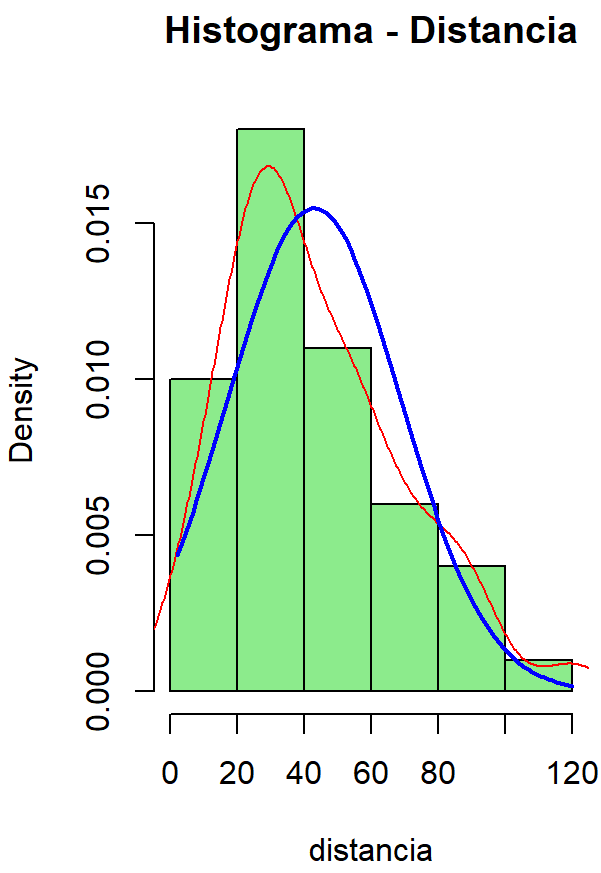
```
# VELOCIDAD

par(mfrow = c(1, 2))

# QQPlot

qqnorm(velocidad, main = "QQ Plot - Velocidad", col = "lightblue")
qqline(velocidad, col = 2)

# Histograma
hist(velocidad, freq = FALSE, main = "Histograma - Velocidad", col = "lightblue")
lines(density(velocidad), col = "red")
curve(dnorm(x, mean = mean(velocidad), sd = sd(velocidad)), from = min(velocidad), to = max(velo
cidad), add = TRUE, col = "blue", lwd = 2)
```






```
# DISTANCIA

par(mfrow=c(1,2))

# QQ Plot
qqnorm(distancia, main = "QQ Plot - Distancia", col = "lightgreen")
qqline(distancia, col = 2)

# Histograma
hist(distancia, freq = FALSE, main = "Histograma - Distancia", col = "lightgreen")
lines(density(distancia), col = "red")
curve(dnorm(x, mean = mean(distancia), sd = sd(distancia)), from = min(distancia), to = max(dist
ancia), add = TRUE, col = "blue", lwd = 2)
```





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

```
# VELOCIDAD

sesgo_velocidad <- skewness(velocidad)
curtosis_velocidad <- kurtosis(velocidad)

cat("Coeficiente de sesgo para velocidad:", sesgo_velocidad, "\n")
```

```
## Coeficiente de sesgo para velocidad: -0.1105533
```

```
cat("Coeficiente de curtosis para velocidad:", curtosis_velocidad, "\n\n")
```

```
## Coeficiente de curtosis para velocidad: -0.6730924
```

```
# DISTANCIA

sesgo_distancia <- skewness(distancia)
curtosis_distancia <- kurtosis(distancia)

cat("Coeficiente de sesgo para distancia:", sesgo_distancia, "\n")
```

```
## Coeficiente de sesgo para distancia: 0.7591268
```

```
cat("Coeficiente de curtosis para distancia:", curtosis_distancia, "\n")
```

```
## Coeficiente de curtosis para distancia: 0.1193971
```

2. Comenta cada gráfico y resultado que hayas obtenido. Emite una conclusión final sobre la normalidad de los datos. Argumenta basándote en todos los análisis realizados en esta parte. Incluye posibles motivos de alejamiento de normalidad.

Con los resultados anteriores se puede decir que la normalidad de los datos es muy diferente hablando de la distancia y la velocidad. En el caso de la velocidad, tanto en la gráfica como en los tests podemos observar que los p-valor son significativos y no se rechaza la hipótesis nula, lo que nos dice que los datos sí siguen una distribución normal. Por otro lado, en el caso de la distancia podemos observar que en el test de Shapiro-Wilk se rechaza la hipótesis nula ya que hay un p-valor menor a 0.05, y en el caso del test de Anderson-Darling, apenas lo supera.

Parte 2: Regresión Lineal

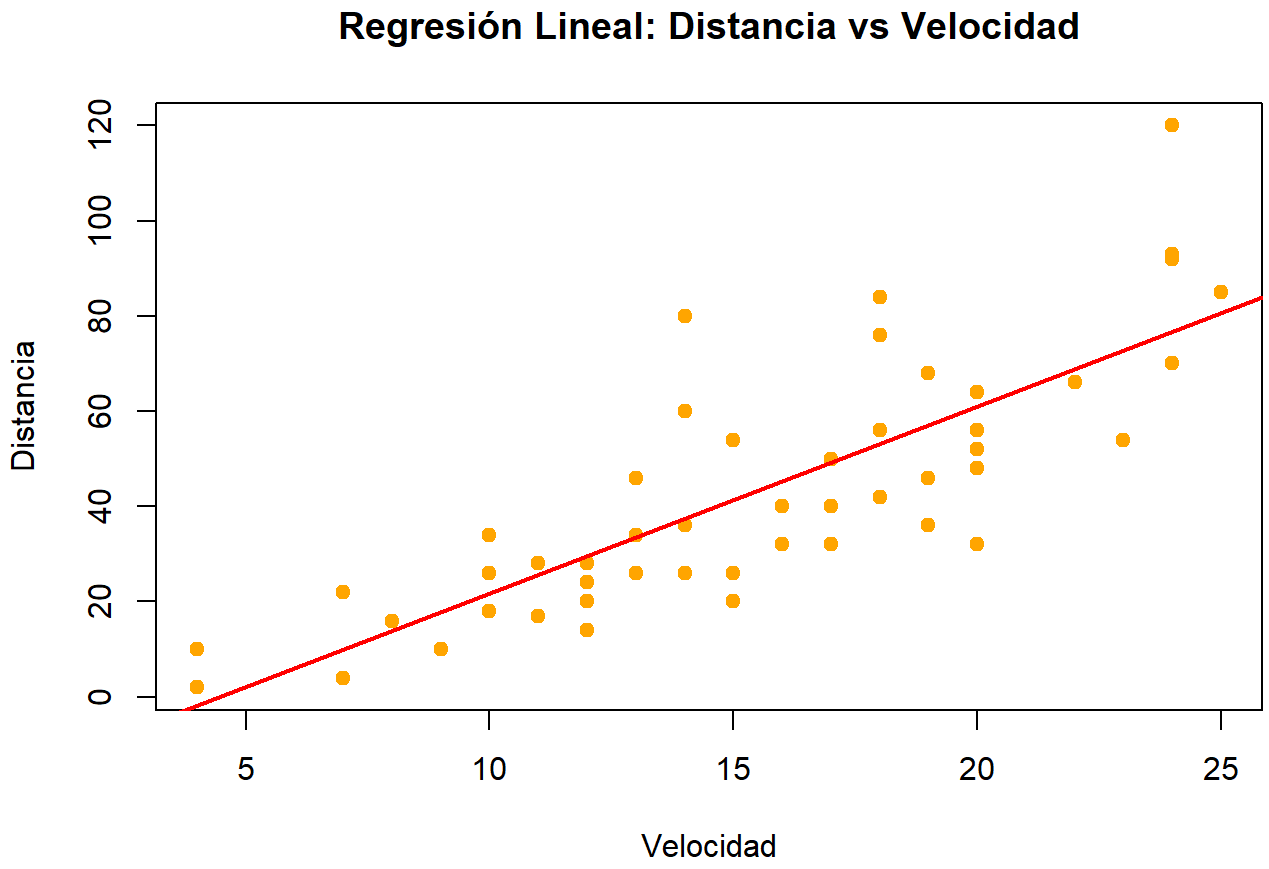
1. Prueba regresión lineal simple entre distancia y velocidad. Usa lm(y~x).

- Escribe el modelo lineal obtenido.
- Grafica los datos y el modelo (ecuación) que obtuviste.

```
# Modelo

modeloRLS <- lm(distancia ~ velocidad)

coeficientes <- coef(modeloRLS)
cat("Modelo: distancia = ", round(coeficientes[1], 2), "+", round(coeficientes[2], 2), "* veloci
dad\n")
```

```
## Modelo: distancia =  -17.58 + 3.93 * velocidad
```

```
# Gráfica

plot(velocidad, distancia, main = "Regresión Lineal: Distancia vs Velocidad",
     xlab = "Velocidad", ylab = "Distancia", pch = 19, col = "orange")

abline(modeloRLS, col = "red", lwd = 2)
```



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

```
resumen_modelo <- summary(modeloRLS)
print(resumen_modelo)
```

```
##
## Call:
## lm(formula = distancia ~ velocidad)
##
## 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 *
## velocidad     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
```

```
cat("\n\n")
```

```
cat("Coeficientes del modelo:\n")
```

```
## Coeficientes del modelo:
```

```
cat("Intercepto:", resumen_modelo$coefficients[1, 1], "\n")
```

```
## Intercepto: -17.57909
```

```
cat("Velocidad:", resumen_modelo$coefficients[2, 1], "\n")
```

```
## Velocidad: 3.932409
```

```
cat("\nValores p para los coeficientes:\n")
```

```
##
## Valores p para los coeficientes:
```

```
cat("Intercepto:", resumen_modelo$coefficients[1, 4], "\n")
```

```
## Intercepto: 0.01231882
```

```
cat("Velocidad:", resumen_modelo$coefficients[2, 4], "\n")
```

```
## Velocidad: 1.489836e-12
```

```
cat("\nValor p del test F para el modelo:", resumen_modelo$fstatistic[1], "\n")
```

```
##
## Valor p del test F para el modelo: 89.56711
```

```
cat("\nCoeficiente de determinación R^2:", resumen_modelo$r.squared, "\n")
```

```
##
## Coeficiente de determinación R^2: 0.6510794
```

