A1 - Regresión No-Lineal

Héctor Hibran Tapia Fernández - A01661114

2024-11-30

Parte 1: Análisis de Normalidad

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

```
data <- cars
```

2. Prueba normalidad univariada de la velocidad y distancia

```
shapiro.test(data$speed)
```

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

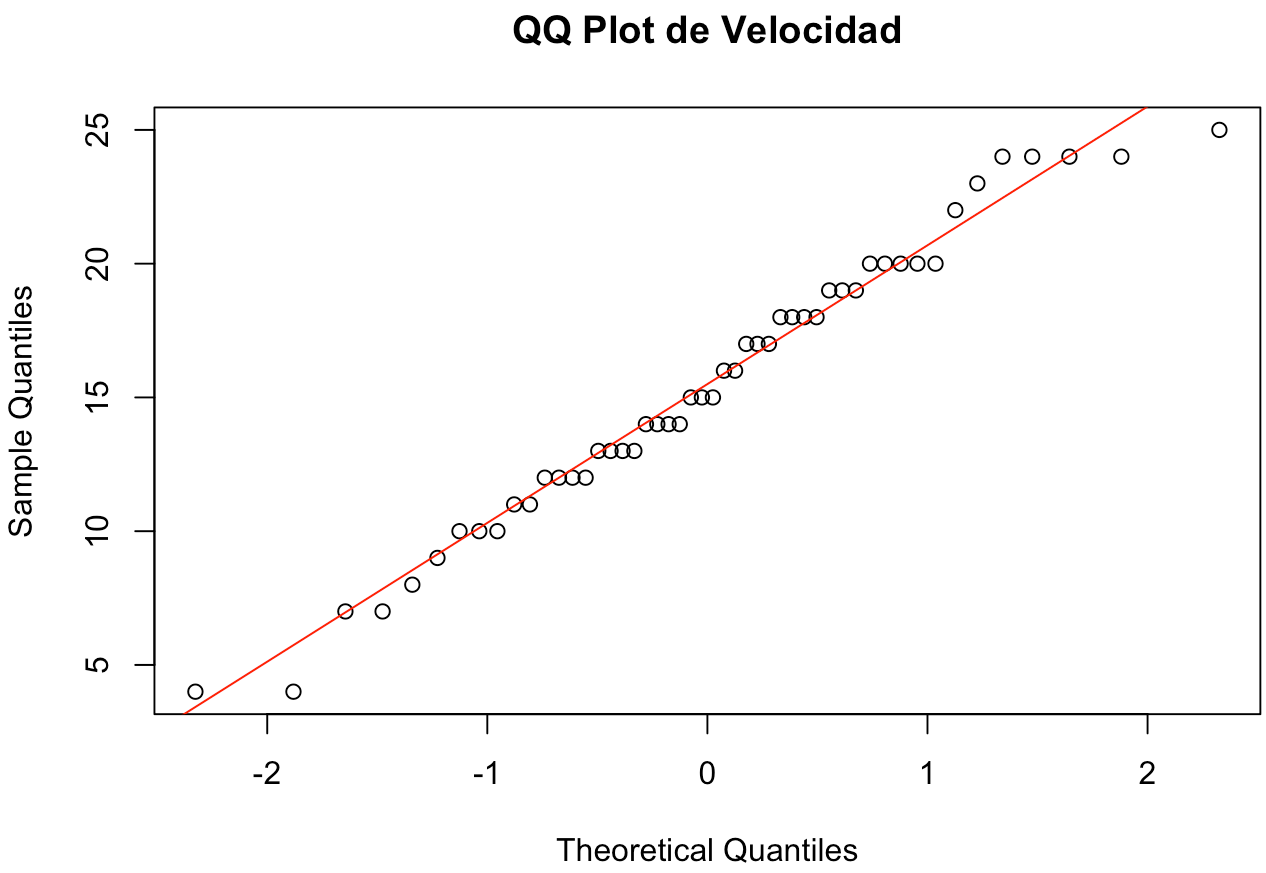
```
shapiro.test(data$dist)
```

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

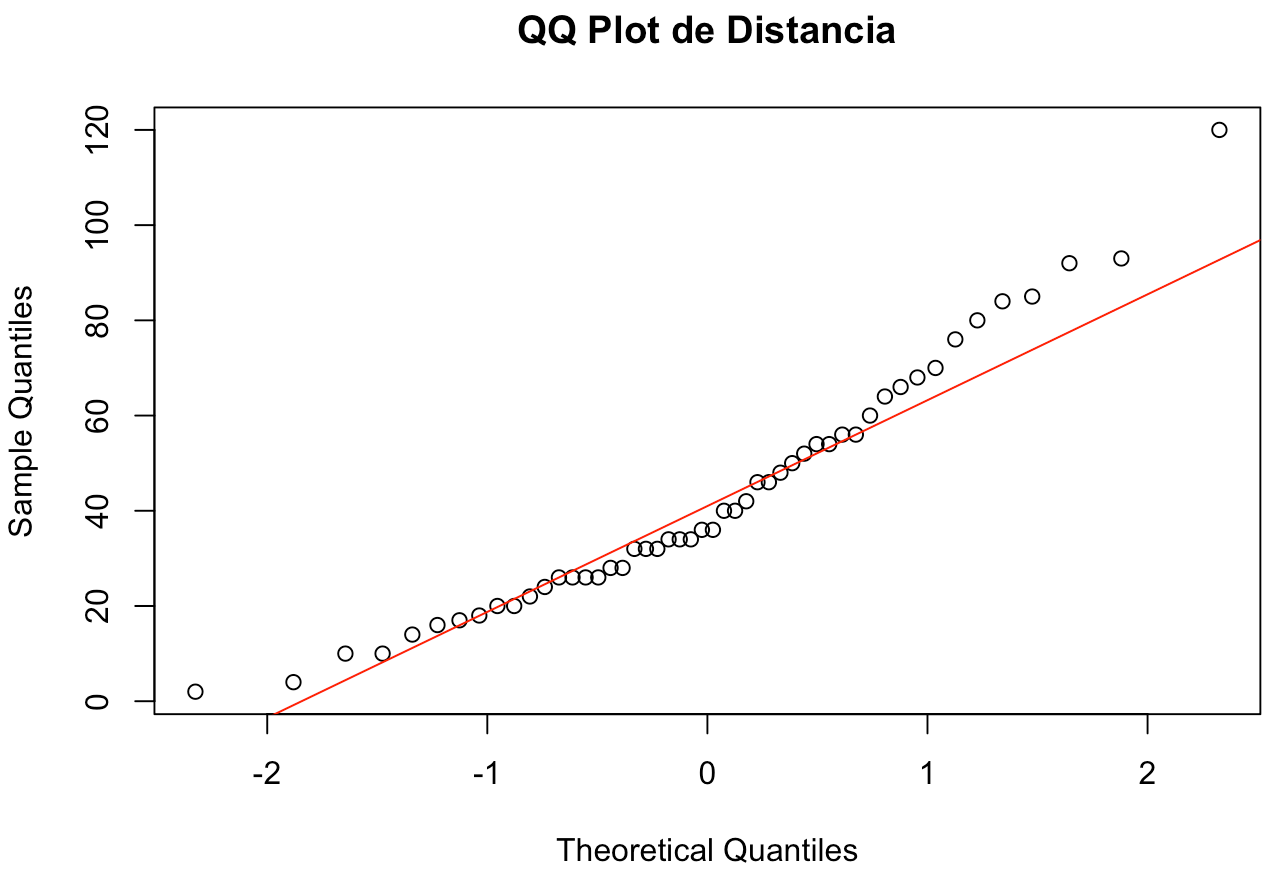
Realiza gráficos que te ayuden a identificar posibles alejamientos de normalidad:

