Act13 Regresión No Lineal

Luis Maximiliano López Ramírez

2024-09-10

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.

 # Parte 1: Análisis de normalidad

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)

```
# Cargar los datos y librerías necesarias
data(cars)
library(tseries)
```

```
## Warning: package 'tseries' was built under R version 4.3.3
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
# Prueba de normalidad Shapiro-Wilk para velocidad (speed)
shapiro_speed <- shapiro.test(cars$speed)
cat("Prueba Shapiro-Wilk para Velocidad (speed):\n")
```

```
## Prueba Shapiro-Wilk para Velocidad (speed):
```

```
print(shapiro_speed)
```

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

```
# Prueba de normalidad Shapiro-Wilk para distancia (dist)
shapiro_dist <- shapiro.test(cars$dist)
cat("\nPrueba Shapiro-Wilk para Distancia (dist):\n")
```

```
##
## Prueba Shapiro-Wilk para Distancia (dist):
```

```r
print(shapiro_dist)
```

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

```r
# Prueba de normalidad Jarque-Bera para velocidad (speed)
jarque_bera_speed <- jarque.bera.test(cars$speed)
cat("\nPrueba Jarque-Bera para Velocidad (speed):\n")
```

```
##
## Prueba Jarque-Bera para Velocidad (speed):
```

```r
print(jarque_bera_speed)
```

```
##
##  Jarque Bera Test
##
## data:  cars$speed
## X-squared = 0.80217, df = 2, p-value = 0.6696
```

```r
# Prueba de normalidad Jarque-Bera para distancia (dist)
jarque_bera_dist <- jarque.bera.test(cars$dist)
cat("\nPrueba Jarque-Bera para Distancia (dist):\n")
```

```
##
## Prueba Jarque-Bera para Distancia (dist):
```

```r
print(jarque_bera_dist)
```

```
##
##  Jarque Bera Test
##
## data:  cars$dist
## X-squared = 5.2305, df = 2, p-value = 0.07315
```

```r
# Función para interpretar resultados de normalidad
interpret_normality <- function(test_result, variable_name) {
  if (test_result$p.value < 0.05) {
    cat(variable_name, "NO cumple con normalidad (p-value:",
test_result$p.value, ").\n")
  } else {
    cat(variable_name, "cumple con normalidad (p-value:",
test_result$p.value, ").\n")
  }
}

# Interpretación de los resultados
cat("\nInterpretación de los resultados:\n")
```

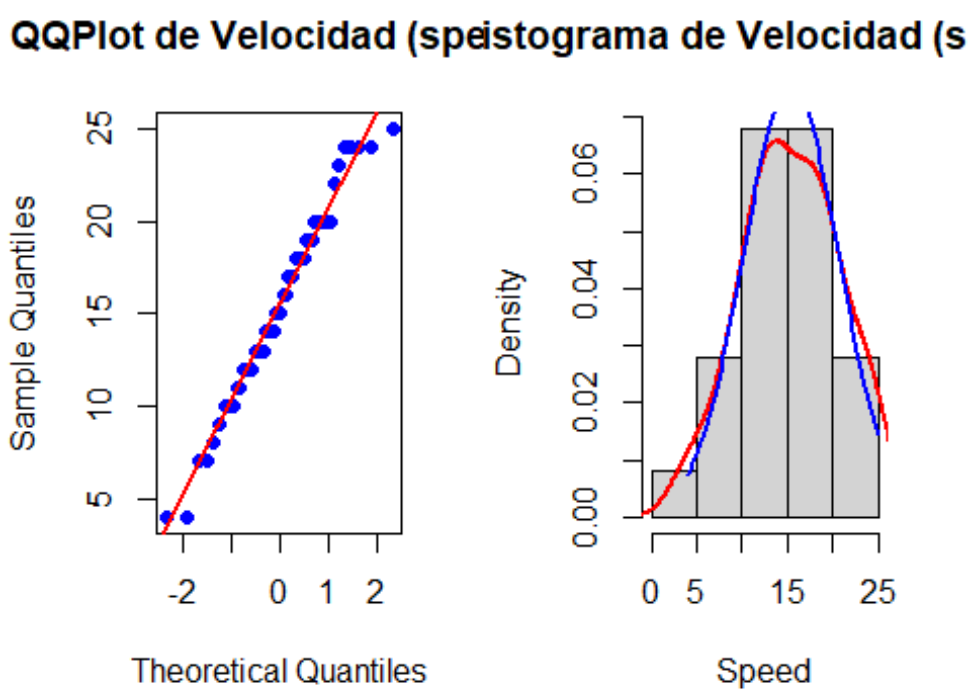
```
##
## Interpretación de los resultados:

interpret_normality(shapiro_speed, "Velocidad (speed) - Shapiro-Wilk")

## Velocidad (speed) - Shapiro-Wilk cumple con normalidad (p-value:
0.4576319 ).

interpret_normality(shapiro_dist, "Distancia (dist) - Shapiro-Wilk")

## Distancia (dist) - Shapiro-Wilk NO cumple con normalidad (p-value:
0.03909968 ).

interpret_normality(jarque_bera_speed, "Velocidad (speed) - Jarque-Bera")

## Velocidad (speed) - Jarque-Bera cumple con normalidad (p-value:
0.6695929 ).

interpret_normality(jarque_bera_dist, "Distancia (dist) - Jarque-Bera")

## Distancia (dist) - Jarque-Bera cumple con normalidad (p-value:
0.07314987 ).
```

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

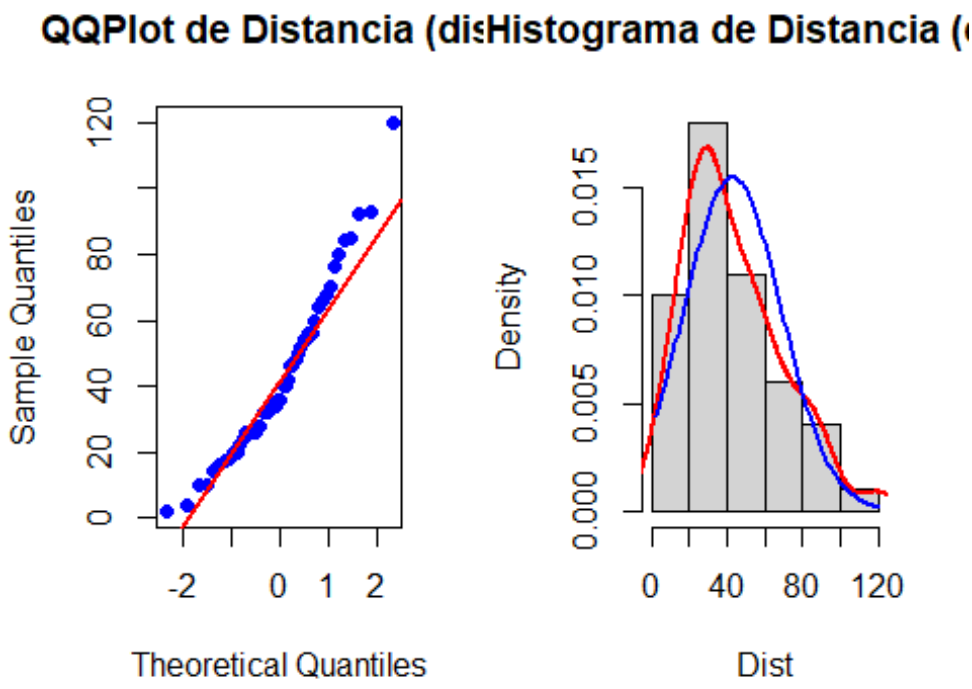
Los datos y su respectivo QQPlot: qqnorm(datos) y qqline(datos) para cada variable.