3. Analiza validez del modelo.

- Residuos con media cero
- Normalidad de los residuos

- Homocedasticidad, independencia y linealidad.
- Usa plot(Modelo) para los gráficos y añade pruebas de hipótesis.

```
# Media cero

mediaCero <- t.test(resid(modeloRLS))

cat("El resultado del test de media cero para el modelo es: \n\n")
```

```
## El resultado del test de media cero para el modelo es:
```

```
print(mediaCero)
```

```
##
##  One Sample t-test
##
## data:  resid(modeloRLS)
## t = 1.0315e-16, df = 49, p-value = 1
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -4.326  4.326
## sample estimates:
##    mean of x
## 2.220446e-16
```

```
cat("\n\n")
```

```
# Normalidad

norm <- shapiro.test(resid(modeloRLS))
cat("Los resultados de la normalidad para el modelo usando el test de Shapiro-Wilk son: \n\n")
```

```
## Los resultados de la normalidad para el modelo usando el test de Shapiro-Wilk son:
```

```
print(norm)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(modeloRLS)
## W = 0.94509, p-value = 0.02152
```

```
cat("\n\n")
```

```
# Homocedasticidad

hom1 <- bptest(modeloRLS)
hom2 <- gqtest(modeloRLS)

cat("Los resultados de los tests de homocedasticidad para el modelo son: \n\n")
```

```
## Los resultados de los tests de homocedasticidad para el modelo son:
```

```
print(hom1)
```

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

```
print(hom2)
```

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

```
cat("\n\n")
```

```
# Independencia

ind1 <- dwtest(modeloRLS)
ind2 <- bgtest(modeloRLS)

cat("Los resultados de los tests de independencia para el modelo son: \n\n")
```

```
## Los resultados de los tests de independencia para el modelo son:
```

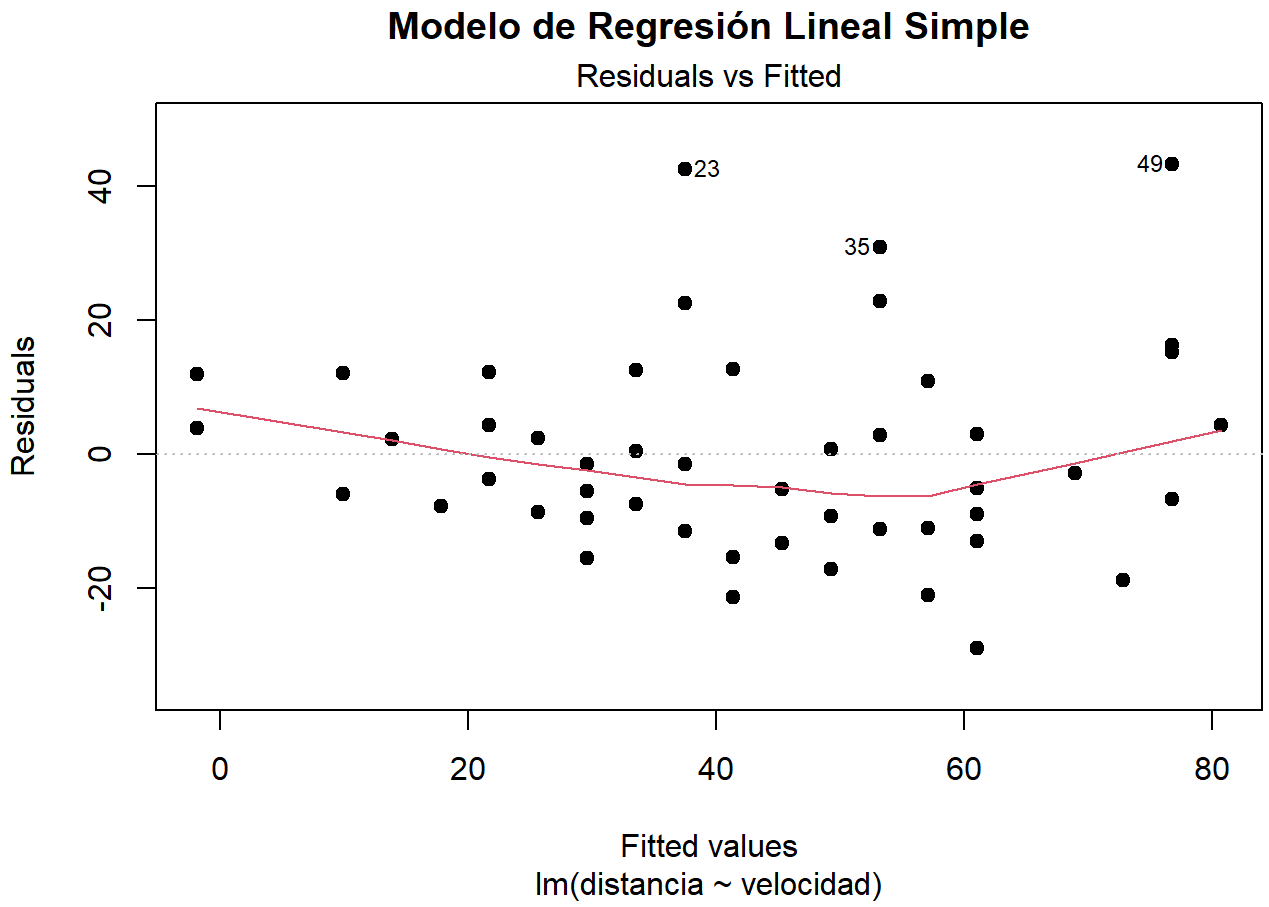
```
print(ind1)
```

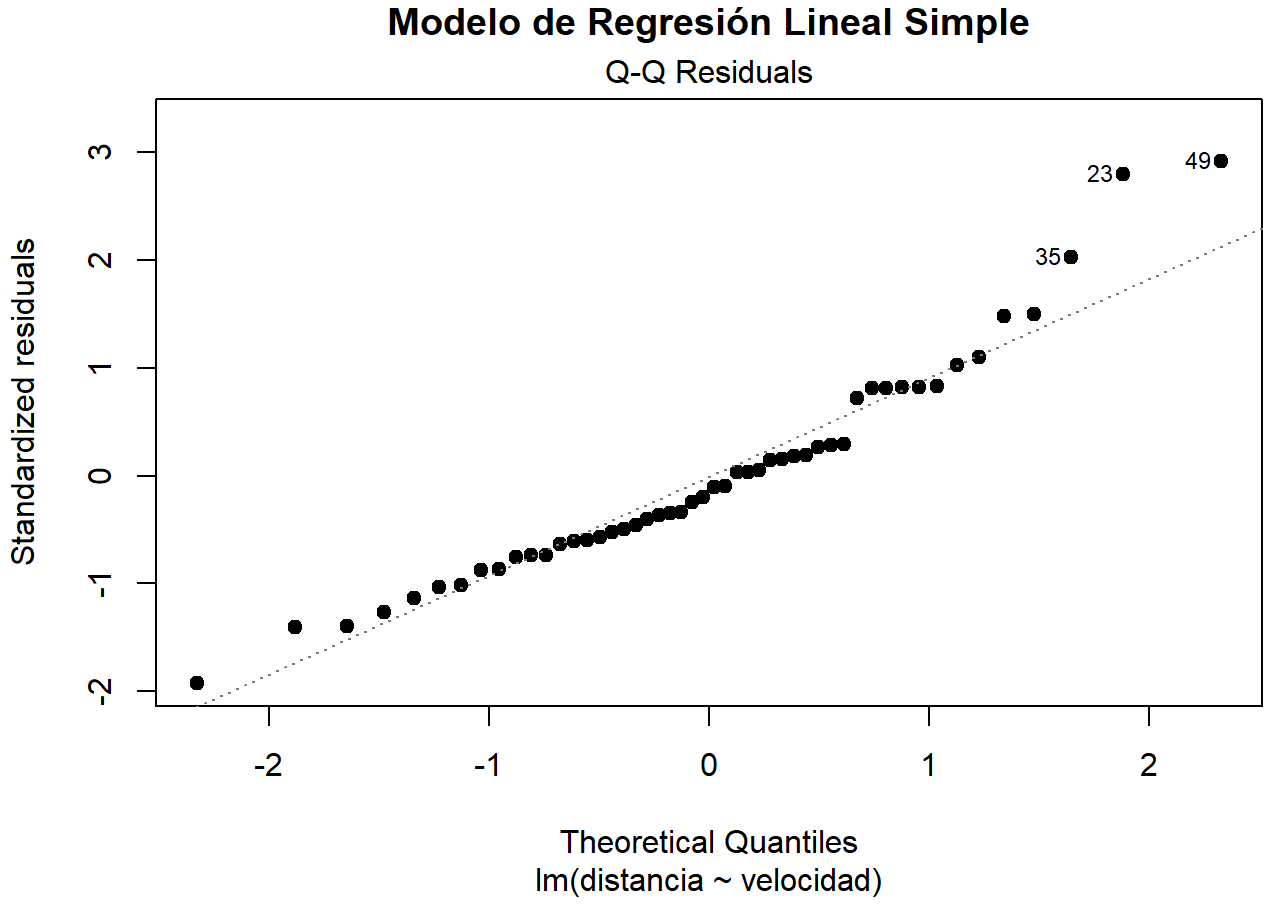
```
##
##  Durbin-Watson test
##
## data:  modeloRLS
## DW = 1.6762, p-value = 0.09522
## alternative hypothesis: true autocorrelation is greater than 0
```