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

```
qqnorm(data$speed, main="QQ Plot de Velocidad")
qqline(data$speed, col="red")
```

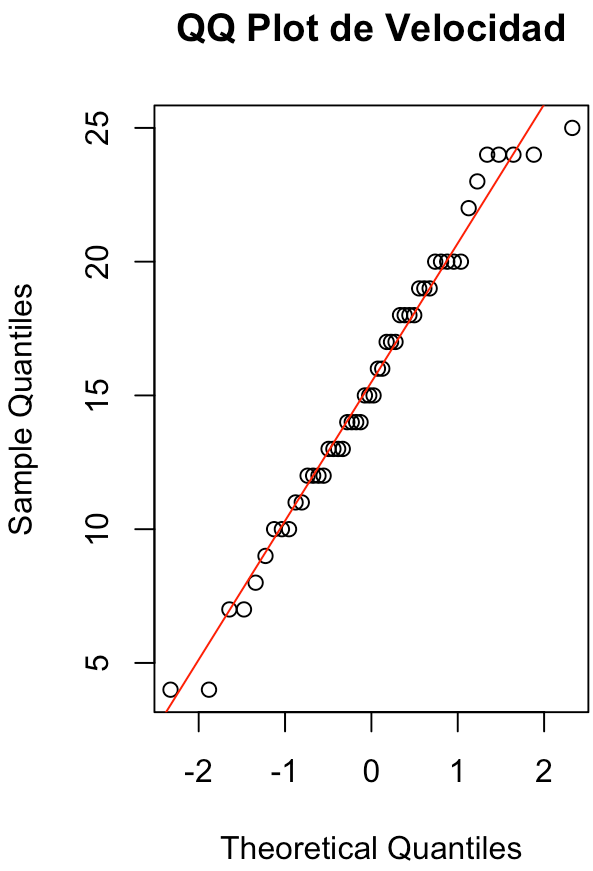


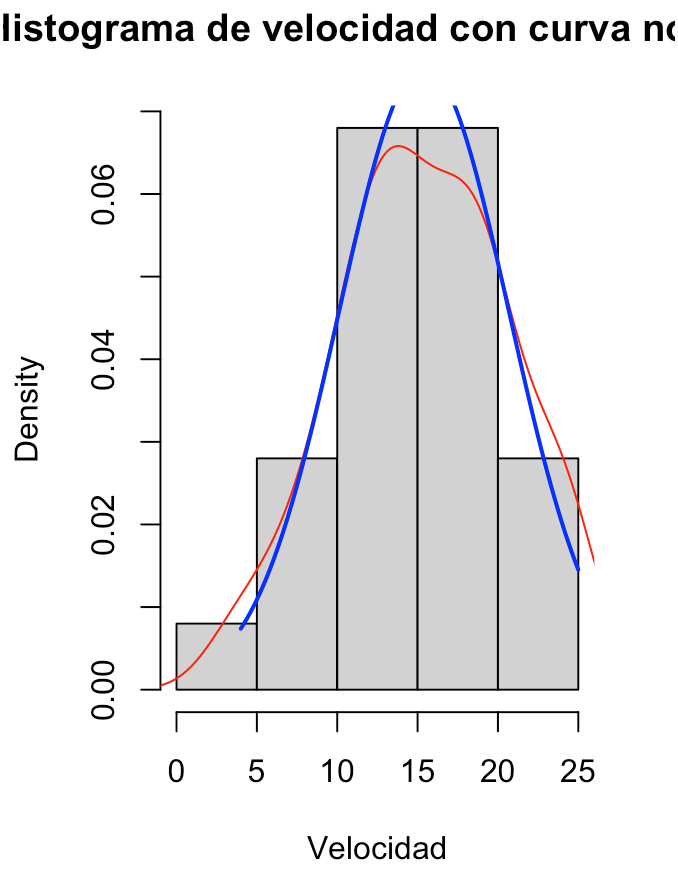

```
qqnorm(data$dist, main="QQ Plot de Distancia")
qqline(data$dist, col="red")
```



Realiza el histograma y su distribución teórica de probabilidad:

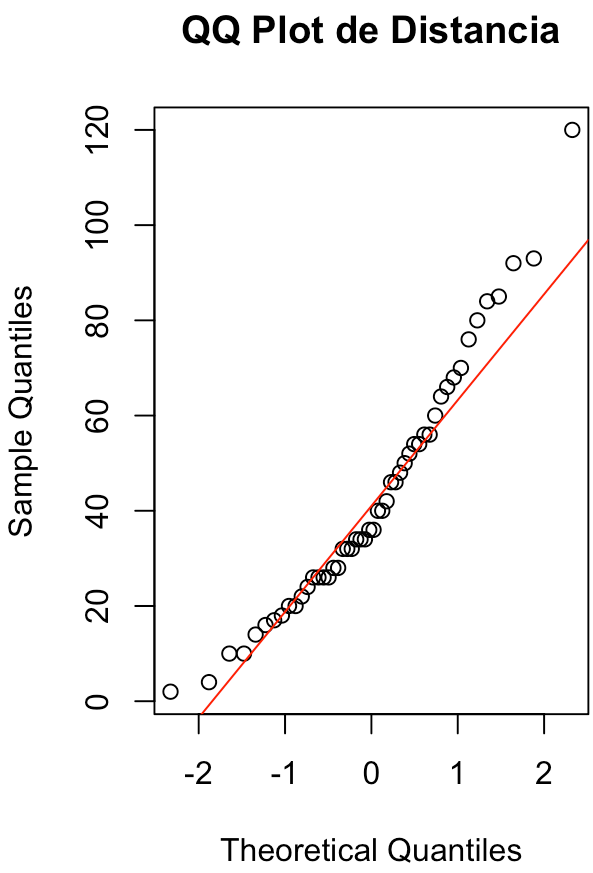
```
par(mfrow=c(1,2))
qqnorm(data$speed, main="QQ Plot de Velocidad")
qqline(data$speed, col="red")
hist(data$speed, freq=FALSE, main="Histograma de velocidad con curva normal", xlab="Velo
cidad")
lines(density(data$speed), col="red")
curve(dnorm(x, mean=mean(data$speed), sd=sd(data$speed)), from=min(data$speed), to=max(d
ata$speed), add=TRUE, col="blue", lwd=2)
```