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

```
# Cargar los datos
data(cars)

# Establecer el espacio de los gráficos para dos columnas
par(mfrow = c(1, 2))

# Gráficos para la variable velocidad (speed)
# QQPlot para velocidad
qqnorm(cars$speed, main = "QQPlot de Velocidad (speed)", pch = 19, col =
"blue")
qqline(cars$speed, col = "red", lwd = 2)

# Histograma y densidad para velocidad
hist(cars$speed, freq = FALSE, main = "Histograma de Velocidad (speed)",
     xlab = "Speed", col = "lightgray", border = "black")
lines(density(cars$speed), col = "red", lwd = 2)
curve(dnorm(x, mean = mean(cars$speed), sd = sd(cars$speed)),
      from = min(cars$speed), to = max(cars$speed), add = TRUE, col =
"blue", lwd = 2)
```




```r
# Gráficos para la variable distancia (dist)
# QQPlot para distancia
qqnorm(cars$dist, main = "QQPlot de Distancia (dist)", pch = 19, col =
"blue")
qqline(cars$dist, col = "red", lwd = 2)

# Histograma y densidad para distancia
hist(cars$dist, freq = FALSE, main = "Histograma de Distancia (dist)",
     xlab = "Dist", col = "lightgray", border = "black")
lines(density(cars$dist), col = "red", lwd = 2)
curve(dnorm(x, mean = mean(cars$dist), sd = sd(cars$dist)),
      from = min(cars$dist), to = max(cars$dist), add = TRUE, col =
"blue", lwd = 2)
```




```
# Resetear el layout de los gráficos
par(mfrow = c(1, 1))
```

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

```
# Cargar los datos y librerías necesarias
data(cars)
library(e1071)

# Calcular el coeficiente de sesgo (skewness) y curtosis (kurtosis) para
velocidad (speed)
skewness_speed <- skewness(cars$speed)
kurtosis_speed <- kurtosis(cars$speed)

# Calcular el coeficiente de sesgo (skewness) y curtosis (kurtosis) para
distancia (dist)
skewness_dist <- skewness(cars$dist)
kurtosis_dist <- kurtosis(cars$dist)

# Mostrar los resultados
cat("Coeficiente de Sesgo y Curtosis:\n")

## Coeficiente de Sesgo y Curtosis:

cat("Velocidad (speed):\n")
```

```
## Velocidad (speed):
cat(" - Sesgo (Skewness):", skewness_speed, "\n")
##  - Sesgo (Skewness): -0.1105533
cat(" - Curtosis:", kurtosis_speed, "\n\n")
##  - Curtosis: -0.6730924
cat("Distancia (dist):\n")
## Distancia (dist):
cat(" - Sesgo (Skewness):", skewness_dist, "\n")
##  - Sesgo (Skewness): 0.7591268
cat(" - Curtosis:", kurtosis_dist, "\n")
##  - Curtosis: 0.1193971
```

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.

Velocidad (speed):

- Cumple con normalidad según las pruebas de Shapiro-Wilk (p-value: 0.4576) y Jarque-Bera (p-value: 0.6696).
- Los gráficos (QQPlot y histograma) muestran una distribución simétrica sin desviaciones significativas.
- Coeficientes: Sesgo cercano a 0 (-0.1106) y curtosis ligeramente plana (-0.6731), lo cual es insignificante.

Distancia (dist):

- No cumple completamente con normalidad: Shapiro-Wilk rechaza la normalidad (p-value: 0.0391), aunque Jarque-Bera no la rechaza del todo (p-value: 0.0731).
- Los gráficos muestran una asimetría positiva y una cola derecha larga.
- Coeficientes: Sesgo positivo moderado (0.7591) y curtosis cercana a la normalidad (0.1194), pero con colas más pronunciadas.

Conclusión Final:

- speed cumple con la normalidad, mientras que dist no lo hace debido a su asimetría y variabilidad en las colas.
- Las desviaciones en dist podrían deberse a factores externos como condiciones de frenado y errores de medición.

- Para modelar la relación entre speed y dist, es importante considerar la no normalidad de dist, pudiendo requerir transformaciones o métodos robustos.

 # Parte 2: Regresión lineal

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

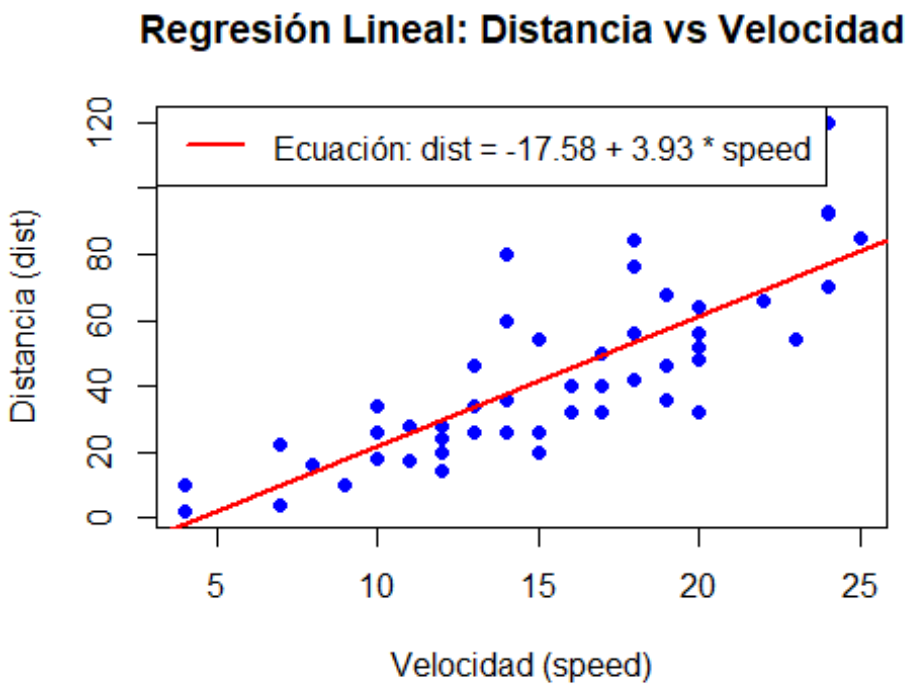
```
# Cargar los datos
data(cars)

# Ajustar el modelo de regresión lineal simple
modelo <- lm(dist ~ speed, data = cars)
```

Escribe el modelo lineal obtenido.

```
# Mostrar el modelo lineal obtenido
cat("Modelo lineal obtenido:\n")

## Modelo lineal obtenido:

print(modelo)

##
## Call:
## lm(formula = dist ~ speed, data = cars)
##
## Coefficients:
## (Intercept)        speed
##     -17.579        3.932
```

Grafica los datos y el modelo (ecuación) que obtuviste.

```
# Graficar los datos y el modelo de regresión
plot(cars$speed, cars$dist,
     main = "Regresión Lineal: Distancia vs Velocidad",
     xlab = "Velocidad (speed)",
     ylab = "Distancia (dist)",
     pch = 19, col = "blue")
abline(modelo, col = "red", lwd = 2)
legend("topleft", legend = paste("Ecuación: dist =",
round(coef(modelo)[1], 2), "+", round(coef(modelo)[2], 2), "* speed"),
       col = "red", lwd = 2)
```



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

```
# Analizar la significancia del modelo
summary_modelo <- summary(modelo)
cat("\nResumen del Modelo:\n")

##
## Resumen del Modelo:

print(summary_modelo)

##
## Call:
## lm(formula = dist ~ speed, data = cars)
##
## 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 *
## speed         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

1. Residuos con media cero

% Hipótesis de Media Diferente de Cero (t de Student)

$$\begin{cases} H_0: \mu = 0 & \text{(El promedio de los residuos es igual a cero)} \\ H_1: \mu \neq 0 & \text{(El promedio de los residuos es diferente de cero)} \end{cases}$$

```
# Realiza la prueba t para los residuos de los modelos sin y con
interacción
resultado_t_modelo <- t.test(modelo$residuals)

# Mostrar los resultados de las pruebas t
cat("Resultados de la prueba t de Student para los residuos:\n")

## Resultados de la prueba t de Student para los residuos:

cat("\nModelo Regresión Lineal:\n")