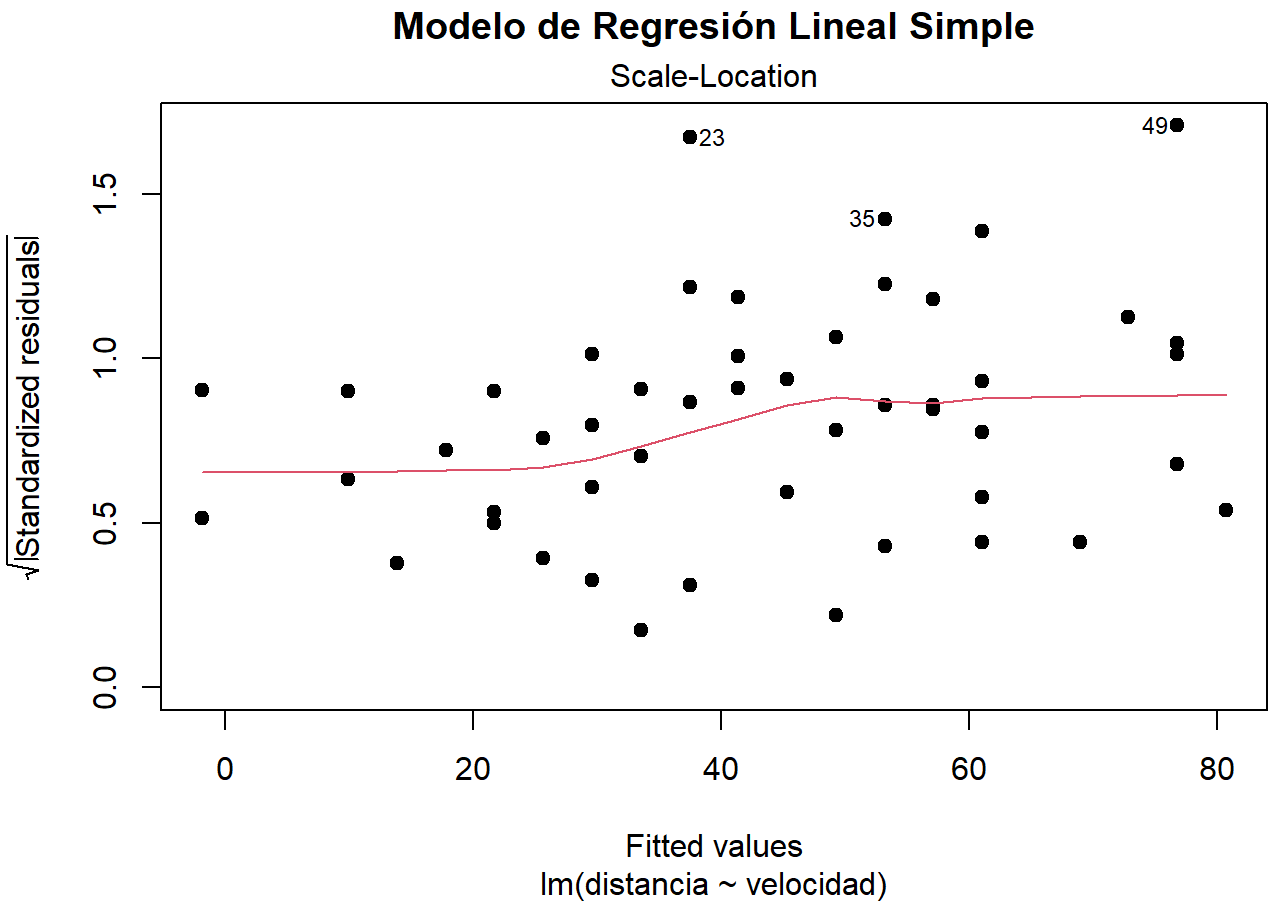
```
print(ind2)
```

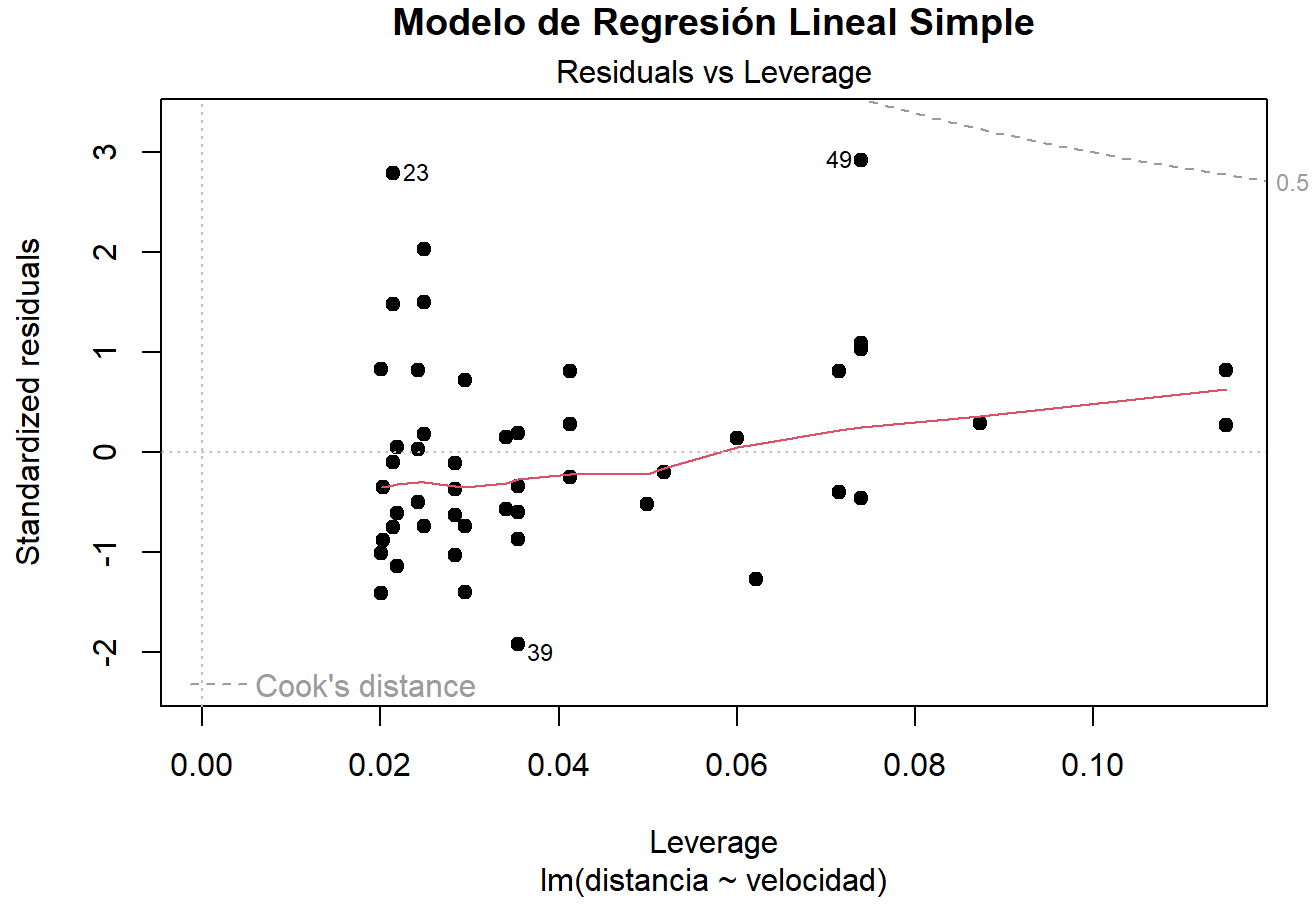
```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  modeloRLS
## LM test = 1.2908, df = 1, p-value = 0.2559
```

```
# Linealidad

# Gráficos

plot(modeloRLS, pch = 19, main = "Modelo de Regresión Lineal Simple")
```


Modelo de Regresión Lineal Simple
Residuals vs Fitted
Residuals
Fitted values
lm(distancia ~ velocidad)
23
35
49

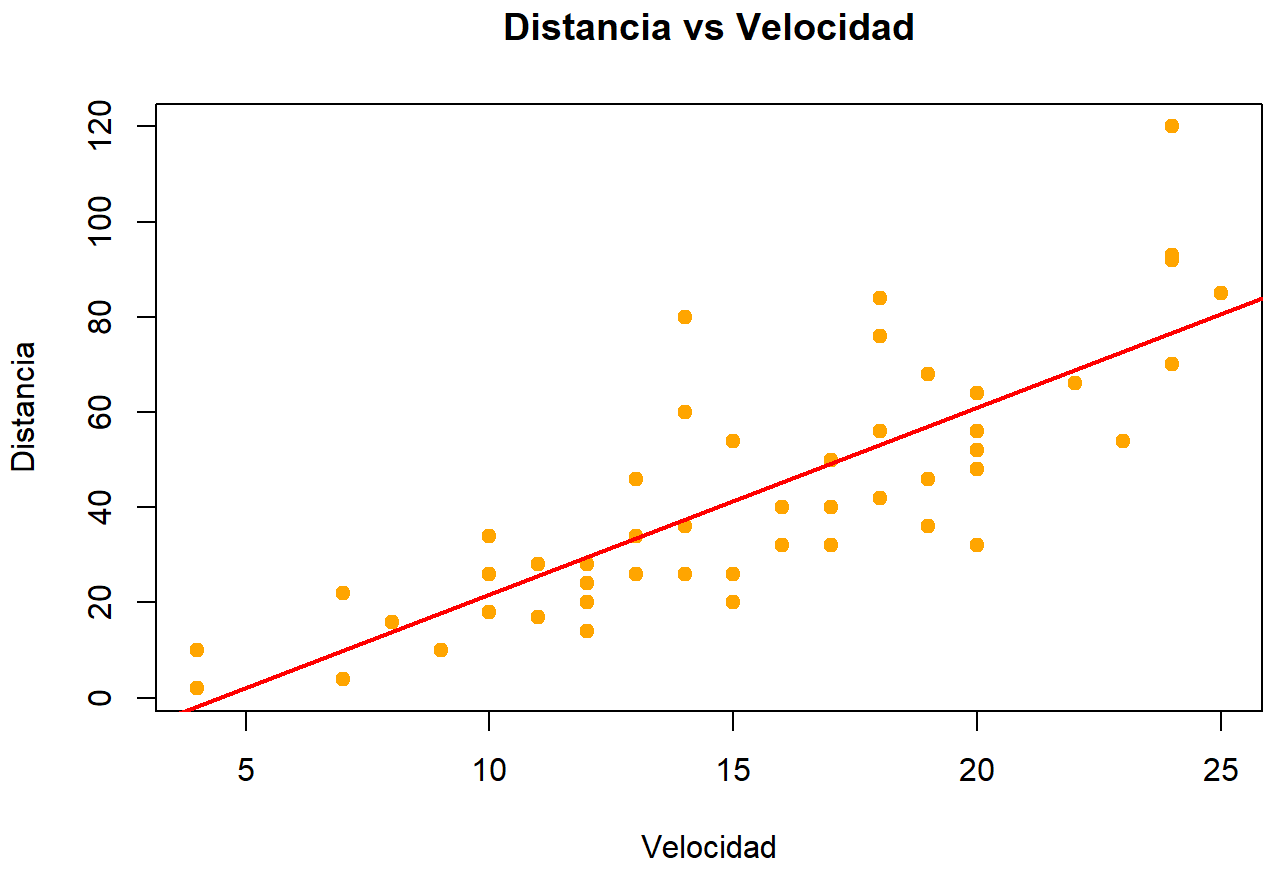

Modelo de Regresión Lineal Simple
Q-Q Residuals
Standardized residuals
Theoretical Quantiles
lm(distancia ~ velocidad)
23
35
49


Modelo de Regresión Lineal Simple
Scale-Location
√|Standardized residuals|
23
49
35
0.0
0.5
1.0
1.5
0
20
40
60
80
Fitted values
lm(distancia ~ velocidad)


Modelo de Regresión Lineal Simple
Residuals vs Leverage
Standardized residuals
23
49
39
0.5
Cook's distance
-2
-1
0
1
2
3
0.00
0.02
0.04
0.06
0.08
0.10
Leverage
lm(distancia ~ velocidad)

4. Grafica los datos y el modelo de la distancia en función de la velocidad.

```
plot(velocidad, distancia, main = "Distancia vs Velocidad", xlab = "Velocidad", ylab = "Distanci
a", pch = 19, col = "orange")

abline(modeloRLS, col = "red", lwd = 2)
```



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

Con base en los resultados anteriores se puede decir que este modelo tiene un buen desempeño, y se ajusta a los datos. Sin embargo, hay espacio para mejoras ya que se puede decir que estos datos no se comportan como una normal ya que el p-valor fue de 0.02152. Igualmente, se observó un valor de R^2 no tan alto con 0.6511.