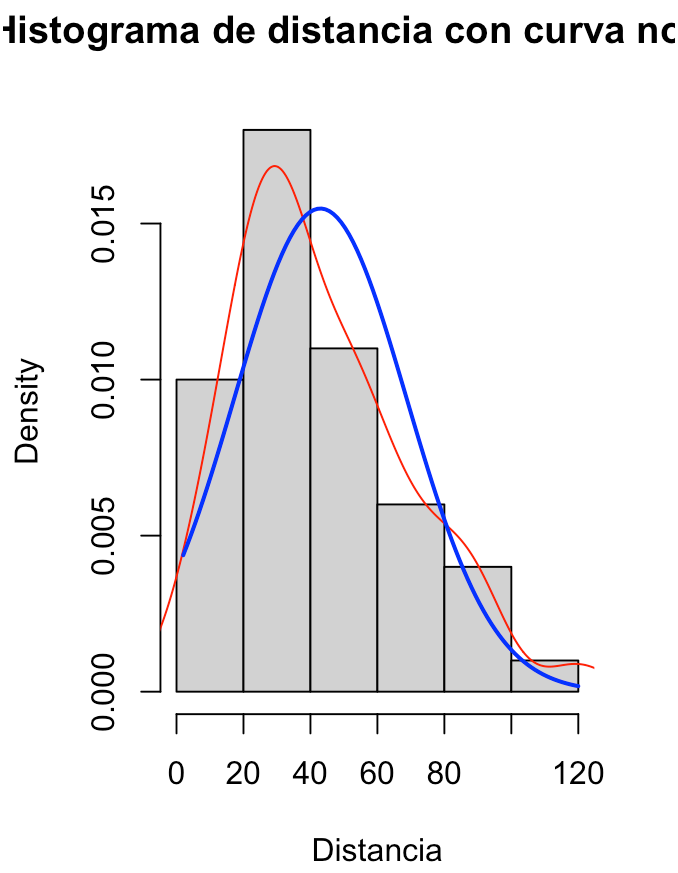





```
par(mfrow=c(1,1))

par(mfrow=c(1,2))
qqnorm(data$dist, main="QQ Plot de Distancia")
qqline(data$dist, col="red")
hist(data$dist, freq=FALSE, main="Histograma de distancia con curva normal", xlab="Dista
ncia")
lines(density(data$dist), col="red")
curve(dnorm(x, mean=mean(data$dist), sd=sd(data$dist)), from=min(data$dist), to=max(data
$dist), add=TRUE, col="blue", lwd=2)
```






```
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.

```
library(e1071)

speed_skewness = skewness(data$speed)
speed_kurtosis = kurtosis(data$speed)
dist_skewness = skewness(data$dist)
dist_kurtosis = kurtosis(data$dist)

cat("Speed:\n")
```

```
## Speed:
```

```
cat("  Skewness:", speed_skewness, "\n")
```

```
##   Skewness: -0.1105533
```

```
cat("  Kurtosis:", speed_kurtosis, "\n\n")
```

```
##   Kurtosis: -0.6730924
```

```
cat("Distance:\n")
```

```
## Distance:
```

```
cat("  Skewness:", dist_skewness, "\n")
```

```
##   Skewness: 0.7591268
```

```
cat("  Kurtosis:", dist_kurtosis, "\n")
```

```
##   Kurtosis: 0.1193971
```

Parte 2: Regresión lineal

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

Escribe el modelo lineal obtenido.

```
modelo = lm(dist ~ speed, data = cars)
coeficientes = coef(modelo)
cat("El modelo lineal es:\n")
```

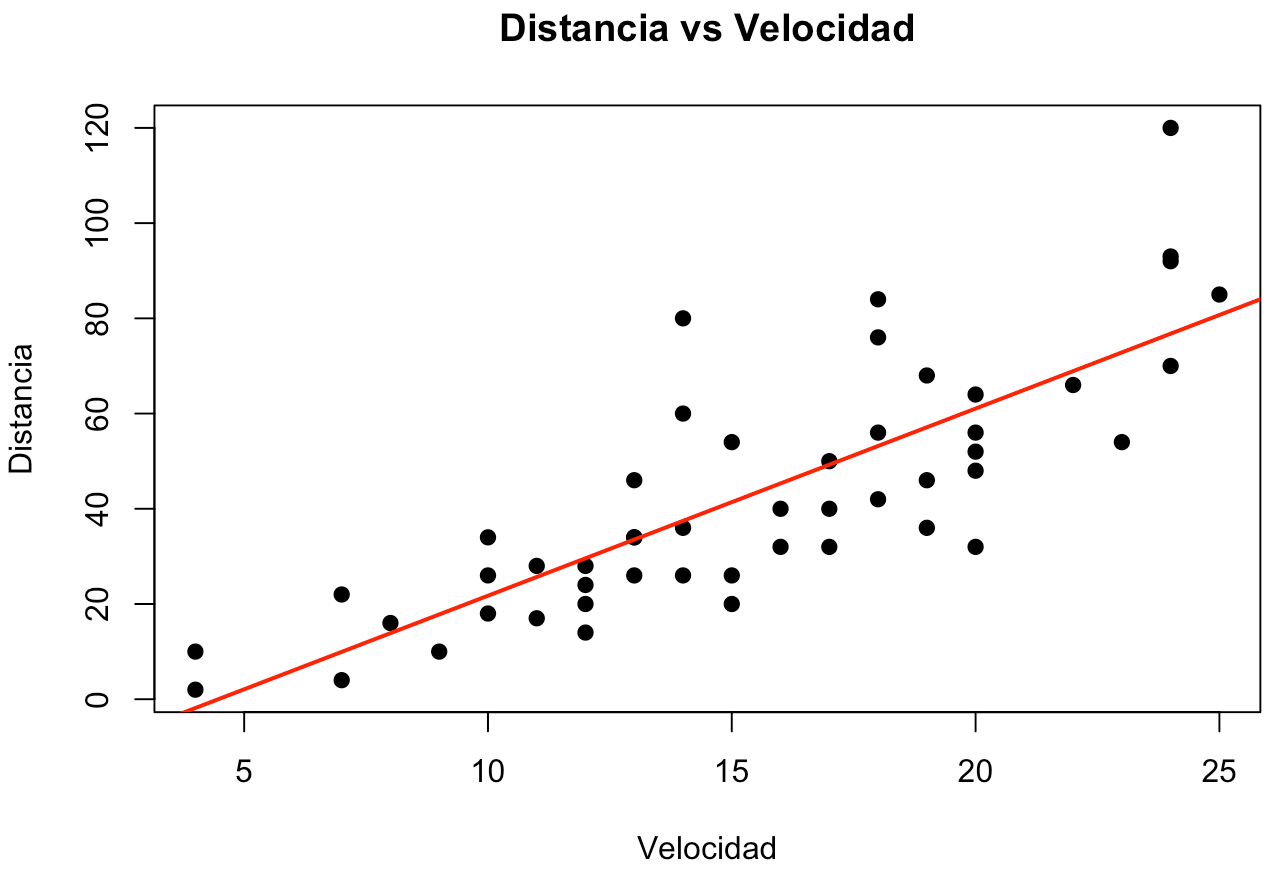
```
## El modelo lineal es:
```

```
cat("Distancia =", round(coeficientes[1], 4), "+", round(coeficientes[2], 4), "* Velocid
ad\n")
```

```
## Distancia = -17.5791 + 3.9324 * Velocidad
```

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

```
plot(cars$speed, cars$dist, main = "Distancia vs Velocidad", xlab = "Velocidad", ylab =
"Distancia", pch = 19)
abline(modelo, col = "red", lwd = 2)
```



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