##
## Modelo Regresión Lineal:

print(resultado_t_modelo)

##
##  One Sample t-test
##
## data:  modelo$residuals
## 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

# Verificar si el promedio de los residuos es significativamente
diferente de cero
if (resultado_t_modelo$p.value > 0.05) {
  cat("\nEl promedio de los residuos del Modelo de Regresión Lineal no es
significativamente diferente de cero.\n")
} else {
  cat("\nEl promedio de los residuos del Modelo de Regresión Lineal es
significativamente diferente de cero.\n")
}
```

```
##
## El promedio de los residuos del Modelo de Regresión Lineal no es
significativamente diferente de cero.
```

2. Normalidad de los residuos

% Hipótesis de Normalidad (Anderson-Darling)

$$\begin{cases} H_0: \text{Los residuos siguen una distribución normal.} \\ H_1: \text{Los residuos no siguen una distribución normal.} \end{cases}$$

```
# Cargar la librería para la prueba Anderson-Darling
library(nortest)

# Realizar la prueba de Jarque Bera sobre los residuos de los modelos
resultado_modelo <- ad.test(modelo$residuals)

# Mostrar los resultados de las pruebas
cat("Resultados de la prueba de normalidad Anderson-Darling:\n")

## Resultados de la prueba de normalidad Anderson-Darling:

cat("\nModelo Regresión Lineal:\n")

##
## Modelo Regresión Lineal:

print(resultado_modelo)

##
##  Anderson-Darling normality test
##
## data:  modelo$residuals
## A = 0.79406, p-value = 0.0369

# Verificar normalidad en base al valor p
if (resultado_modelo$p.value > 0.05) {
  cat("\nSe tiene normalidad en el Modelo Regresión Lineal.\n")
} else {
  cat("\nNo se tiene normalidad en el Modelo Regresión Lineal.\n")
}

##
## No se tiene normalidad en el Modelo Regresión Lineal.
```

3. Homocedasticidad, linealidad e independencia

% Hipótesis de Autocorrelación (Durbin-Watson y Breusch-Godfrey)

$$\begin{cases} H_0: \text{Los errores no están autocorrelacionados (independencia).} \\ H_1: \text{Los errores están autocorrelacionados.} \end{cases}$$

% Hipótesis de Homocedasticidad (Breusch-Pagan y Goldfeld-Quandt)

$$\begin{cases} H_0: \text{La varianza de los errores es constante (homocedasticidad).} \\ H_1: \text{La varianza de los errores no es constante (heterocedasticidad).} \end{cases}$$

```
# Cargar la librería necesaria
library(lmtest)

## Loading required package: zoo

##
## Attaching package: 'zoo'

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

# Modelo 1: Sin Interacción
dw_modelo <- dwtest(modelo)
bg_modelo <- bgtest(modelo)

cat("\nModelo Regresión Lineal:\n")

##
## Modelo Regresión Lineal:

# Verificar autocorrelación en base al valor p
if (dw_modelo$p.value > 0.05) {
  cat("\nNo se rechaza H₀: Los errores no están autocorrelacionados
(Durbin-Watson).\nTiene independencia.\n")
} else {
  cat("\nSe rechaza H₀: Los errores están autocorrelacionados (Durbin-
Watson).\nNo tiene independencia.\n")
}

##
## No se rechaza H₀: Los errores no están autocorrelacionados (Durbin-
Watson).
## Tiene independencia.

if (bg_modelo$p.value > 0.05) {
  cat("No se rechaza H₀: Los errores no están autocorrelacionados
(Breusch-Godfrey).\nTiene independencia.\n")
} else {
  cat("Se rechaza H₀: Los errores están autocorrelacionados (Breusch-
Godfrey).\nNo tiene independencia.\n")
}

## No se rechaza H₀: Los errores no están autocorrelacionados (Breusch-
Godfrey).
## Tiene independencia.

# Modelo 2: Con Interacción
bp_modelo <- bptest(modelo)
```

```r
gq_modelo <- gqtest(modelo)

# Verificar homocedasticidad en base al valor p
if (bp_modelo$p.value > 0.05) {
  cat("\nNo se rechaza H₀: La varianza de los errores es constante
(Breusch-Pagan).\nTiene homocedasticidad.\n")
} else {
  cat("\nSe rechaza H₀: La varianza de los errores no es constante
(Breusch-Pagan).\nNo tiene homocedasticidad.\n")
}
```

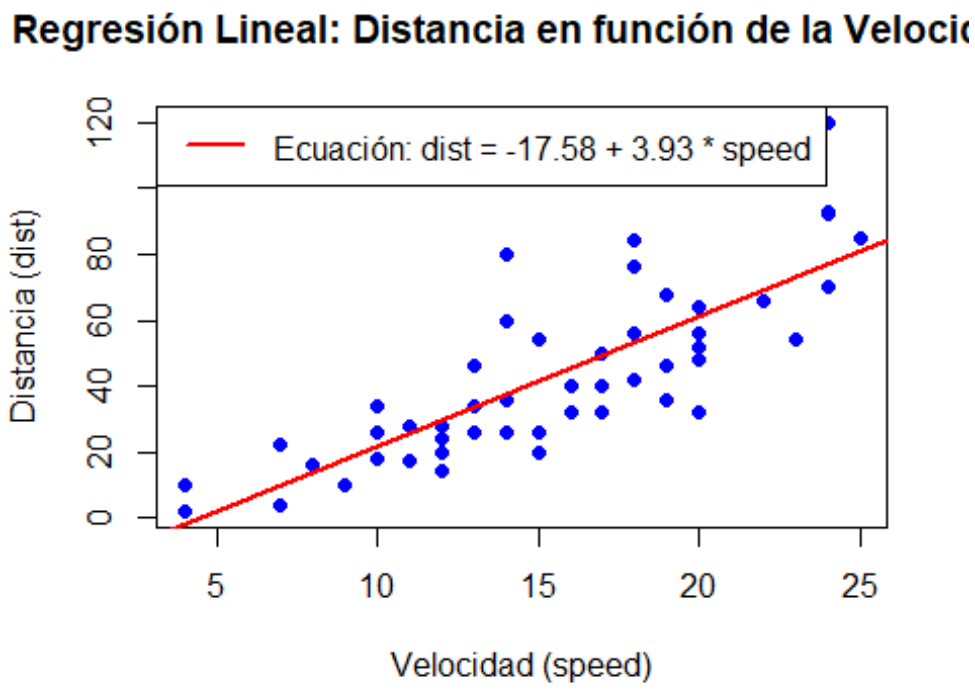
```
##
## No se rechaza H₀: La varianza de los errores es constante (Breusch-
Pagan).
## Tiene homocedasticidad.
```

```r
if (gq_modelo$p.value > 0.05) {
  cat("No se rechaza H₀: La varianza de los errores es constante
(Goldfeld-Quandt).\nTiene homocedasticidad.\n")
} else {
  cat("Se rechaza H₀: La varianza de los errores no es constante
(Goldfeld-Quandt).\nNo tiene homocedasticidad.\n")
}
```

```
## No se rechaza H₀: La varianza de los errores es constante (Goldfeld-
Quandt).
## Tiene homocedasticidad.
```

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