Parte 3: Regresión No Lineal

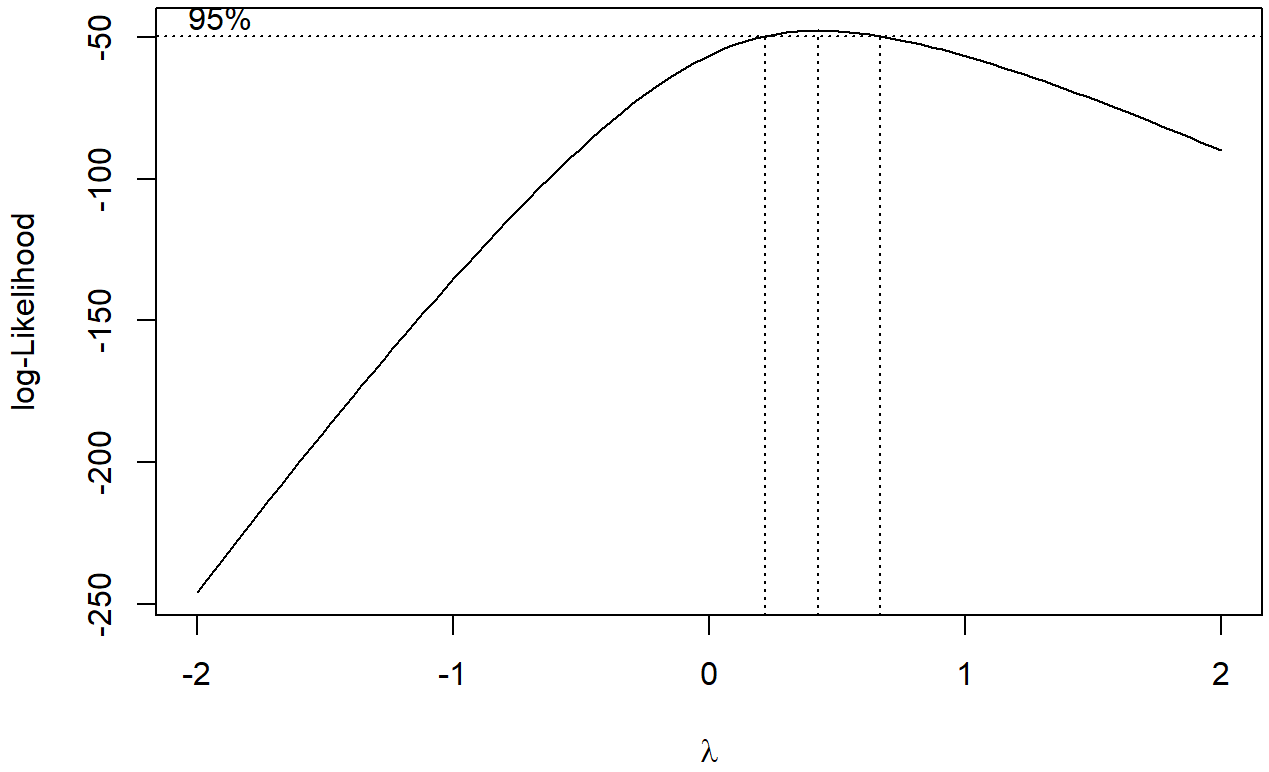
1. Con el objetivo de probar un modelo no lineal que explique la relación entre la distancia y la velocidad, haz una transformación con la base de datos car que te garantice normalidad en ambas variables (ojo: concentrate solo en la variable que tiene más alejamiento de normalidad).

- Encuentra el valor de λ en la transformación Box-Cox para el modelo lineal: $Y = \beta_0 + \beta_1 X$ 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.

- Utiliza: boxcox(lm(Velocidad~Distancia)) si la variable con más alejamiento de normalidad es la velocidad.
- La transformación se hará sobre la variable que usas como dependiente en el comando lm(y~x).

De acuerdo con los test de normalidad hechos anteriormente y usando un nivel de significancia de 0.05, se puede decir que la variable con más alejamiento de normalidad es la distancia ya que tiene un p-valor de 0.0391.

```
# Distancia

modeloDist <- lm(distancia ~ velocidad)
bc <- boxcox(modeloDist)
```




```
lambda_optimo <- bc$x[which.max(bc$y)]

cat("El valor óptimo de lambda es:", lambda_optimo)
```

```
## El valor óptimo de lambda es: 0.4242424
```

- Define la transformación exacta y el aproximada de acuerdo con el valor de que encontraste en la transformación de Box y Cox. Escribe las ecuaciones de las dos transformaciones encontradas.

Exacta

$$Y(x) = \frac{(x+1)^{\lambda} - 1}{\lambda}$$

donde $Y = $ distancia, y $\lambda = 0.4242$

Aproximada

$$Y(x) = \sqrt{x+1}$$

```
bc_exacto <- ((distancia + 1) ^ lambda_optimo - 1) / lambda_optimo

bc_aprox <- sqrt(distancia + 1)
```

- Analiza la normalidad de las transformaciones obtenidas. Utiliza como argumento de normalidad:
    - Compara las medidas: sesgo y curtosis.
    - Obten el histograma de los 2 modelos obtenidos (exacto y aproximado) y los datos originales.
    - Realiza algunas pruebas de normalidad para los datos transformados.

```
# Sesgo y curtosis

sesgo_original <- skewness(distancia)
curtosis_original <- kurtosis(distancia)

cat("Coeficiente de sesgo para distancia:", sesgo_original, "\n")
```

```
## Coeficiente de sesgo para distancia: 0.7591268
```

```
cat("Coeficiente de curtosis para distancia:", curtosis_original, "\n\n")
```

```
## Coeficiente de curtosis para distancia: 0.1193971
```

```
sesgo_exacta <- skewness(bc_exacto)
curtosis_exacta <- kurtosis(bc_exacto)

cat("Coeficiente de sesgo para la transformación exacta:", sesgo_exacta, "\n")
```

```
## Coeficiente de sesgo para la transformación exacta: -0.1126696
```

```
cat("Coeficiente de curtosis para la transformación exacta:", curtosis_exacta, "\n\n")
```

```
## Coeficiente de curtosis para la transformación exacta: -0.2758653
```

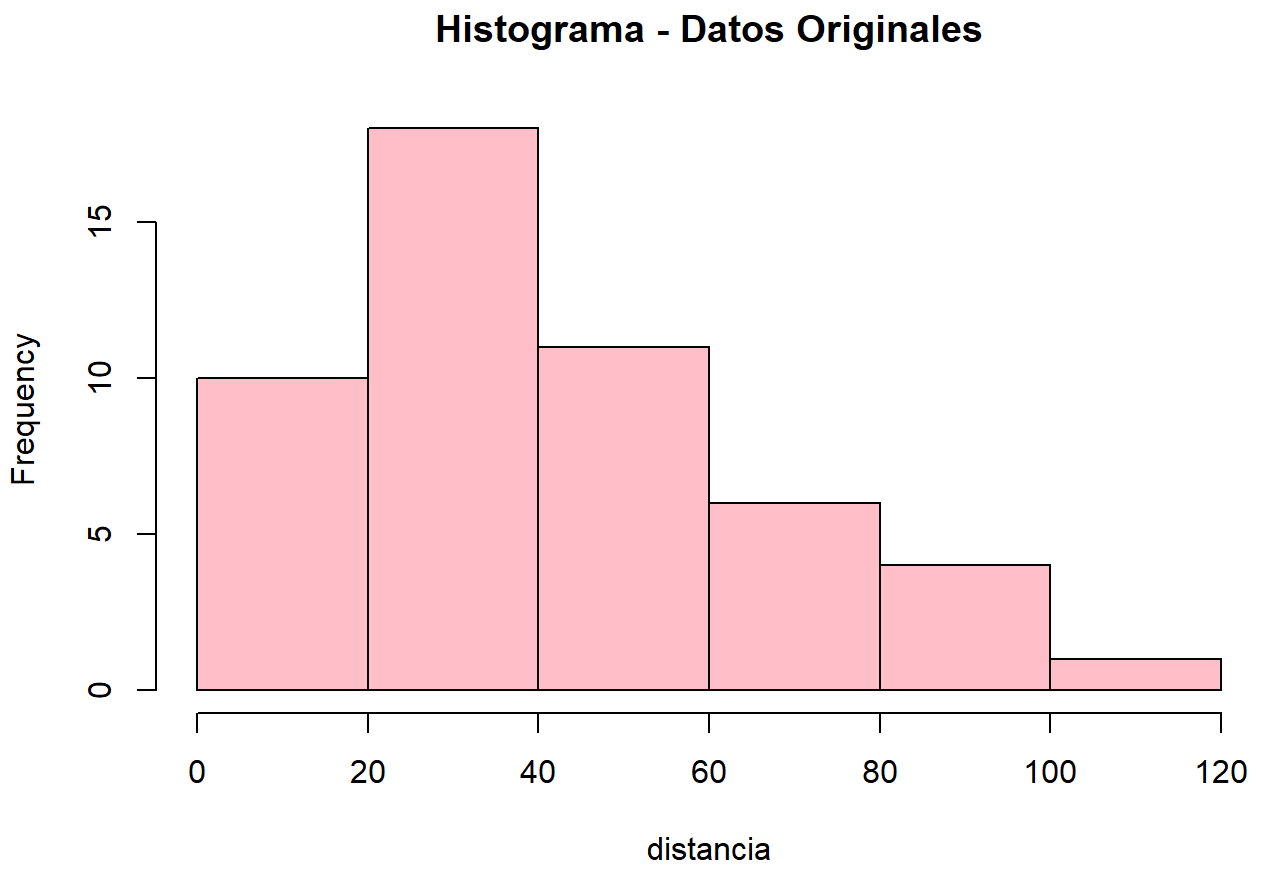
```
sesgo_aproximada <- skewness(bc_aprox)
curtosis_aproximada <- kurtosis(bc_aprox)

cat("Coeficiente de sesgo para la transformación aproximada:", sesgo_aproximada, "\n")
```

```
## Coeficiente de sesgo para la transformación aproximada: 0.02430858
```