```
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
```

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.

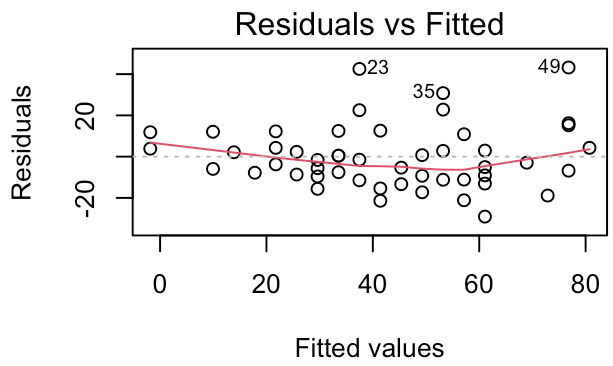
```
media_residuos = mean(modelo$residuals)
cat("La media de los residuos es:", round(media_residuos, 4), "\n")
```

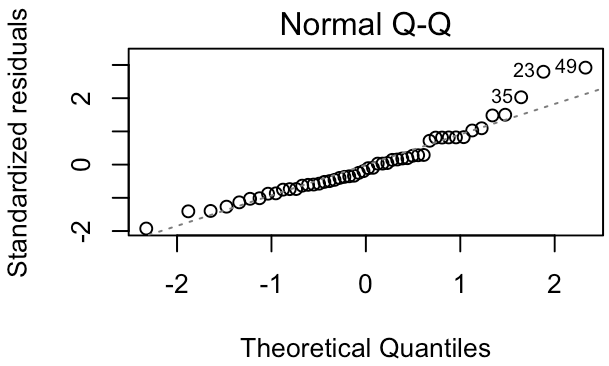
```
## La media de los residuos es: 0
```

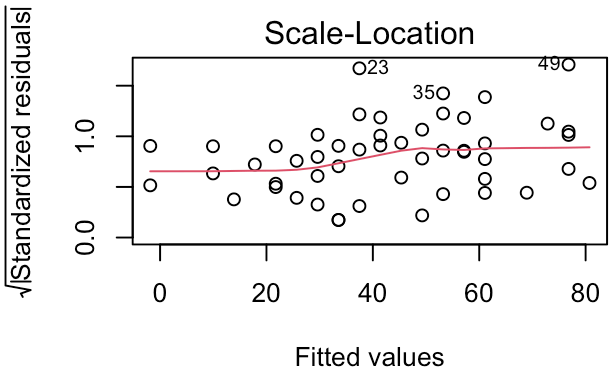
```
shapiro.test(modelo$residuals)
```

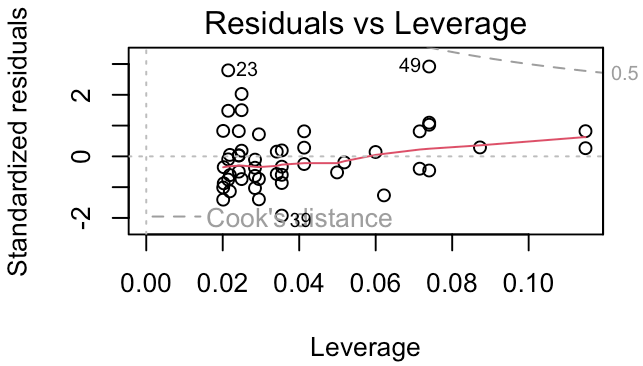
```
##
##  Shapiro-Wilk normality test
##
## data:  modelo$residuals
## W = 0.94509, p-value = 0.02152
```

```
par(mfrow = c(2, 2))
plot(modelo)
```










```
par(mfrow = c(1, 1))
```

```
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
```

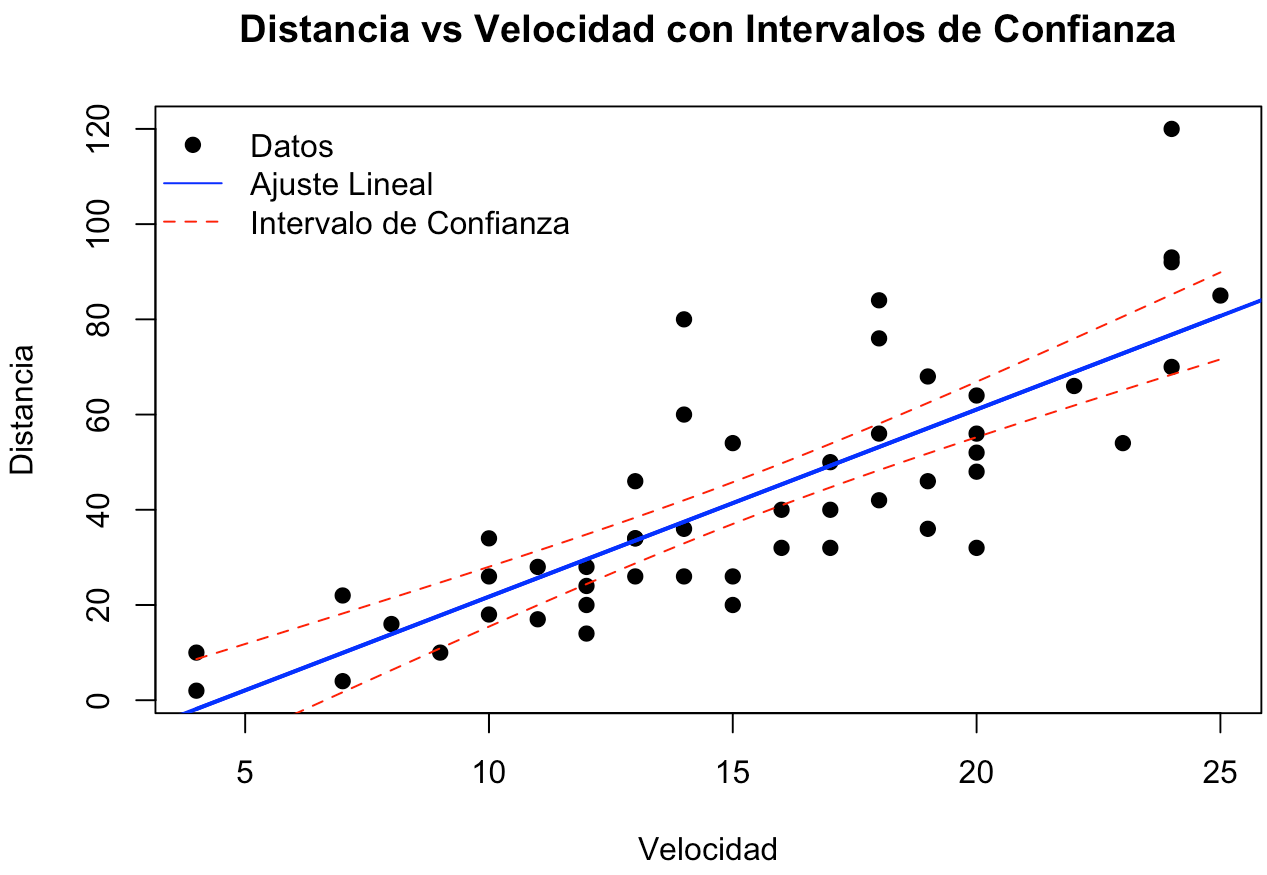
4. Grafica los datos y el modelo de la distancia en función de la velocidad.

```
plot(cars$speed, cars$dist, main = "Distancia vs Velocidad con Intervalos de Confianza",
xlab = "Velocidad", ylab = "Distancia", pch = 19)
abline(modelo, col = "blue", lwd = 2)

nuevos_datos = data.frame(speed = seq(min(cars$speed), max(cars$speed), length.out = 10
0))
predicciones = predict(modelo, newdata = nuevos_datos, interval = "confidence")

lines(nuevos_datos$speed, predicciones[, "fit"], col = "blue", lwd = 2)
lines(nuevos_datos$speed, predicciones[, "lwr"], col = "red", lty = 2)
lines(nuevos_datos$speed, predicciones[, "upr"], col = "red", lty = 2)

legend("topleft", legend = c("Datos", "Ajuste Lineal", "Intervalo de Confianza"), col =
c("black", "blue", "red"), pch = c(19, NA, NA), lty = c(NA, 1, 2), bty = "n")
```



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).

Recordando…

```
library(e1071)
speed_skewness = skewness(data$speed)
speed_kurtosis = kurtosis(data$speed)
dist_skewness = skewness(data$dist)
dist_kurtosis = kurtosis(data$dist)

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

```
## Velocidad:
```

```
cat("  Sesgo:", round(speed_skewness, 4), "\n")
```

```
##   Sesgo: -0.1106
```

```
cat("  Curtosis:", round(speed_kurtosis, 4), "\n\n")
```

```
##   Curtosis: -0.6731
```

```
cat("Distancia:\n")
```

```
## Distancia:
```

```
cat("  Sesgo:", round(dist_skewness, 4), "\n")
```

```
##   Sesgo: 0.7591
```