```r
# Graficar los datos y la línea de regresión
plot(cars$speed, cars$dist,
     main = "Regresión Lineal: Distancia en función de la Velocidad",
     xlab = "Velocidad (speed)",
     ylab = "Distancia (dist)",
     pch = 19, col = "blue")
abline(modelo, col = "red", lwd = 2)
legend("topleft", legend = paste("Ecuación: dist =",
round(coef(modelo)[1], 2), "+", round(coef(modelo)[2], 2), "* speed"),
       col = "red", lwd = 2)
```



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

El modelo de regresión lineal ajustado es adecuado y válido para describir la relación entre la velocidad y la distancia de frenado de los autos. La alta significancia de los coeficientes y del modelo en su conjunto, junto con un R^2 razonablemente alto, sugieren que el modelo capta bien la relación subyacente entre las variables. Sin embargo, la normalidad de los residuos es un punto ligeramente débil y podría requerir una revisión adicional si se busca un ajuste más riguroso. En general, el modelo es válido y adecuado para la predicción y análisis de la relación velocidad-distancia.

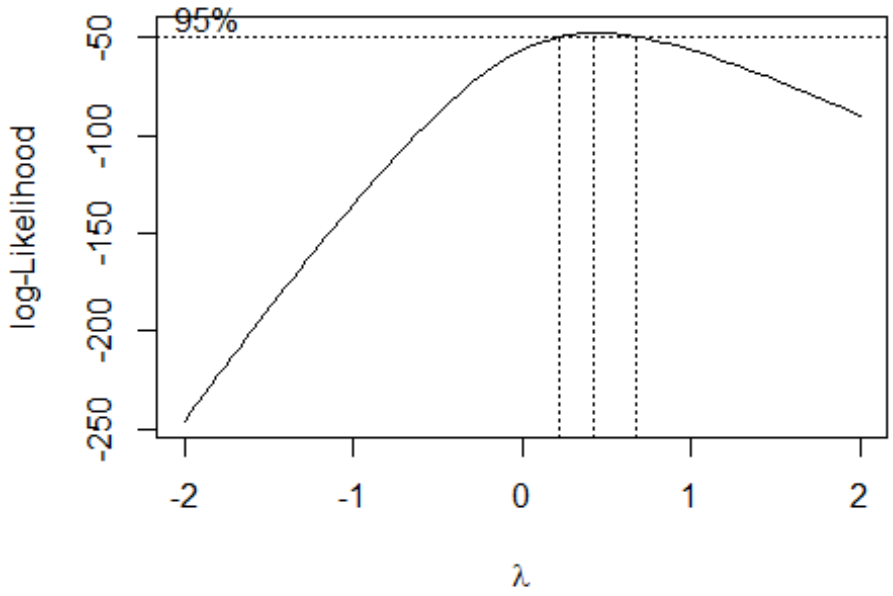
 # Parte 3: Regresión no lineal

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).

1. 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:

```
# Cargar la librería necesaria
library(MASS)
```

```r
# Ajustar el modelo lineal con distancia como variable dependiente y
velocidad como independiente
modelo <- lm(dist ~ speed, data = cars)

# Aplicar la transformación de Box-Cox
boxcox_result <- boxcox(modelo, lambda = seq(-2, 2, by = 0.1))
```




```r
# Encontrar el valor óptimo de lambda
lambda_optimo <- boxcox_result$x[which.max(boxcox_result$y)]
cat("El valor óptimo de lambda para la transformación Box-Cox es:",
lambda_optimo, "\n")

## El valor óptimo de lambda para la transformación Box-Cox es: 0.4242424
```

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

% Transformación Aproximada de Box-Cox $X_1 = \sqrt{X+1}$

% Transformación Exacta de Box-Cox $X_2 = \frac{(X+1)^{0.4242424}-1}{0.4242424}$

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

1. Compara las medidas: sesgo y curtosis.

```r
# Cargar los datos y librerías necesarias
data(cars)
library(e1071)  # Para funciones de skewness y kurtosis

# Transformaciones de Box-Cox
X_original <- cars$dist  # Distancia original

# Transformación aproximada de Box-Cox
X1 <- sqrt(X_original + 1)

# Transformación exacta de Box-Cox
lambda <- 0.4242424
X2 <- ((X_original + 1)^lambda - 1) / lambda

# Calcular sesgo y curtosis
sesgo_original <- skewness(X_original)
curtosis_original <- kurtosis(X_original)

sesgo_X1 <- skewness(X1)
curtosis_X1 <- kurtosis(X1)

sesgo_X2 <- skewness(X2)
curtosis_X2 <- kurtosis(X2)

# Imprimir resultados
cat("Medidas de Sesgo y Curtosis:\n")
```

```
## Medidas de Sesgo y Curtosis:
```

```r
cat("Original:\n")
```

```
## Original:
```

```r
cat("  - Sesgo:", sesgo_original, "\n")
```

```
##   - Sesgo: 0.7591268
```

```r
cat("  - Curtosis:", curtosis_original, "\n\n")
```

```
##   - Curtosis: 0.1193971
```

```r
cat("Transformación Aproximada (X1 = sqrt(X + 1)):\n")
```

```
## Transformación Aproximada (X1 = sqrt(X + 1)):
```

```r
cat("  - Sesgo:", sesgo_X1, "\n")
```

```
##   - Sesgo: 0.02430858
```

```r
cat("  - Curtosis:", curtosis_X1, "\n\n")
```

```
##   - Curtosis: -0.3647174
```

```r
cat("Transformación Exacta (X2 = ((X + 1)^lambda - 1) / lambda):\n")
```

```
## Transformación Exacta (X2 = ((X + 1)^lambda - 1) / lambda):
```

```r
cat("  - Sesgo:", sesgo_X2, "\n")
```

```
##   - Sesgo: -0.1126697
```

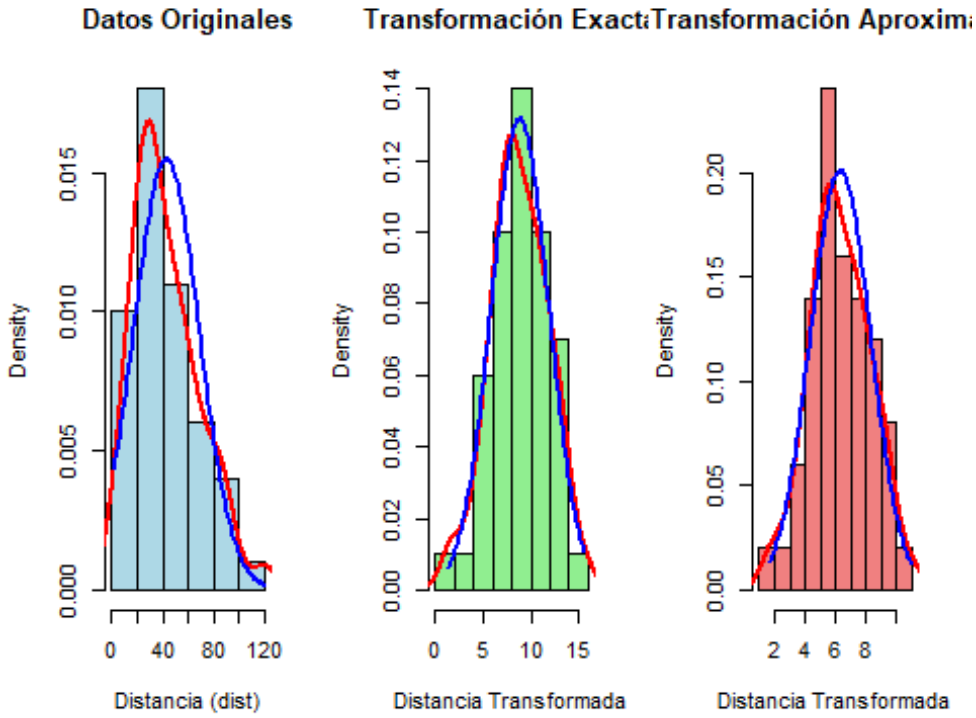
```r
cat("  - Curtosis:", curtosis_X2, "\n")
```

```
##   - Curtosis: -0.2758653
```

2. Obten el histograma de los 2 modelos obtenidos (exacto y aproximado) y los datos originales.