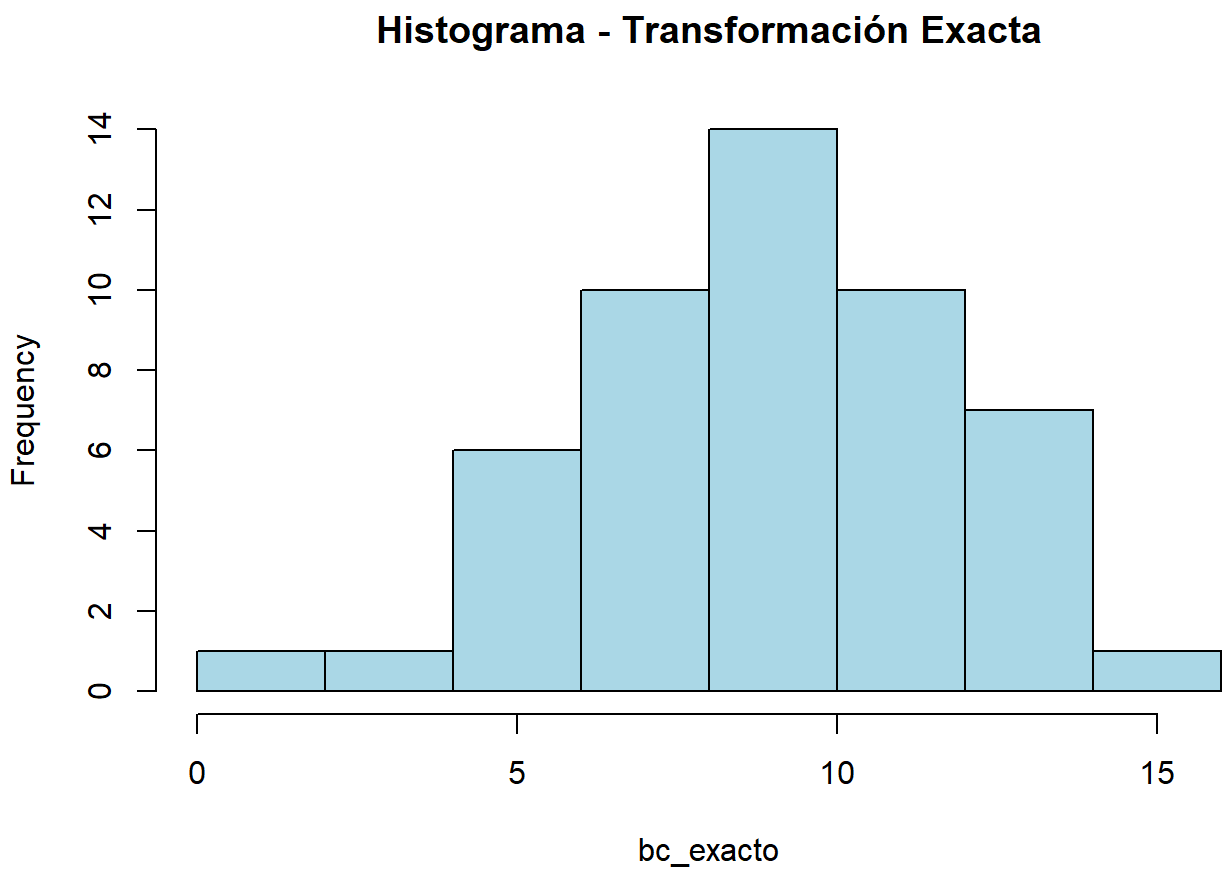
```
cat("Coeficiente de curtosis para la transformación aproximada:", curtosis_aproximada, "\n\n")
```

```
## Coeficiente de curtosis para la transformación aproximada: -0.3647174
```

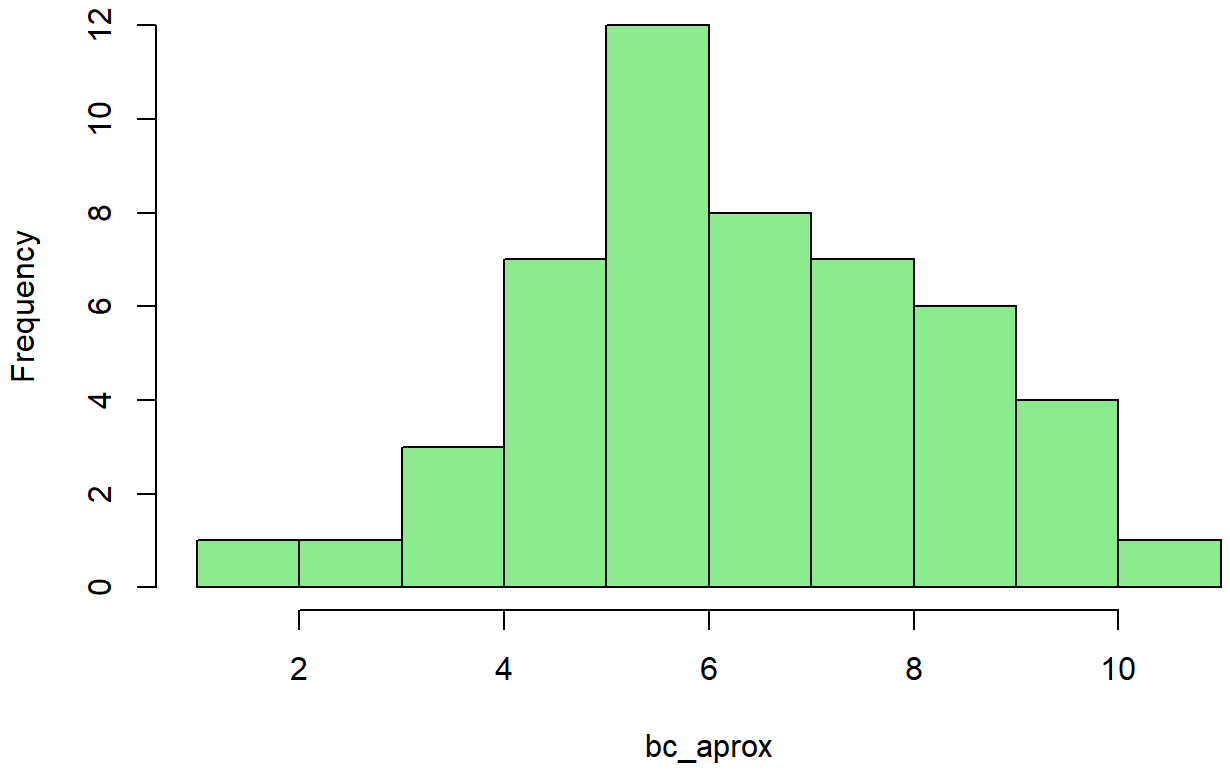
```
# Histogramas

hist(distancia, main = "Histograma - Datos Originales", col = "pink")
```




```
hist(bc_exacto, main = "Histograma - Transformación Exacta", col = "lightblue")
```




```
hist(bc_aprox, main = "Histograma - Transformación Aproximada", col = "lightgreen")
```




```
# Normalidad

shap_original <- shapiro.test(distancia)
shap_exacta <- shapiro.test(bc_exacto)
shap_aproximada <- shapiro.test(bc_aprox)

cat("Resultados del test de Shapiro-Wilk para distancia: \n\n")
```

```
## Resultados del test de Shapiro-Wilk para distancia:
```

```
print(shap_original)
```

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

```
cat("\n\n")
```

```
cat("Resultados del test de Shapiro-Wilk para la transformación exacta: \n\n")
```

```
## Resultados del test de Shapiro-Wilk para la transformación exacta:
```

```
print(shap_exacta)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  bc_exacto
## W = 0.99268, p-value = 0.9885
```

```
cat("\n\n")
```

```
cat("Resultados del test de Shapiro-Wilk para la transformación aproximada: \n\n")
```

```
## Resultados del test de Shapiro-Wilk para la transformación aproximada:
```

```
print(shap_aproximada)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  bc_aprox
## W = 0.99338, p-value = 0.9936
```

- Detecta anomalías y corrige tu base de datos tranformado (datos atípicos, ceros anámalos, etc): solo en caso de no tener normalidad en las transformaciones. En caso de corrección de los datos por anomalías, vuelve a buscar la λ para tus nuevos datos.

```
detect_outliers <- function(x) {
  Q1 <- quantile(x, 0.25)
  Q3 <- quantile(x, 0.75)
  IQR <- Q3 - Q1
  lower_bound <- Q1 - 1.5 * IQR
  upper_bound <- Q3 + 1.5 * IQR
  return(which(x < lower_bound | x > upper_bound))}

outliers_exacta <- detect_outliers(bc_exacto)
outliers_aproximada <- detect_outliers(bc_aprox)

print("Outliers en la transformación exacta:")
```

```
## [1] "Outliers en la transformación exacta:"
```

```
print(outliers_exacta)
```

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

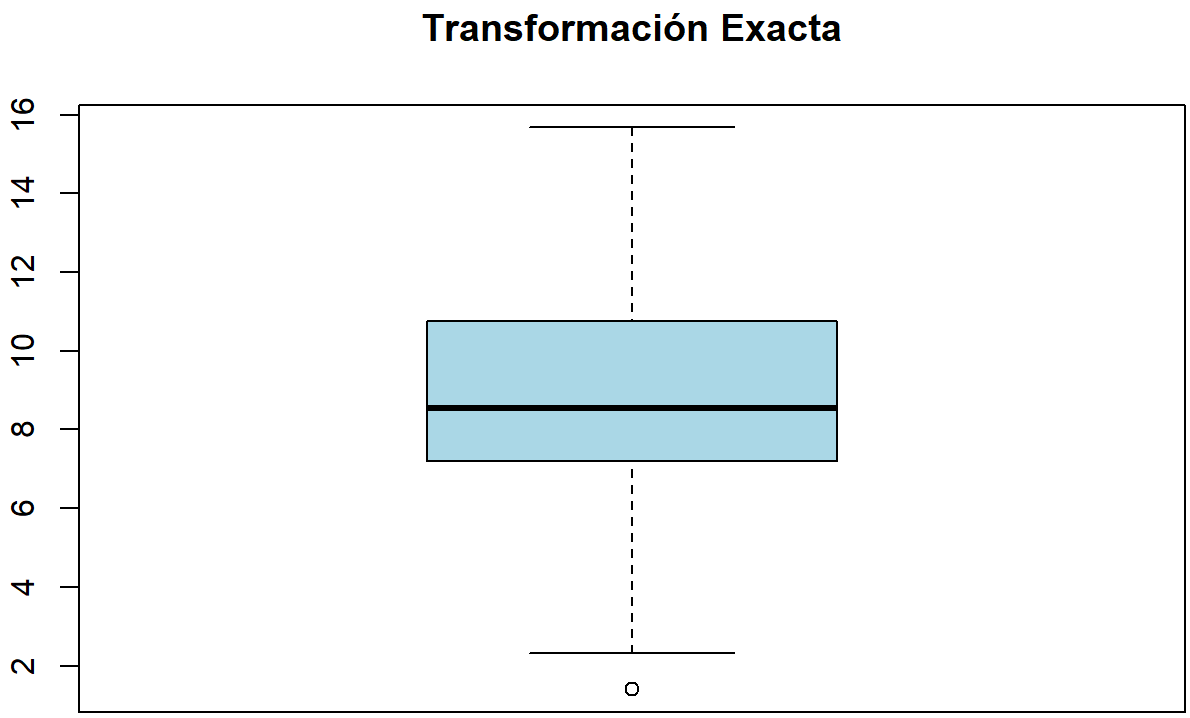
```
print("Outliers en la transformación aproximada:")
```

```
## [1] "Outliers en la transformación aproximada:"
```