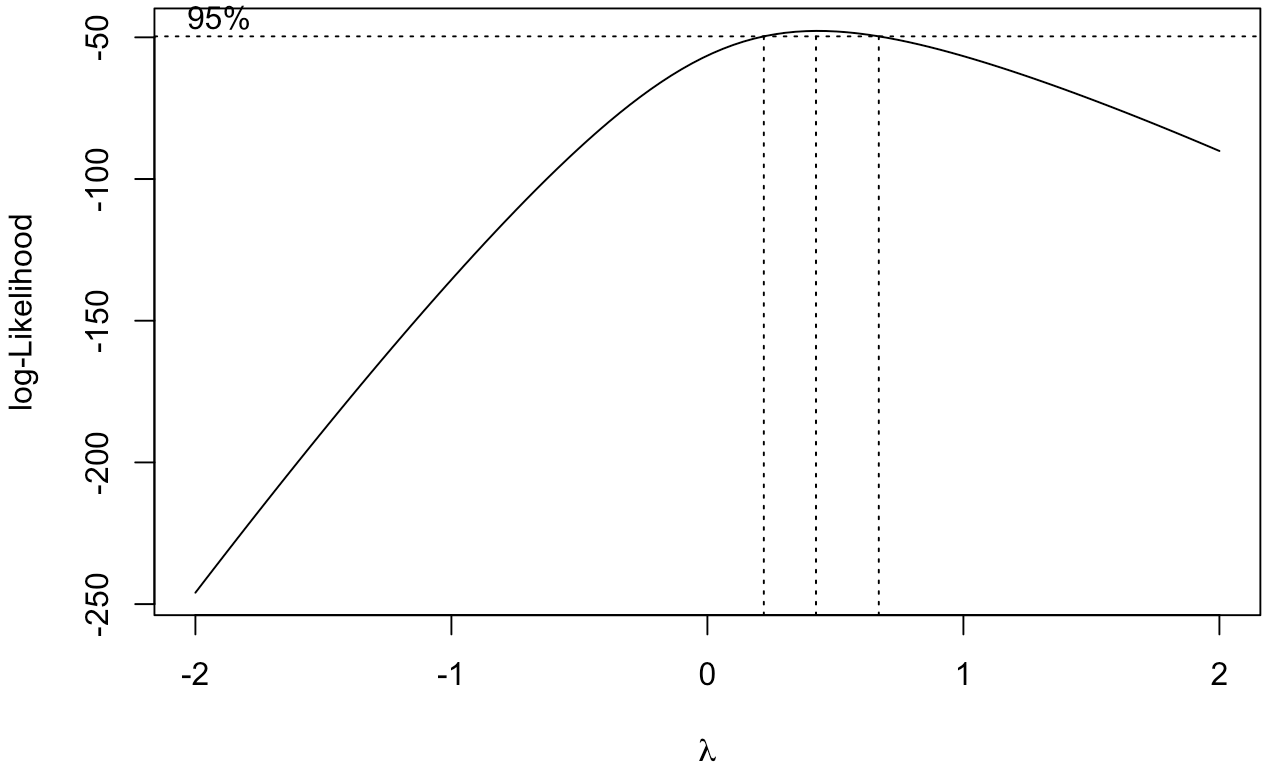
```
cat("  Curtosis:", round(dist_kurtosis, 4), "\n")
```

```
##   Curtosis: 0.1194
```

1.1. 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)

```
library(MASS)
boxcox_result = boxcox(lm(dist ~ speed, data = data), plotit = TRUE)
```




```
lambda_optimo = boxcox_result$x[which.max(boxcox_result$y)]
cat("\nEl valor óptimo de lambda es:", round(lambda_optimo, 4), "\n")
```

```
##
## El valor óptimo de lambda es: 0.4242
```

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

transformaciones encontradas.

```
lambda = lambda_optimo
data$dist_trans = (data$dist^lambda - 1) / lambda
```

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

- Compara las medidas: sesgo y curtosis

```
dist_trans_skewness = skewness(data$dist_trans)
dist_trans_kurtosis = kurtosis(data$dist_trans)

dist_trans_kurtosis_excess = dist_trans_kurtosis - 3

cat("Distancia Transformada:\n")
```

```
## Distancia Transformada:
```

```
cat("  Sesgo:", round(dist_trans_skewness, 4), "\n")
```

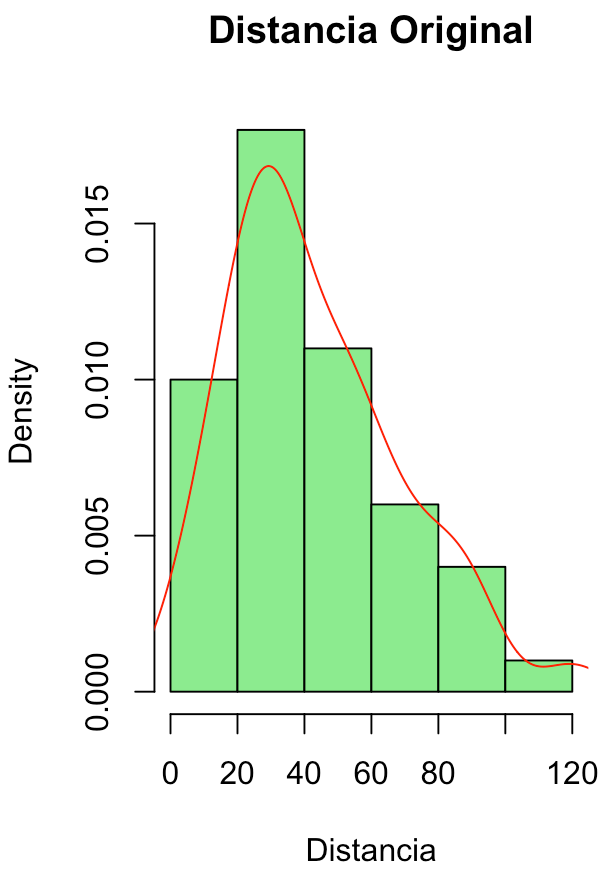
```
##   Sesgo: -0.1702
```

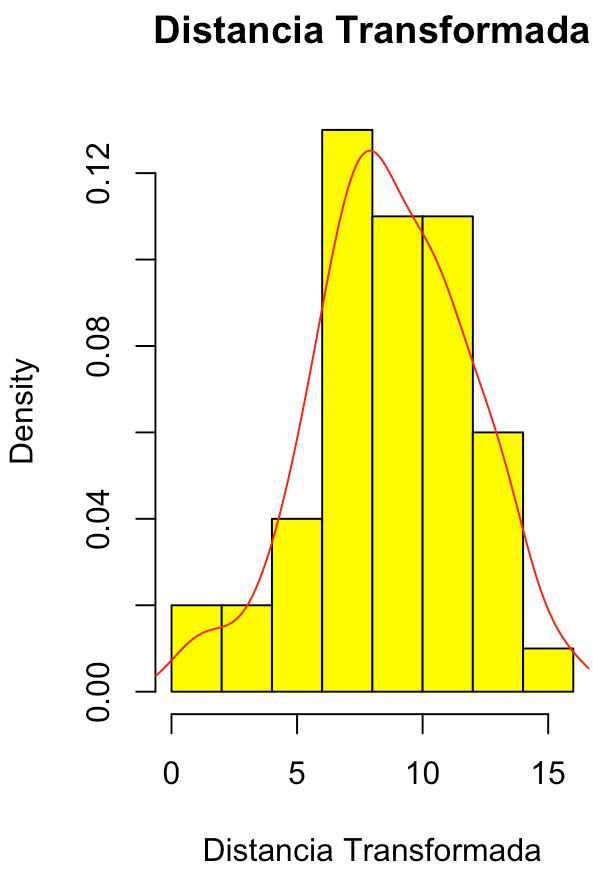
```
cat("  Curtosis en exceso:", round(dist_trans_kurtosis_excess, 4), "\n")
```

```
##   Curtosis en exceso: -3.1869
```