```r
# Cargar los datos y librerías necesarias
data(cars)
library(MASS)
library(e1071)  # Para skewness y kurtosis

# Datos originales
X_original <- cars$dist

# Transformación exacta de Box-Cox con lambda = 0.4242424
lambda <- 0.4242424
X_exacta <- ((X_original + 1)^lambda - 1) / lambda

# Transformación aproximada de Box-Cox
X_aproximada <- sqrt(X_original + 1)

# Configuración de los gráficos en 3 paneles
par(mfrow = c(1, 3))

# Histograma de los datos originales
hist(X_original, freq = FALSE, main = "Datos Originales",
     xlab = "Distancia (dist)", col = "lightblue", border = "black")
lines(density(X_original), col = "red", lwd = 2)
curve(dnorm(x, mean = mean(X_original), sd = sd(X_original)),
      from = min(X_original), to = max(X_original), add = TRUE, col =
"blue", lwd = 2)

# Histograma de la transformación exacta de Box-Cox
hist(X_exacta, freq = FALSE, main = "Transformación Exacta",
     xlab = "Distancia Transformada", col = "lightgreen", border =
"black")
lines(density(X_exacta), col = "red", lwd = 2)
curve(dnorm(x, mean = mean(X_exacta), sd = sd(X_exacta)),
      from = min(X_exacta), to = max(X_exacta), add = TRUE, col = "blue",
lwd = 2)

# Histograma de la transformación aproximada de Box-Cox
```

```r
hist(X_aproximada, freq = FALSE, main = "Transformación Aproximada",
     xlab = "Distancia Transformada", col = "lightcoral", border =
"black")
lines(density(X_aproximada), col = "red", lwd = 2)
curve(dnorm(x, mean = mean(X_aproximada), sd = sd(X_aproximada)),
      from = min(X_aproximada), to = max(X_aproximada), add = TRUE, col =
"blue", lwd = 2)
```




```r
# Restaurar la configuración de gráficos original
par(mfrow = c(1, 1))
```

3. Realiza algunas pruebas de normalidad para los datos transformados.

```r
# Cargar los datos y librerías necesarias
data(cars)
library(MASS)
library(e1071)  # Para skewness y kurtosis
library(tseries) # Para la prueba de Jarque-Bera

# Datos originales
X_original <- cars$dist

# Transformación exacta de Box-Cox con lambda = 0.4242424
lambda <- 0.4242424
X_exacta <- ((X_original + 1)^lambda - 1) / lambda

# Transformación aproximada de Box-Cox
```

```r
X_aproximada <- sqrt(X_original + 1)

# Realizar pruebas de normalidad para los datos originales
shapiro_original <- shapiro.test(X_original)
jarque_bera_original <- jarque.bera.test(X_original)

# Realizar pruebas de normalidad para la transformación exacta
shapiro_exacta <- shapiro.test(X_exacta)
jarque_bera_exacta <- jarque.bera.test(X_exacta)

# Realizar pruebas de normalidad para la transformación aproximada
shapiro_aproximada <- shapiro.test(X_aproximada)
jarque_bera_aproximada <- jarque.bera.test(X_aproximada)

# Mostrar resultados y comentarios sobre la normalidad
cat("Pruebas de Normalidad para los Datos Originales:\n")
```

```
## Pruebas de Normalidad para los Datos Originales:
```

```r
print(shapiro_original)
```

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

```r
if (shapiro_original$p.value < 0.05) {
  cat(" - Shapiro-Wilk: NO cumple con la normalidad (p-value:",
shapiro_original$p.value, ").\n")
} else {
  cat(" - Shapiro-Wilk: Cumple con la normalidad (p-value:",
shapiro_original$p.value, ").\n")
}
```

```
##  - Shapiro-Wilk: NO cumple con la normalidad (p-value: 0.03909968 ).
```

```r
print(jarque_bera_original)
```

```
##
##  Jarque Bera Test
##
## data:  X_original
## X-squared = 5.2305, df = 2, p-value = 0.07315
```

```r
if (jarque_bera_original$p.value < 0.05) {
  cat(" - Jarque-Bera: NO cumple con la normalidad (p-value:",
jarque_bera_original$p.value, ").\n")
} else {
  cat(" - Jarque-Bera: Cumple con la normalidad (p-value:",
```

```
jarque_bera_original$p.value, ").\n")
}

##  - Jarque-Bera: Cumple con la normalidad (p-value: 0.07314987 ).

cat("\nPruebas de Normalidad para la Transformación Exacta:\n")

##
## Pruebas de Normalidad para la Transformación Exacta:

print(shapiro_exacta)

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

if (shapiro_exacta$p.value < 0.05) {
  cat(" - Shapiro-Wilk: NO cumple con la normalidad (p-value:",
shapiro_exacta$p.value, ").\n")
} else {
  cat(" - Shapiro-Wilk: Cumple con la normalidad (p-value:",
shapiro_exacta$p.value, ").\n")
}

##  - Shapiro-Wilk: Cumple con la normalidad (p-value: 0.9884953 ).

print(jarque_bera_exacta)

##
##  Jarque Bera Test
##
## data:  X_exacta
## X-squared = 0.16812, df = 2, p-value = 0.9194

if (jarque_bera_exacta$p.value < 0.05) {
  cat(" - Jarque-Bera: NO cumple con la normalidad (p-value:",
jarque_bera_exacta$p.value, ").\n")
} else {
  cat(" - Jarque-Bera: Cumple con la normalidad (p-value:",
jarque_bera_exacta$p.value, ").\n")
}

##  - Jarque-Bera: Cumple con la normalidad (p-value: 0.9193773 ).

cat("\nPruebas de Normalidad para la Transformación Aproximada:\n")

##
## Pruebas de Normalidad para la Transformación Aproximada:

print(shapiro_aproximada)
```

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

```
if (shapiro_aproximada$p.value < 0.05) {
  cat(" - Shapiro-Wilk: NO cumple con la normalidad (p-value:",
shapiro_aproximada$p.value, ").\n")
} else {
  cat(" - Shapiro-Wilk: Cumple con la normalidad (p-value:",
shapiro_aproximada$p.value, ").\n")
}
```

```
##  - Shapiro-Wilk: Cumple con la normalidad (p-value: 0.993603 ).
```

```
print(jarque_bera_aproximada)
```

```
##
##  Jarque Bera Test
##
## data:  X_aproximada
## X-squared = 0.14183, df = 2, p-value = 0.9315
```

```
if (jarque_bera_aproximada$p.value < 0.05) {
  cat(" - Jarque-Bera: NO cumple con la normalidad (p-value:",
jarque_bera_aproximada$p.value, ").\n")
} else {
  cat(" - Jarque-Bera: Cumple con la normalidad (p-value:",
jarque_bera_aproximada$p.value, ").\n")
}
```

```
##  - Jarque-Bera: Cumple con la normalidad (p-value: 0.9315428 ).
```

4. 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 lambda para tus nuevos datos.

Dado que las transformaciones exacta y aproximada cumplen con la normalidad según las pruebas realizadas (Shapiro-Wilk y Jarque-Bera), no sería necesario aplicar corrección de datos por anomalías como detección y eliminación de datos atípicos o corrección de ceros anómalos.

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.

Ambas transformación redujeron significativamente el sesgo y la curtosis de los datos sin embargo, la mejor tranformación fue la aproximada ya que si bien tiene más

curtosis que la transformación exacta, tiene menos sesgo que la tranformación exacta y a aparte

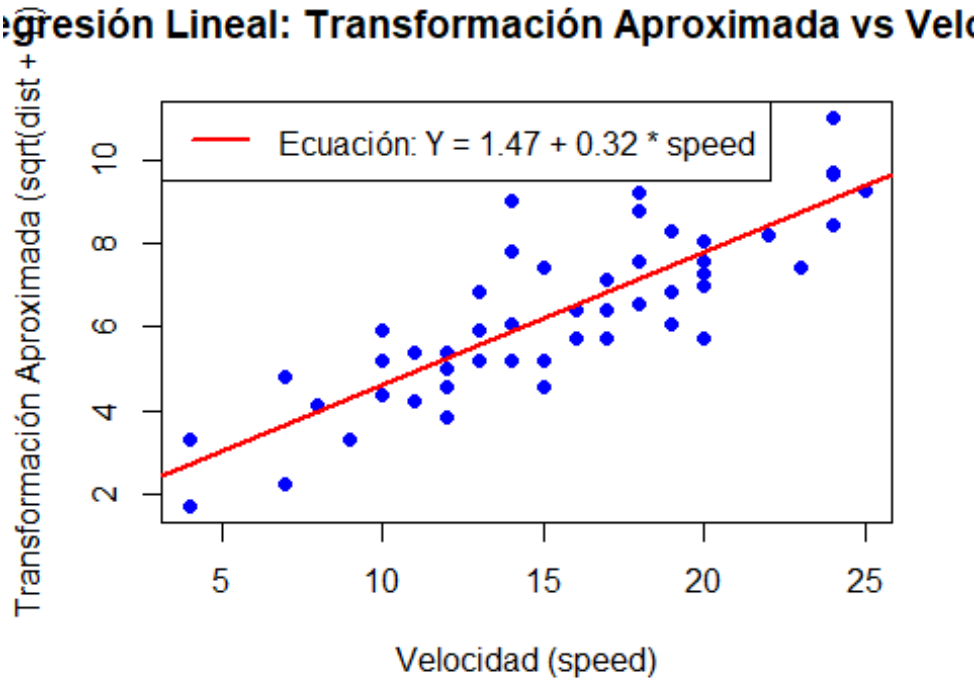
3. Con la mejor transformación (punto 2), realiza la regresión lineal simple entre la mejor transformación (exacta o aproximada) y la variable velocidad:

```
# Cargar los datos necesarios
data(cars)

# Datos originales
X_original <- cars$dist
speed <- cars$speed

# Aplicar la mejor transformación aproximada: sqrt(X + 1)
X_aproximada <- sqrt(X_original + 1)

# Ajustar el modelo de regresión lineal con la transformación aproximada
modelo_aproximado <- lm(X_aproximada ~ speed)

# Mostrar los resultados del modelo
summary(modelo_aproximado)
```

```
##
## Call:
## lm(formula = X_aproximada ~ speed)
##
## 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 **
## speed        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
```

```
# Graficar los datos transformados y la línea de regresión
plot(speed, X_aproximada,
     main = "Regresión Lineal: Transformación Aproximada vs Velocidad",
     xlab = "Velocidad (speed)",
     ylab = "Transformación Aproximada (sqrt(dist + 1))",
     pch = 19, col = "blue")
abline(modelo_aproximado, col = "red", lwd = 2)
legend("topleft", legend = paste("Ecuación: Y =",
round(coef(modelo_aproximado)[1], 2),
```

```
                                    "+", round(coef(modelo_aproximado)[2],
2), "* speed"),
       col = "red", lwd = 2)
```



1. Escribe el modelo lineal para la transformación.

```
# Cargar los datos necesarios
data(cars)

# Datos originales
X_original <- cars$dist
speed <- cars$speed

# Aplicar la mejor transformación aproximada: sqrt(X + 1)
X_aproximada <- sqrt(X_original + 1)

# Ajustar el modelo de regresión lineal con la transformación aproximada
modelo_aproximado <- lm(X_aproximada ~ speed)

# Mostrar el resumen del modelo para obtener los coeficientes
summary_modelo <- summary(modelo_aproximado)
print(summary_modelo)

##
## Call:
## lm(formula = X_aproximada ~ speed)
##
```

```
## 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 **
## speed        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
```

```
# Extraer los coeficientes del modelo
intercepto <- coef(modelo_aproximado)[1]
pendiente <- coef(modelo_aproximado)[2]

# Mostrar la ecuación del modelo lineal
cat("\nEl modelo lineal para la transformación aproximada es:\n")
```

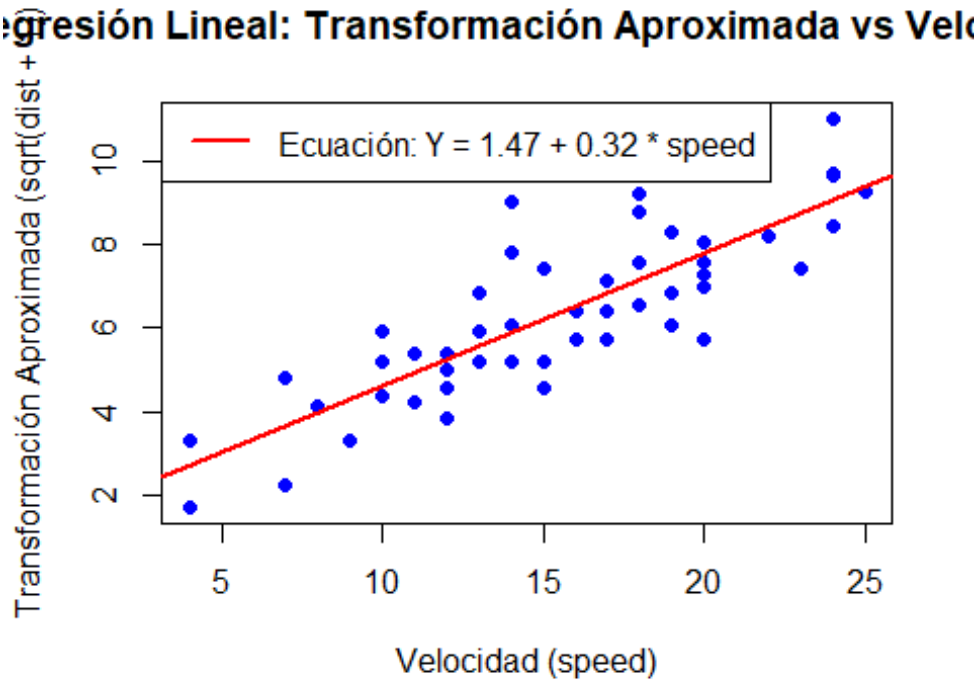
```
##
## El modelo lineal para la transformación aproximada es:
```

```
cat("sqrt(dist + 1) =", round(intercepto, 4), "+", round(pendiente, 4),
"* speed\n")
```

```
## sqrt(dist + 1) = 1.4668 + 0.316 * speed
```

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

```
# Graficar los datos transformados y la línea de regresión
plot(speed, X_aproximada,
     main = "Regresión Lineal: Transformación Aproximada vs Velocidad",
     xlab = "Velocidad (speed)",
     ylab = "Transformación Aproximada (sqrt(dist + 1))",
     pch = 19, col = "blue")
abline(modelo_aproximado, col = "red", lwd = 2)
legend("topleft", legend = paste("Ecuación: Y =", round(intercepto, 2),
                                 "+", round(pendiente, 2), "* speed"),
       col = "red", lwd = 2)
```



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

```r
# Cargar los datos necesarios
data(cars)

# Datos originales
X_original <- cars$dist
speed <- cars$speed

# Aplicar la mejor transformación aproximada: sqrt(dist + 1)
X_aproximada <- sqrt(X_original + 1)

# Ajustar el modelo de regresión lineal con la transformación aproximada
modelo_aproximado <- lm(X_aproximada ~ speed)

# Mostrar el resumen del modelo para evaluar la significancia
summary_modelo <- summary(modelo_aproximado)

# Extraer información de significancia individual y conjunta
coeficientes <- summary_modelo$coefficients
r_squared <- summary_modelo$r.squared
p_value_conjunto <- summary_modelo$fstatistic[3]

# Imprimir la significancia individual de los coeficientes
cat("\nSignificancia Individual de los Coeficientes:\n")
```

```
##
## Significancia Individual de los Coeficientes:

cat("Intercepto (p-value):", coeficientes[1, 4], "\n")

## Intercepto (p-value): 0.003409087

cat("Velocidad (speed) (p-value):", coeficientes[2, 4], "\n")

## Velocidad (speed) (p-value): 1.931291e-14

# Imprimir la significancia conjunta del modelo
cat("\nSignificancia Conjunta del Modelo (Prueba F):\n")

##
## Significancia Conjunta del Modelo (Prueba F):

cat("p-value:", pf(summary_modelo$fstatistic[1],
summary_modelo$fstatistic[2], summary_modelo$fstatistic[3], lower.tail =
FALSE), "\n")