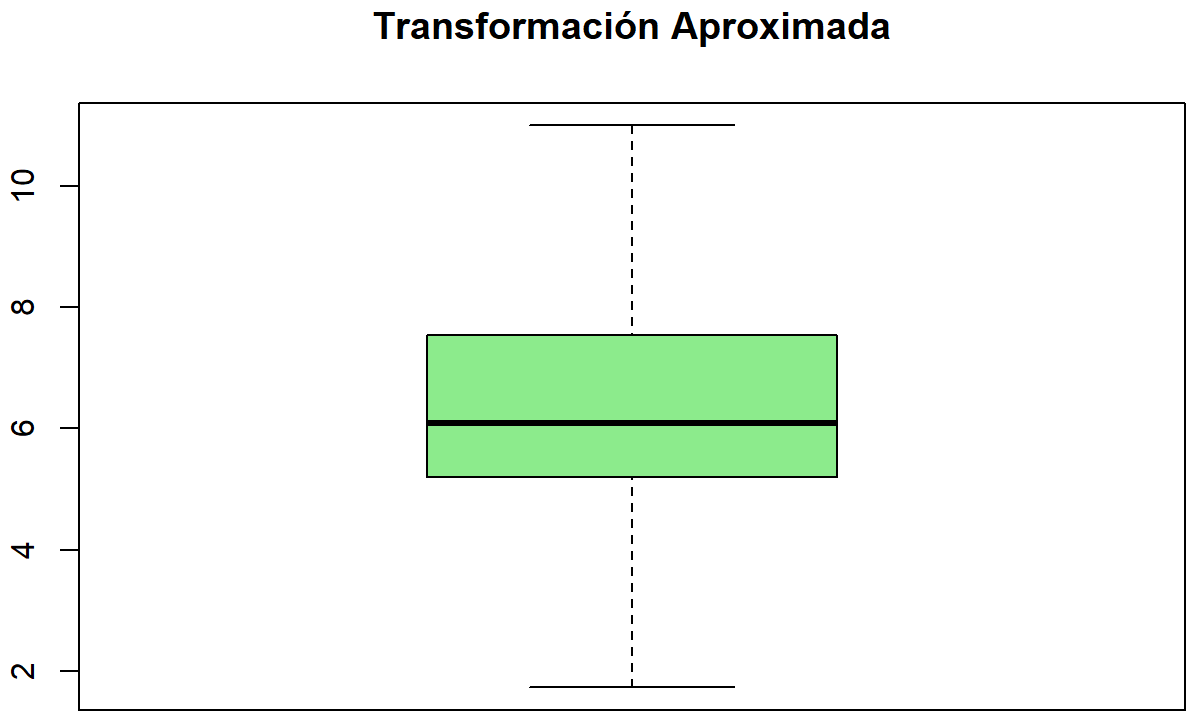
```
print(outliers_aproximada)
```

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

```
boxplot(bc_exacto, main = "Transformación Exacta", col = "lightblue")
```




```
boxplot(bc_aprox, main = "Transformación Aproximada", col = "lightgreen")
```




```
bc_exacto <- bc_exacto[-outliers_exacta[1]]
```

A pesar de que había un outlier, sí se pasaron las pruebas de normalidad.

2. 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.

Observando los datos, se puede decir que en el caso de la transformación exacta hay un sesgo de -0.1127, que a comparación del de la transformación aproximada, indica una mejor simetría. Por otro lado, en el caso del p-value de la exacta, se tiene que este es 0.9885, lo que nos dice que los datos se comportan como una normal. En el caso de la aproximada, hubo un p-value de 0.9936, lo que nos dice lo mismo.

Para concluir, ambas transformaciones mejoraron la normalidad de los datos, como se puede observar en los diferentes tests que se hicieron y en las gráficas. Sin embargo, se cree que la mejor transformación es la aproximada, ya que además de ser más fácil de implementar, mostró un mejor desmpeño por casi nada.

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

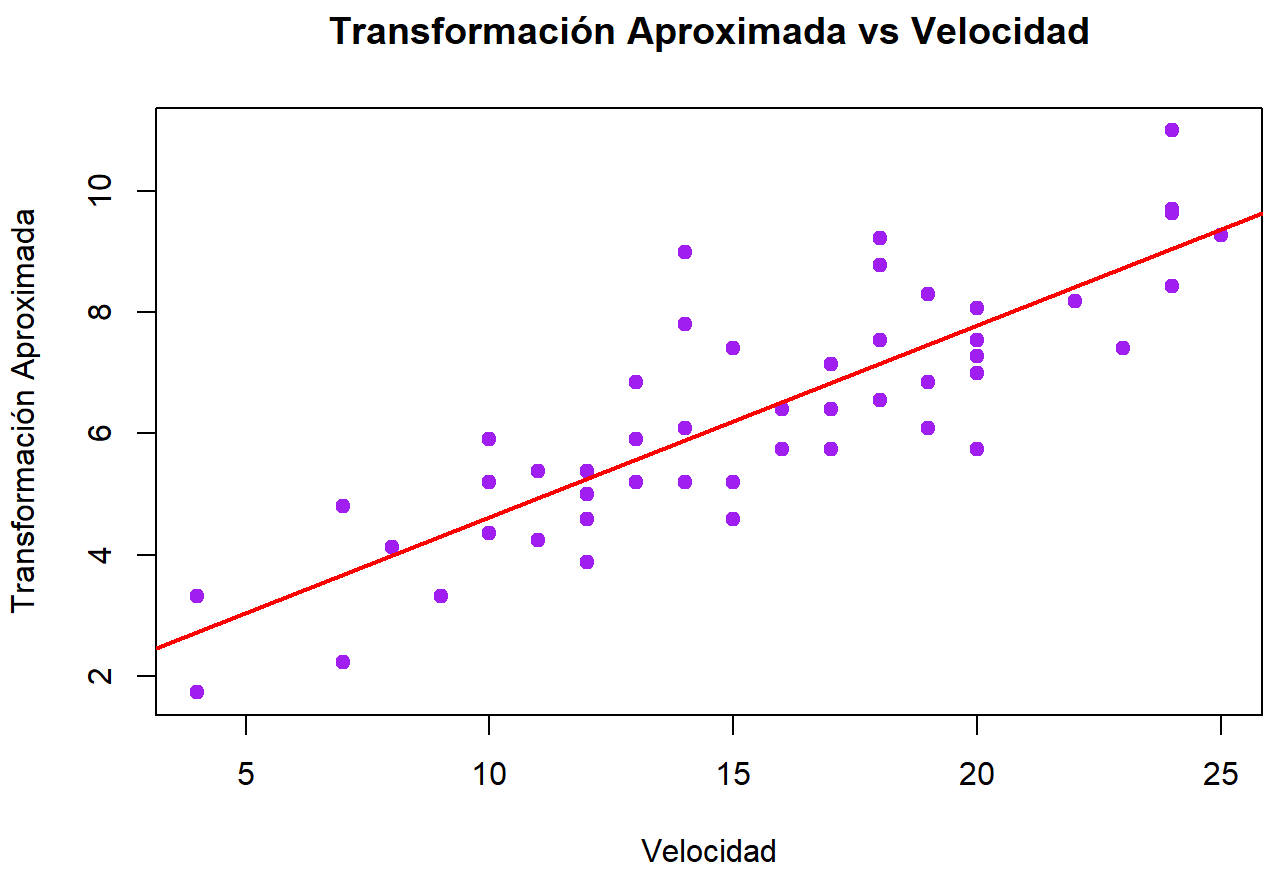
- Escribe el modelo lineal para la transformación.

```
modeloAprox <- lm(bc_aprox ~ velocidad)

summary(modeloAprox)
```

```
##
## Call:
## lm(formula = bc_aprox ~ velocidad)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0430 -0.6992 -0.1773  0.5815  3.1087
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.46680    0.47605   3.081  0.00341 **
## velocidad    0.31604    0.02927  10.798 1.93e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.083 on 48 degrees of freedom
## Multiple R-squared:  0.7084, Adjusted R-squared:  0.7023
## F-statistic: 116.6 on 1 and 48 DF,  p-value: 1.931e-14
```

- Grafica los datos y el modelo lineal (ecuación) de la transformación elegida vs velocidad.

```
plot(velocidad, bc_aprox, main = "Transformación Aproximada vs Velocidad",
     xlab = "Velocidad", ylab = "Transformación Aproximada", pch = 19, col = "purple")
abline(modeloAprox, col = "red", lwd = 2)
```



- Analiza significancia del modelo (individual, conjunta y coeficiente de correlación)

```
resumen_modelo <- summary(modeloAprox)

cat("Coeficientes del modelo:\n")
```

```
## Coeficientes del modelo:
```

```
cat("Intercepto:", resumen_modelo$coefficients[1, 1], "\n")
```

```
## Intercepto: 1.4668
```

```
cat("Velocidad:", resumen_modelo$coefficients[2, 1], "\n")
```

```
## Velocidad: 0.3160374
```

```
cat("\nValores p para los coeficientes:\n")
```

```
##
## Valores p para los coeficientes:
```

```
cat("Intercepto:", resumen_modelo$coefficients[1, 4], "\n")
```

```
## Intercepto: 0.003409087
```

```
cat("Velocidad:", resumen_modelo$coefficients[2, 4], "\n")
```

```
## Velocidad: 1.931291e-14
```

```
cat("\nValor p del test F para el modelo:", resumen_modelo$fstatistic[1], "\n")
```

```
##
## Valor p del test F para el modelo: 116.6006
```

```
cat("\nCoeficiente de determinación R^2:", resumen_modelo$r.squared, "\n")
```

```
##
## Coeficiente de determinación R^2: 0.7083851
```

- 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.

```
# Media cero

mediaCero <- t.test(resid(modeloAprox))

cat("El resultado del test de media cero para el modelo es: \n\n")
```

```
## El resultado del test de media cero para el modelo es:
```

```
print(mediaCero)
```

```
##
##  One Sample t-test
##
## data:  resid(modeloAprox)
## t = -1.2242e-16, df = 49, p-value = 1
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.3047124  0.3047124
## sample estimates:
##     mean of x
## -1.856263e-17
```

```
cat("\n\n")
```

```
# Normalidad

norm <- shapiro.test(resid(modeloAprox))
cat("Los resultados de la normalidad para el modelo usando el test de Shapiro-Wilk son: \n\n")
```

```
## Los resultados de la normalidad para el modelo usando el test de Shapiro-Wilk son:
```

```
print(norm)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(modeloAprox)
## W = 0.9723, p-value = 0.2863
```

```
cat("\n\n")
```

```
# Homocedasticidad

hom1 <- bptest(modeloAprox)
hom2 <- gqtest(modeloAprox)

cat("Los resultados de los tests de homocedasticidad para el modelo son: \n\n")
```

```
## Los resultados de los tests de homocedasticidad para el modelo son:
```

```
print(hom1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  modeloAprox
## BP = 0.054505, df = 1, p-value = 0.8154
```

```
print(hom2)
```

```
##
##  Goldfeld-Quandt test
##
## data:  modeloAprox
## GQ = 0.85916, df1 = 23, df2 = 23, p-value = 0.6405
## alternative hypothesis: variance increases from segment 1 to 2
```

```
cat("\n\n")
```

```
# Independencia

ind1 <- dwtest(modeloAprox)
ind2 <- bgtest(modeloAprox)

cat("Los resultados de los tests de independencia para el modelo son: \n\n")
```

```
## Los resultados de los tests de independencia para el modelo son:
```