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

```
par(mfrow = c(1, 2))
hist(data$dist, main = "Distancia Original", xlab = "Distancia", col = "lightgreen", fre
q = FALSE)
lines(density(data$dist), col = "red")
hist(data$dist_trans, main = "Distancia Transformada", xlab = "Distancia Transformada",
col = "yellow", freq = FALSE)
lines(density(data$dist_trans), col = "red")
```






```
par(mfrow = c(1, 1))
```

- Realiza algunas pruebas de normalidad para los datos transformados.

```
shapiro_original = shapiro.test(data$dist)
cat("Prueba de Shapiro-Wilk para la distancia original:\n")
```

```
## Prueba de Shapiro-Wilk para la distancia original:
```

```
print(shapiro_original)
```

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

```
shapiro_transformada = shapiro.test(data$dist_trans)
cat("\nPrueba de Shapiro-Wilk para la distancia transformada:\n")
```

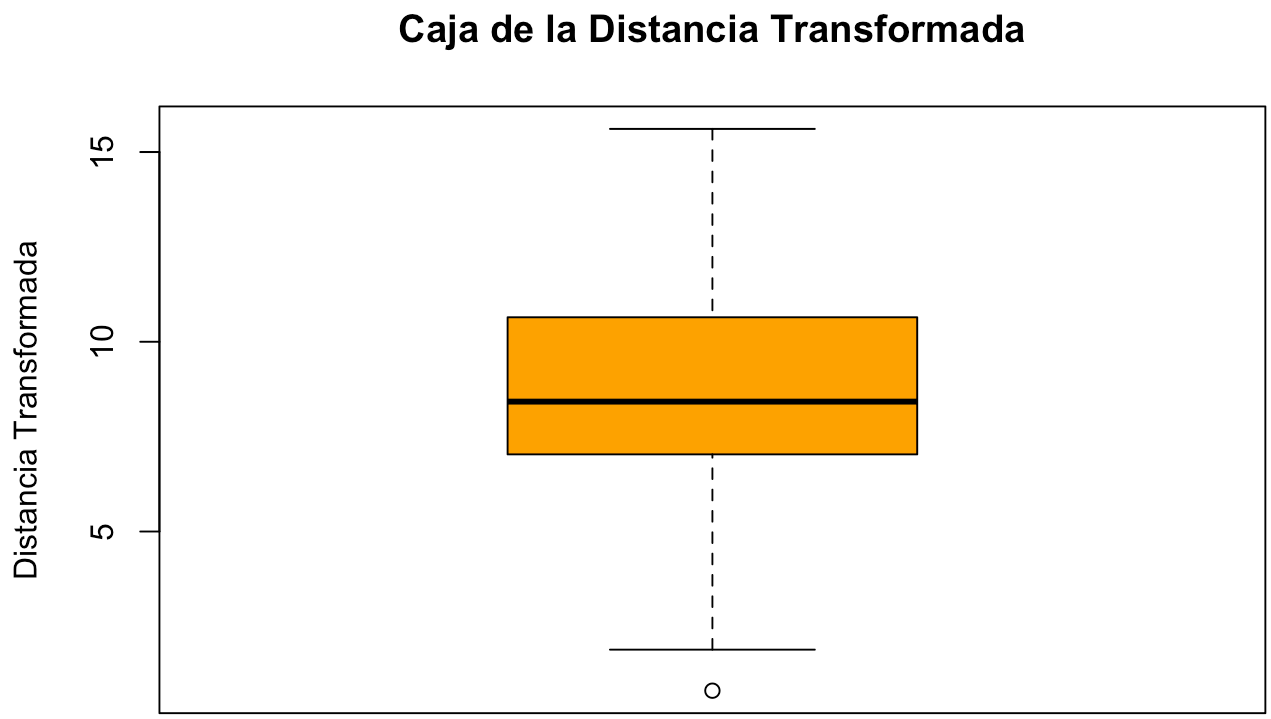
```
##
## Prueba de Shapiro-Wilk para la distancia transformada:
```

```
print(shapiro_transformada)
```

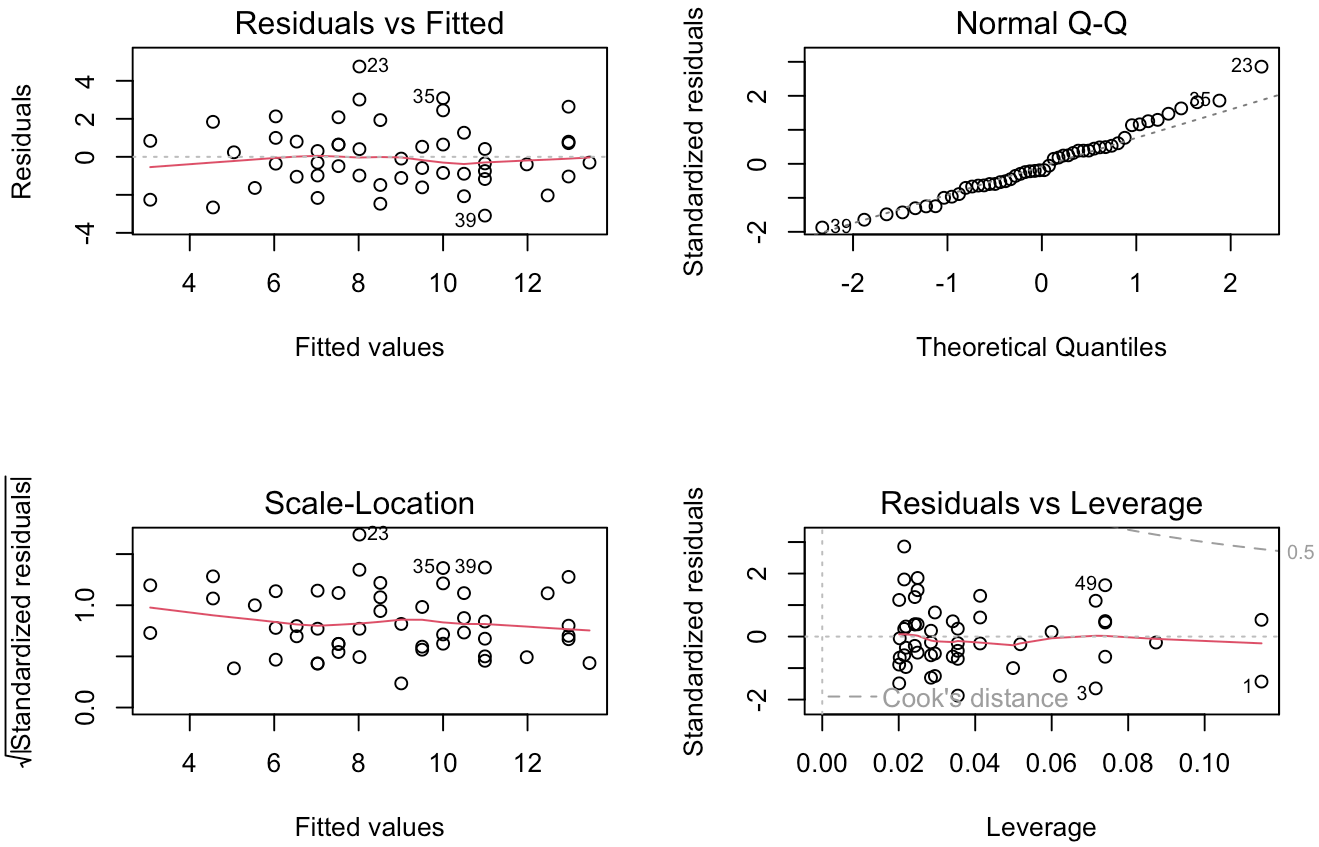
```
##
##  Shapiro-Wilk normality test
##
## data:  data$dist_trans
## W = 0.99168, p-value = 0.9773
```