## p-value: 1.931291e-14

# Imprimir el coeficiente de determinación (R^2)
cat("\nCoeficiente de Determinación (R^2):", r_squared, "\n")

##
## Coeficiente de Determinación (R^2): 0.7083851

# Calcular el coeficiente de correlación entre la transformación
aproximada y la velocidad
coef_correlacion <- cor(X_aproximada, speed)
cat("\nCoeficiente de Correlación:", coef_correlacion, "\n")

##
## Coeficiente de Correlación: 0.8416562
```

4. 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.

1. Residuos con media cero

% Hipótesis de Media Diferente de Cero (t de Student)

$$\begin{cases} H_0: \mu = 0 & \text{(El promedio de los residuos es igual a cero)} \\ H_1: \mu \neq 0 & \text{(El promedio de los residuos es diferente de cero)} \end{cases}$$

2. Normalidad de los residuos

% Hipótesis de Normalidad (Anderson-Darling)

$$\begin{cases} H_0: \text{Los residuos siguen una distribución normal.} \\ H_1: \text{Los residuos no siguen una distribución normal.} \end{cases}$$

3. Homocedasticidad, linealidad e independencia

% Hipótesis de Autocorrelación (Durbin-Watson y Breusch-Godfrey)

$$\begin{cases} H_0: \text{Los errores no están autocorrelacionados (independencia).} \\ H_1: \text{Los errores están autocorrelacionados.} \end{cases}$$

% Hipótesis de Homocedasticidad (Breusch-Pagan y Goldfeld-Quandt)

$$\begin{cases} H_0: \text{La varianza de los errores es constante (homocedasticidad).} \\ H_1: \text{La varianza de los errores no es constante (heterocedasticidad).} \end{cases}$$

```
# Realiza la prueba t para los residuos de los modelos sin y con
interacción
cat("\nModelo Transformación Aproximada:\n")

##
## Modelo Transformación Aproximada:

resultado_t_modelo <- t.test(modelo_aproximado$residuals)

# Mostrar los resultados de las pruebas t
cat("Resultados de la prueba t de Student para los residuos:\n")

## Resultados de la prueba t de Student para los residuos:

# Verificar si el promedio de los residuos es significativamente
diferente de cero
if (resultado_t_modelo$p.value > 0.05) {
  cat("\nEl promedio de los residuos del Modelo Transformación Aproximada
no es significativamente diferente de cero.\n")
} else {
  cat("\nEl promedio de los residuos del Modelo Transformación Aproximada
es significativamente diferente de cero.\n")
}

##
## El promedio de los residuos del Modelo Transformación Aproximada no es
significativamente diferente de cero.

# Cargar la librería para la prueba Anderson-Darling
library(nortest)

# Realizar la prueba de Jarque Bera sobre los residuos de los modelos
resultado_modelo <- ad.test(modelo_aproximado$residuals)

# Mostrar los resultados de las pruebas
cat("\nResultados de la prueba de normalidad Anderson-Darling:\n")
```

```
##
## Resultados de la prueba de normalidad Anderson-Darling:

# Verificar normalidad en base al valor p
if (resultado_modelo$p.value > 0.05) {
  cat("\nSe tiene normalidad en el Modelo Transformación Aproximada.\n")
} else {
  cat("\nNo se tiene normalidad en el Modelo Transformación
Aproximada.\n")
}

##
## Se tiene normalidad en el Modelo Transformación Aproximada.

# Cargar la librería necesaria
library(lmtest)

# Modelo 1: Sin Interacción
dw_modelo <- dwtest(modelo_aproximado)
bg_modelo <- bgtest(modelo_aproximado)

# Verificar autocorrelación en base al valor p
if (dw_modelo$p.value > 0.05) {
  cat("\nNo se rechaza H₀: Los errores no están autocorrelacionados
(Durbin-Watson).\nTiene independencia.\n")
} else {
  cat("\nSe rechaza H₀: Los errores están autocorrelacionados (Durbin-
Watson).\nNo tiene independencia.\n")
}

##
## No se rechaza H₀: Los errores no están autocorrelacionados (Durbin-
Watson).
## Tiene independencia.

if (bg_modelo$p.value > 0.05) {
  cat("No se rechaza H₀: Los errores no están autocorrelacionados
(Breusch-Godfrey).\nTiene independencia.\n")
} else {
  cat("Se rechaza H₀: Los errores están autocorrelacionados (Breusch-
Godfrey).\nNo tiene independencia.\n")
}

## No se rechaza H₀: Los errores no están autocorrelacionados (Breusch-
Godfrey).
## Tiene independencia.

# Modelo 2: Con Interacción
bp_modelo <- bptest(modelo_aproximado)
gq_modelo <- gqtest(modelo_aproximado)
```

```r
# Verificar homocedasticidad en base al valor p
if (bp_modelo$p.value > 0.05) {
  cat("\nNo se rechaza H₀: La varianza de los errores es constante
(Breusch-Pagan).\nTiene homocedasticidad.\n")
} else {
  cat("\nSe rechaza H₀: La varianza de los errores no es constante
(Breusch-Pagan).\nNo tiene homocedasticidad.\n")
}
```

```
##
## No se rechaza H₀: La varianza de los errores es constante (Breusch-
Pagan).
## Tiene homocedasticidad.
```

```r
if (gq_modelo$p.value > 0.05) {
  cat("No se rechaza H₀: La varianza de los errores es constante
(Goldfeld-Quandt).\nTiene homocedasticidad.\n")
} else {
  cat("Se rechaza H₀: La varianza de los errores no es constante
(Goldfeld-Quandt).\nNo tiene homocedasticidad.\n")
}
```

```
## No se rechaza H₀: La varianza de los errores es constante (Goldfeld-
Quandt).
## Tiene homocedasticidad.
```

```r
# Calcular residuos estandarizados
residuos_estandarizados <- rstandard(modelo_aproximado)

# Calcular la distancia de Cook
cooks_distance <- cooks.distance(modelo_aproximado)

# Calcular leverage (hat values)
hat_values <- hatvalues(modelo_aproximado)

# Umbrales típicos para identificar atípicos e influyentes
threshold_residuos <- 2  # Residuos estandarizados mayores a 2 o menores
a -2 pueden considerarse atípicos
threshold_cooks <- 4 / length(X_original)  # Umbral común para la
distancia de Cook
threshold_leverage <- 2 * mean(hat_values)  # Puntos con leverage mayor a
2 veces el promedio

# Identificar puntos atípicos según los umbrales
puntos_atipicos_residuos <- which(abs(residuos_estandarizados) >
threshold_residuos)
puntos_influyentes_cooks <- which(cooks_distance > threshold_cooks)
puntos_altos_leverage <- which(hat_values > threshold_leverage)
```

```r
# Mostrar los índices y valores de los puntos atípicos e influyentes
cat("Puntos Atípicos por Residuos Estandarizados (mayores a ±2):\n")
```

```
## Puntos Atípicos por Residuos Estandarizados (mayores a ±2):
```

```r
print(puntos_atipicos_residuos)
```

```
## 23
## 23
```

```r
cat("\nPuntos Influyentes por Distancia de Cook (mayores a",
round(threshold_cooks, 4), "):\n")
```