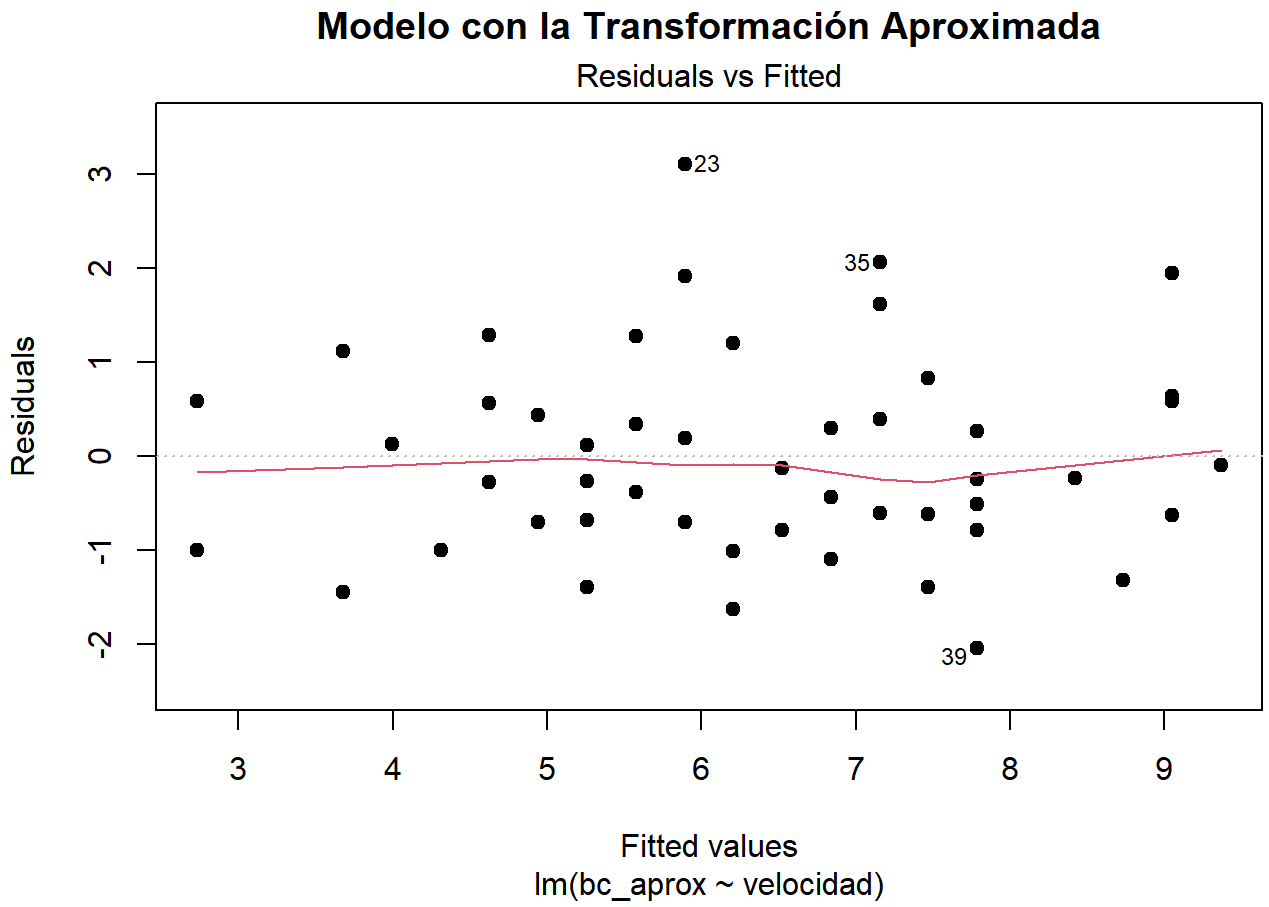
```
print(ind1)
```

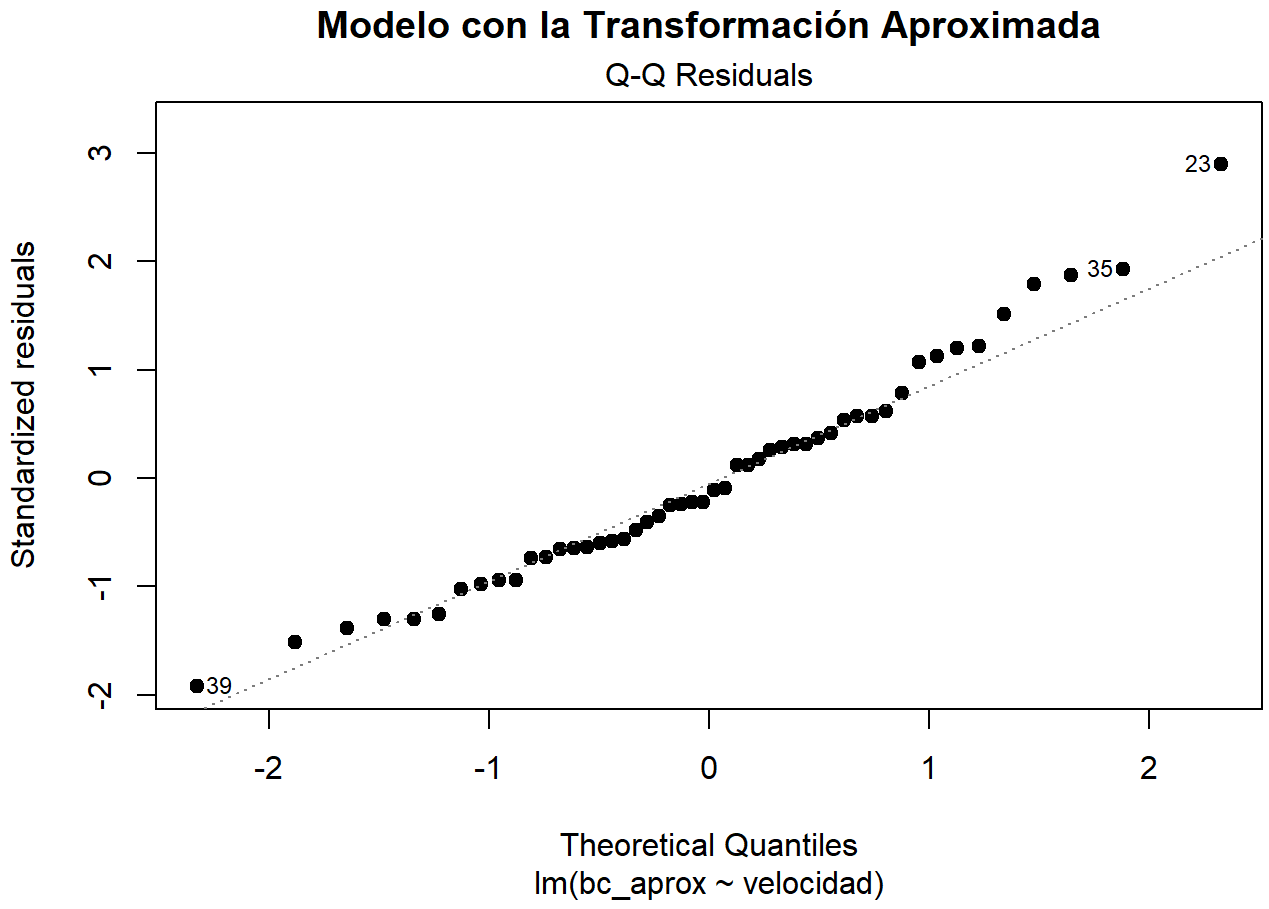
```
##
##  Durbin-Watson test
##
## data:  modeloAprox
## DW = 1.9356, p-value = 0.3527
## alternative hypothesis: true autocorrelation is greater than 0
```