1.4. 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 λ para tus nuevos datos.

```
boxplot(data$dist_trans, main = "Caja de la Distancia Transformada", ylab = "Distancia T
ransformada", col = "orange")
```



```
modelo_trans = lm(dist_trans ~ speed, data = data)
par(mfrow = c(2, 2))
plot(modelo_trans)
```




```
par(mfrow = c(1, 1))
```

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.

La transformación aproximada es la mejor ya que es más fácil de calcular y explicar, lo que es beneficioso para la comunicación de los resultados y proporciona una mejora significativa en la normalidad de los datos, con sesgo y curtosis aceptables y una prueba de normalidad favorable.

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

1. Escribe el modelo lineal para la transformación.

```
data(cars)
cars$log_dist = log(cars$dist)
modelo_log = lm(log_dist ~ speed, data = cars)
coeficientes_log = coef(modelo_log)
cat("El modelo lineal es:\n")
```

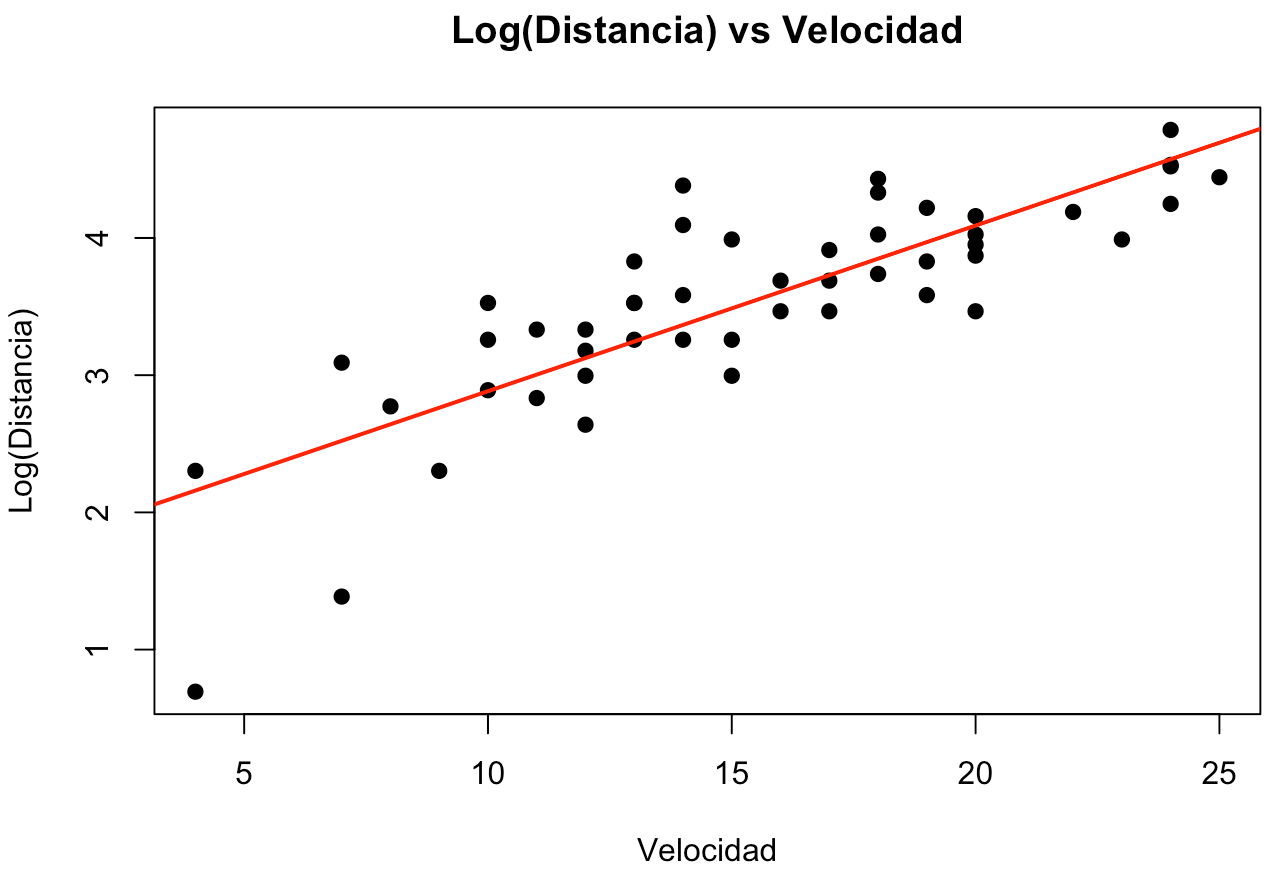
```
## El modelo lineal es:
```

```
cat("log(Distancia) =", round(coeficientes_log[1], 4), "+", round(coeficientes_log[2],
4), "* Velocidad\n")
```

```
## log(Distancia) = 1.6761 + 0.1208 * Velocidad
```

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

```
plot(cars$speed, cars$log_dist, main = "Log(Distancia) vs Velocidad", xlab = "Velocida
d", ylab = "Log(Distancia)", pch = 19)
abline(modelo_log, col = "red", lwd = 2)
```



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

```
summary_log = summary(modelo_log)
print(summary_log)
```

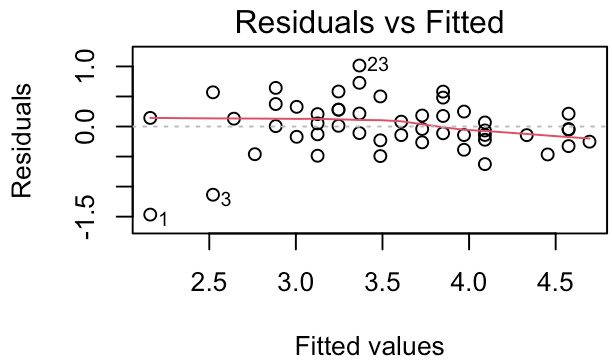
```
##
## Call:
## lm(formula = log_dist ~ speed, data = cars)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.46604 -0.20800 -0.01683  0.24080  1.01519
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.67612    0.19614   8.546 3.34e-11 ***
## speed        0.12077    0.01206  10.015 2.41e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4463 on 48 degrees of freedom
## Multiple R-squared:  0.6763, Adjusted R-squared:  0.6696
## F-statistic: 100.3 on 1 and 48 DF,  p-value: 2.413e-13
```

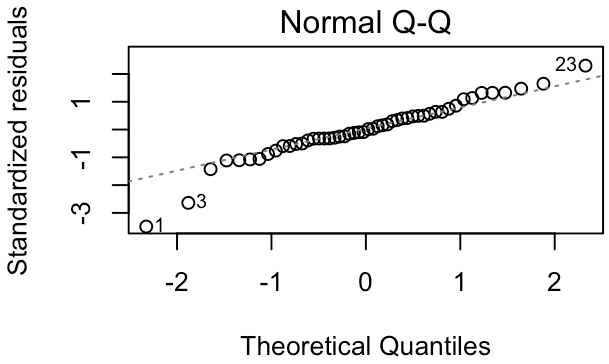
```
correlacion = cor(cars$speed, cars$log_dist)
cat("\nCoeficiente de correlación:", round(correlacion, 4), "\n")
```