```
##
## Puntos Influyentes por Distancia de Cook (mayores a 0.08 ):
```

```r
print(puntos_influyentes_cooks)
```

```
## 23 49
## 23 49
```

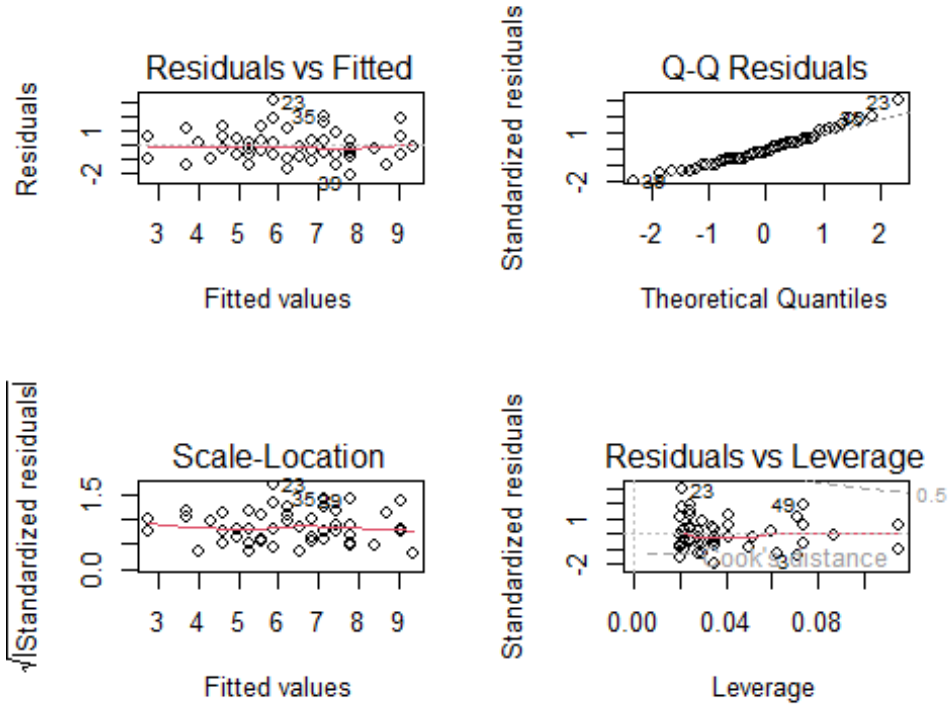
```r
cat("\nPuntos con Alto Leverage (mayores a", round(threshold_leverage,
4), "):\n")
```

```
##
## Puntos con Alto Leverage (mayores a 0.08 ):
```

```r
print(puntos_altos_leverage)
```

```
##  1  2 50
##  1  2 50
```

```r
# Graficar los gráficos diagnósticos del modelo
par(mfrow = c(2, 2)) # Organizar los gráficos en una cuadrícula 2x2
plot(modelo_aproximado)
```




```
par(mfrow = c(1, 1)) # Restaurar la configuración de gráficos
```

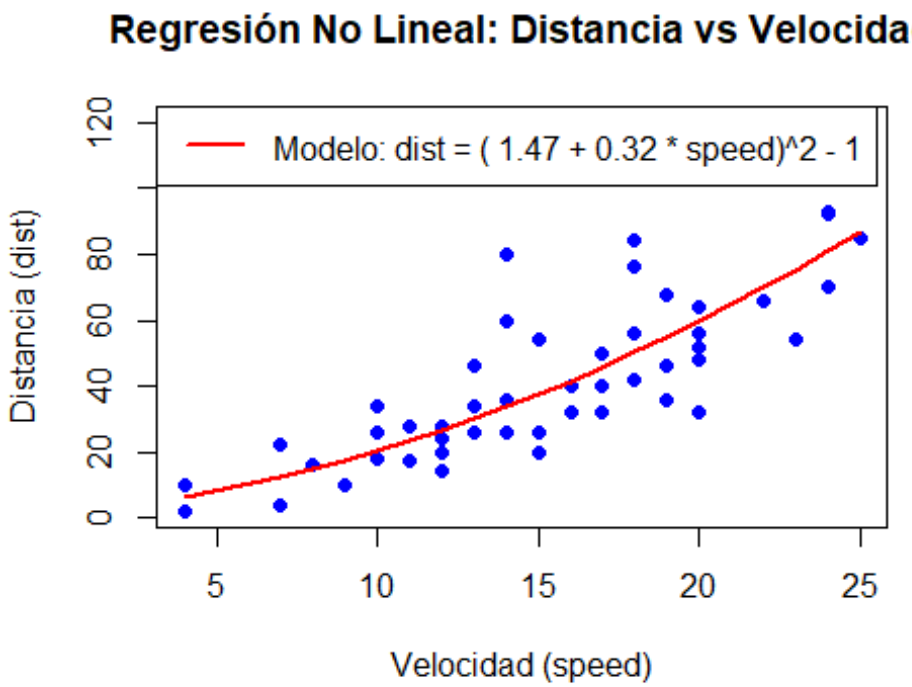
5. Despeja la distancia del modelo lineal obtenido entre la transformación y la velocidad. Obtendrás el modelo no lineal que relaciona la distancia con la velocidad directamente (y no con su transformación).

```
# Extraer los coeficientes del modelo lineal
intercepto <- coef(modelo_aproximado)[1]
pendiente <- coef(modelo_aproximado)[2]

# Mostrar la ecuación del modelo lineal
cat("Modelo lineal obtenido:\n")

## Modelo lineal obtenido:

cat("sqrt(dist + 1) =", round(intercepto, 4), "+", round(pendiente, 4),
"* speed\n")

## sqrt(dist + 1) = 1.4668 + 0.316 * speed

# Despejar la distancia (dist) del modelo no lineal
cat("\nEl modelo no lineal que relaciona la distancia con la velocidad
es:\n")

##
## El modelo no lineal que relaciona la distancia con la velocidad es:
```

```r
cat("dist = (", round(intercepto, 4), "+", round(pendiente, 4), "*
speed)^2 - 1\n")

## dist = ( 1.4668 + 0.316 * speed)^2 - 1
```

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

```r
# Extraer los coeficientes del modelo lineal
intercepto <- coef(modelo_aproximado)[1]
pendiente <- coef(modelo_aproximado)[2]

# Calcular el modelo no lineal: dist = (intercepto + pendiente * speed)^2
- 1
dist_predicha <- (intercepto + pendiente * speed)^2 - 1

# Graficar los datos originales y el modelo no lineal ajustado
plot(speed, X_original,
     main = "Regresión No Lineal: Distancia vs Velocidad",
     xlab = "Velocidad (speed)",
     ylab = "Distancia (dist)",
     pch = 19, col = "blue")
lines(speed, dist_predicha, col = "red", lwd = 2)
legend("topleft", legend = paste("Modelo: dist = (", round(intercepto,
2), "+",
                                 round(pendiente, 2), "* speed)^2 - 1"),
       col = "red", lwd = 2)
```



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

- Significancia: El modelo es altamente significativo a nivel individual (coeficientes) y conjunto (prueba F), con p-valores extremadamente bajos, confirmando que la velocidad es un predictor importante de la distancia.
- Ajuste del Modelo: El coeficiente de determinación (R^2=0.7084) indica que el modelo explica el 71% de la variabilidad en la distancia, mostrando un ajuste razonablemente bueno.
- Correlación: Existe una fuerte correlación positiva (r=0.8417 r=0.8417) entre la transformación de la distancia y la velocidad, lo que refuerza la relación lineal.
- Supuestos: El modelo cumple con todos los supuestos lo cual lo hace un modelo válido.
- Conclusión: El modelo es adecuado y válido para describir la relación entre velocidad y distancia, aunque la falta de normalidad en los residuos sugiere que podría mejorarse si se requiere una alta precisión en las inferencias.

Parte 4

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.

El mejor modelo es el del Punto 2 ya que en sí son bastante similares tiene sus coeficientes y modelos de manera significativa. Las únicas diferencias es que en el modelo del Punto 2 mejora el R^2 de 64% a 70%, además de cumplir con todos los supuestos incluyendo la normalidad en los residuos que no se había cumplido en el Punto 1 por lo que es mejor el modelo con la trasnformación.

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

- Datos Atípicos: Los valores extremos pueden distorsionar los resultados y afectar la precisión del modelo.
- Dificultad de Cálculo e Interpretación: El modelo no lineal resultante y las transformaciones complejas complican la comprensión y comunicación de los resultados.
- Sensibilidad del Modelo: Es sensible a cambios en los datos y la elección de transformaciones, lo que puede requerir ajustes continuos.
- Conclusión: El modelo es útil pero tiene limitaciones en la precisión y la interpretación debido a atípicos y la complejidad de las transformaciones.