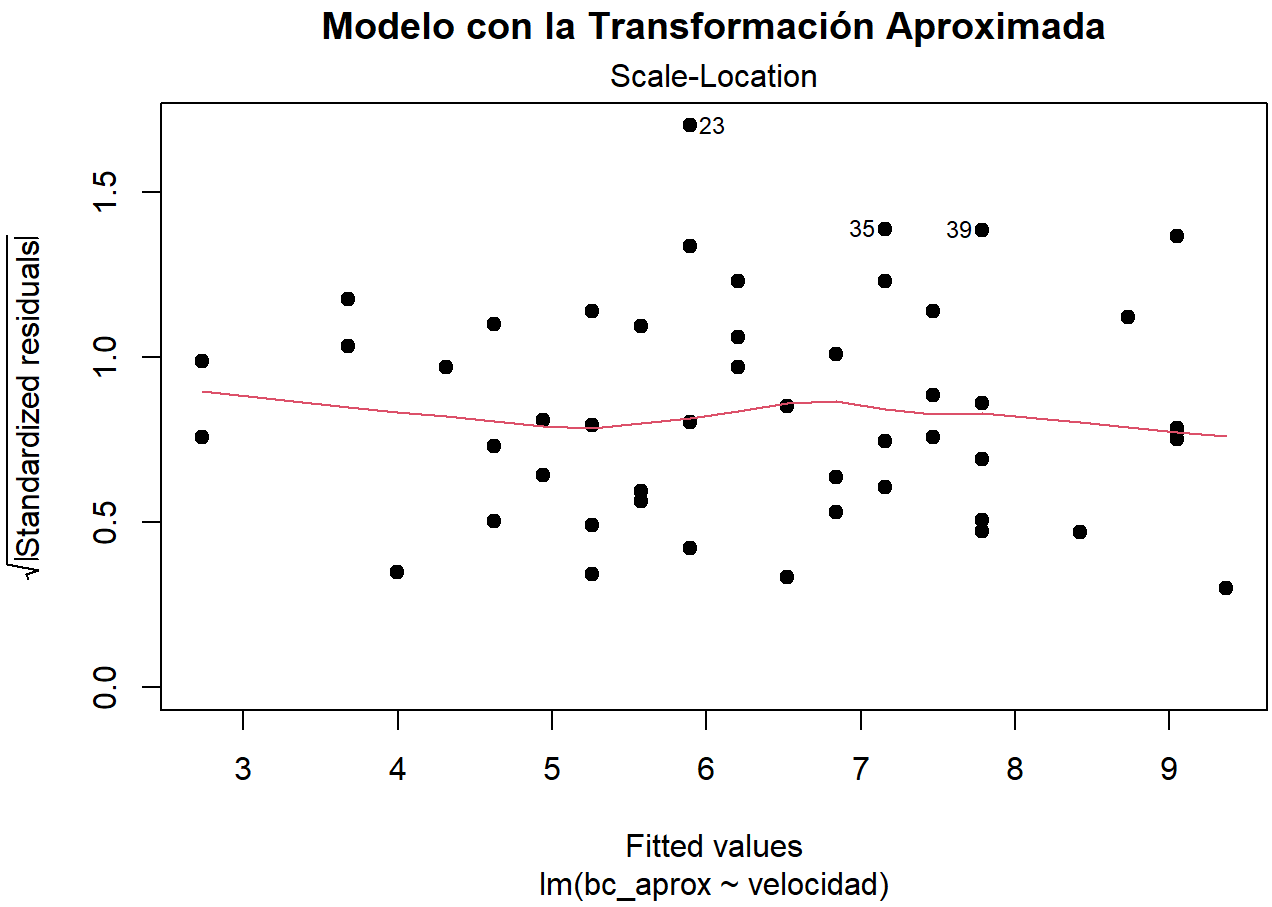
```
print(ind2)
```

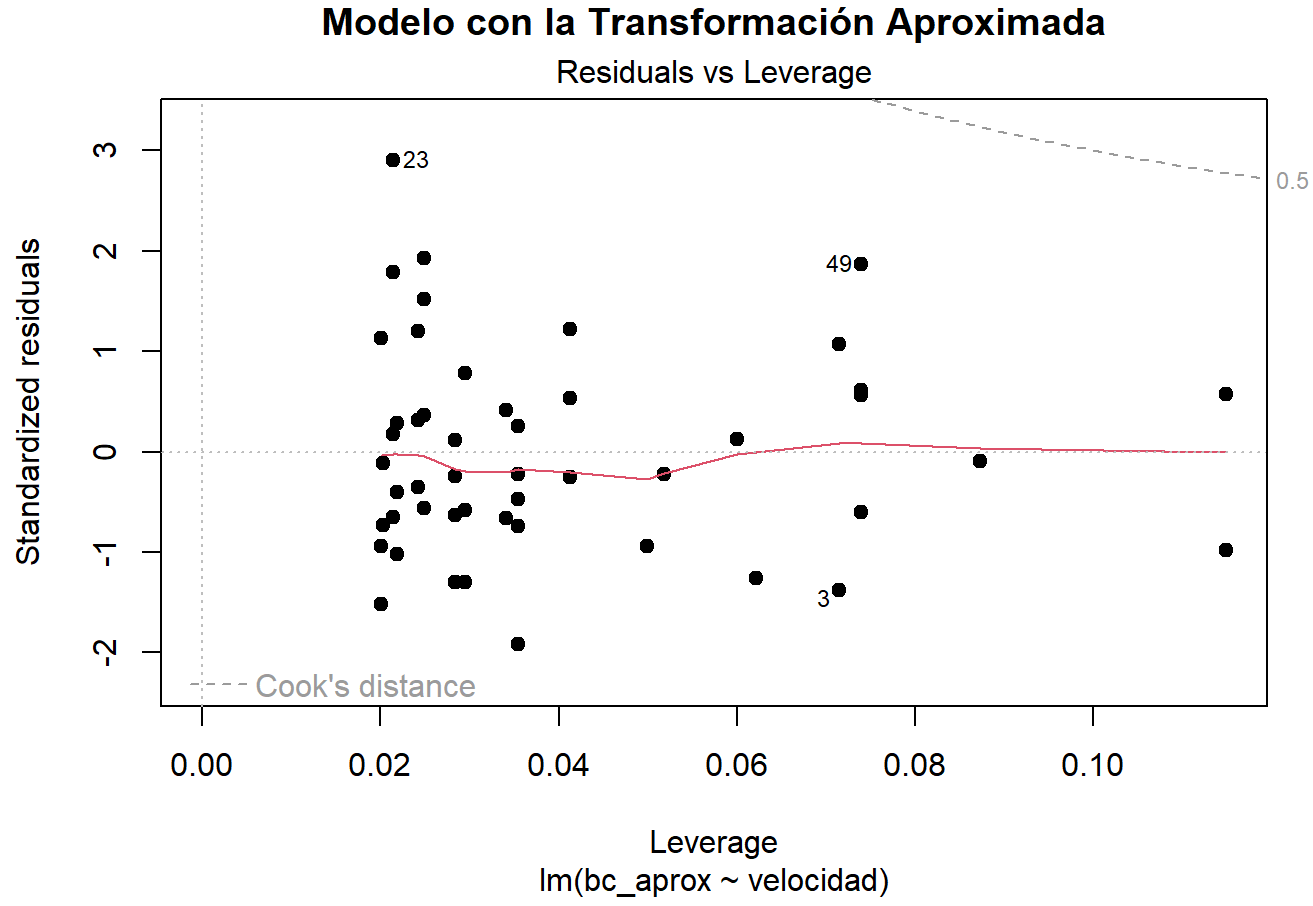
```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  modeloAprox
## LM test = 0.027137, df = 1, p-value = 0.8692
```

```
# Linealidad

# Gráficos

plot(modeloAprox, pch = 19, main = "Modelo con la Transformación Aproximada")
```


Modelo con la Transformación Aproximada
Residuals vs Fitted
Residuals
Fitted values
lm(bc_aprox ~ velocidad)
23
35
39


Modelo con la Transformación Aproximada
Q-Q Residuals
Standardized residuals
Theoretical Quantiles
lm(bc_aprox ~ velocidad)
23
35
39


Modelo con la Transformación Aproximada
Scale-Location
√|Standardized residuals|
23
35
39
0.0
0.5
1.0
1.5
3
4
5
6
7
8
9
Fitted values
lm(bc_aprox ~ velocidad)

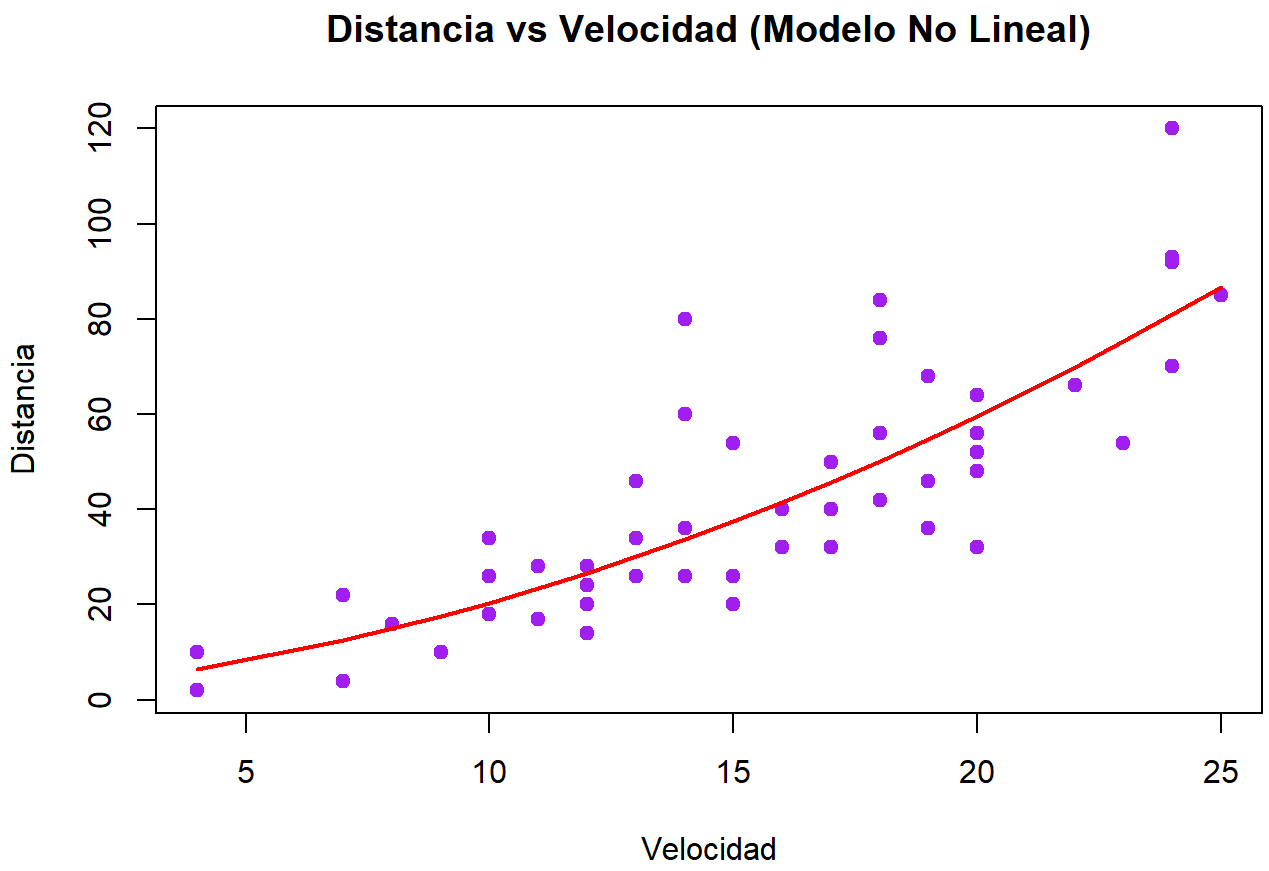

Modelo con la Transformación Aproximada
Residuals vs Leverage
Standardized residuals
23
49
3
0.5
Cook's distance
-2
-1
0
1
2
3
0.00
0.02
0.04
0.06
0.08
0.10
Leverage
lm(bc_aprox ~ velocidad)

- Despeja la distancia del modelo lineal obtenido entre la transformación y la velocidad. Obtendrás el modelo no lineal que relaciona la distancia con la velocidad directamente (y no con su transformación).

```
beta_0 <- coef(modeloAprox)[1]
beta_1 <- coef(modeloAprox)[2]

despeje <- (beta_0 + beta_1 * velocidad)^2 - 1
```

- Grafica los datos y el modelo de la distancia en función de la velocidad.

```
plot(velocidad, distancia, main = "Distancia vs Velocidad (Modelo No Lineal)",
     xlab = "Velocidad", ylab = "Distancia", pch = 19, col = "purple")
lines(velocidad, despeje, col = "red", lwd = 2)
```



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

Con los análisis anteriores se puede decir que el modelo es bueno ya que hay alta significancia y un R^2 alto, de 0.7083. Igualmente, pasa con todos los tests de normalidad, homocedasticidad e independencia. Por otro lado, dado que ya se removieron los valores atípicos, puede haber un mejor desempeño. En el caso de la gráfica, observamos cómo se ajusta bien a los datos.

Parte 4: Conclusiones

1. 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.

Habiendo hecho ambos modelos, regresión lineal simple y regresión no lineal, se puede decir que el mejor es el modelo de regresión no lineal, aplicando la transformación de datos. Esto es ya que tiene un mejor desempeño en sus métricas, como lo es la R^2, además de que obtuvo mejores resultados en la normalidad de los residuos. Asimismo, se puede observar un mejor ajuste a los datos.

2. Comenta sobre posibles problemas del modelo elegido (datos atípicos, alejamiento de los supuestos, dificultad de cálculo o interpretación).

A pesar de que el modelo fue bueno, a veces la transformación no lineal puede llegar a complicar la interpretación de los coeficientes. Igualmente, para que esta funcione, es necesario que se invierta para interpretar los resultados de la distancia, lo que puede llegar a dificultar las cosas.