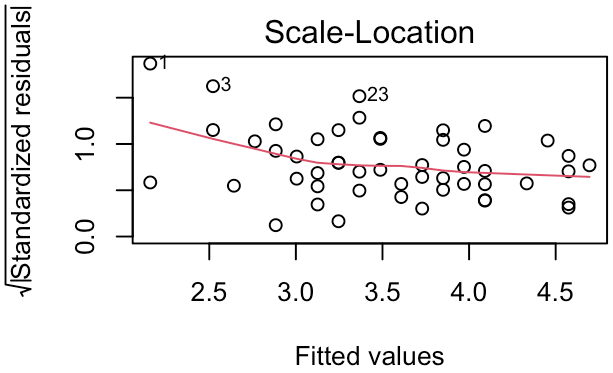
```
##
## Coeficiente de correlación: 0.8224
```

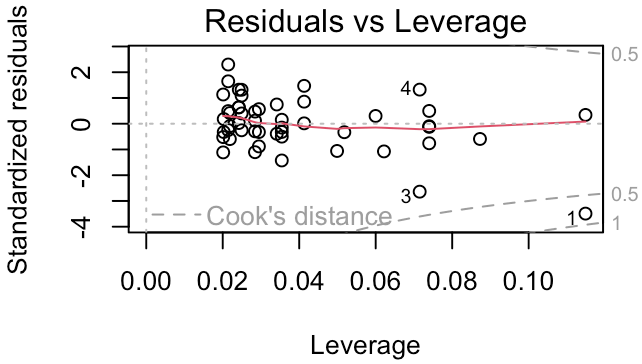
4. Analiza validez del modelo: normalidad de los residuos, homocedasticidad e independencia. Indica si hay candidatos a datos atípicos o influyentes en la regresión. Usa plot(Modelo) para los gráficos y añade pruebas de hipótesis.

```
par(mfrow = c(2, 2))
plot(modelo_log)
```










```
par(mfrow = c(1, 1))
```

```
shapiro_residuos = shapiro.test(residuals(modelo_log))
print(shapiro_residuos)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(modelo_log)
## W = 0.95734, p-value = 0.06879
```

```
bp_test = bptest(modelo_log)
print(bp_test)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  modelo_log
## BP = 8.2963, df = 1, p-value = 0.003973
```

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).

```
beta0 = coeficientes_log[1]
beta1 = coeficientes_log[2]
A = exp(beta0)
B = beta1
cat("El modelo no lineal es:\n")
```

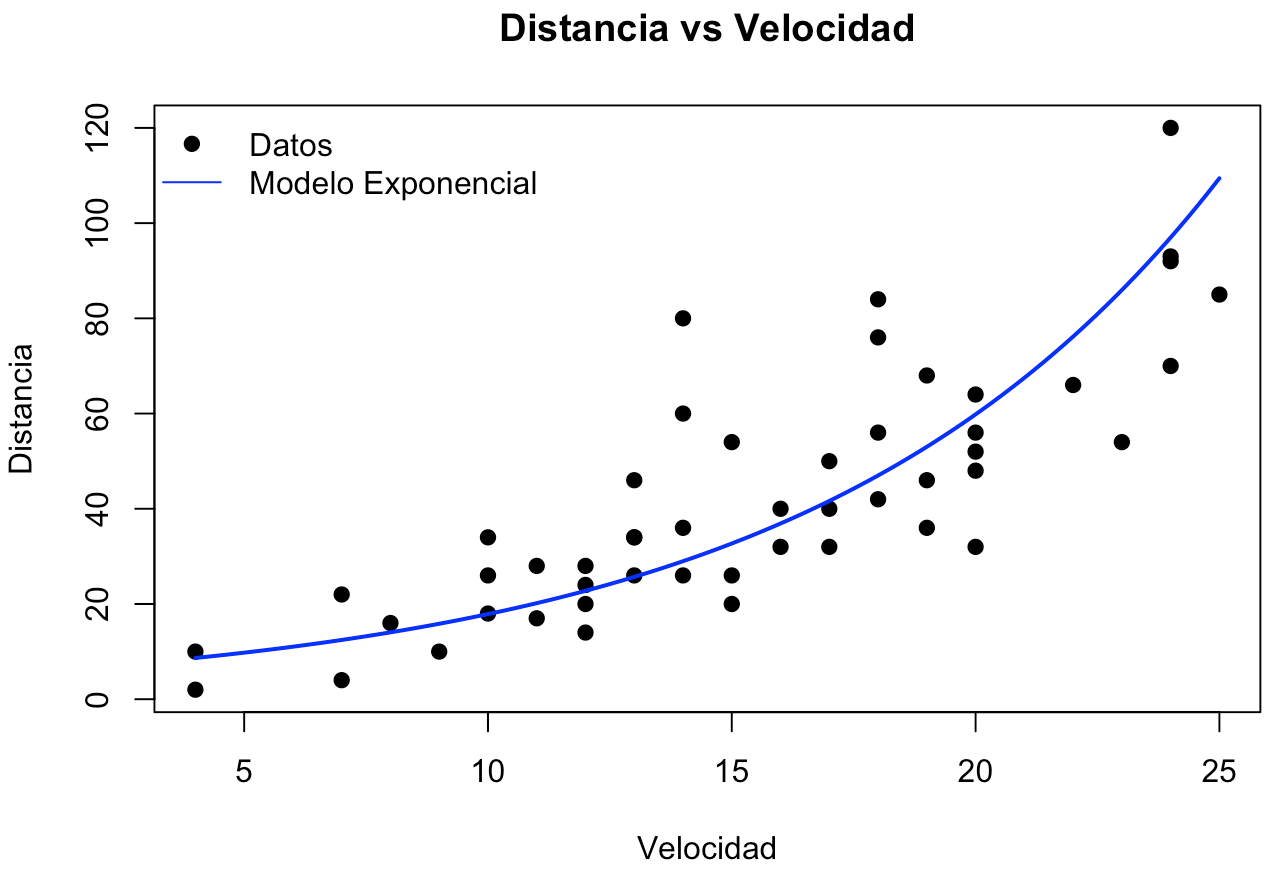
```
## El modelo no lineal es:
```

```
cat("Distancia =", round(A, 4), "* e^(", round(B, 4), "* Velocidad)\n")
```

```
## Distancia = 5.3448 * e^( 0.1208 * Velocidad)
```

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

```
plot(cars$speed, cars$dist, main = "Distancia vs Velocidad", xlab = "Velocidad", ylab =
"Distancia", pch = 19)
velocidad_nueva = seq(min(cars$speed), max(cars$speed), length.out = 100)
distancia_ajustada = A * exp(B * velocidad_nueva)
lines(velocidad_nueva, distancia_ajustada, col = "blue", lwd = 2)
legend("topleft", legend = c("Datos", "Modelo Exponencial"), col = c("black", "blue"), p
ch = c(19, NA), lty = c(NA, 1), bty = "n")
```



Parte 4: Conclusión

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

Una vez realizados ambos modelos podemos observar que el modelo no lineal o modelo exponencial es el mejor para describir la relación entre la distancia y la velocidad, ya que nos ofrece un mejor ajuste, cumple con los supuestos estadísticos necesarios y proporciona una interpretación más realista de cómo la velocidad afecta la distancia en esta caso de frenado.