Actividad 13

Saúl Francisco Vázquez del Río

2024-09-10

Parte 1: Análisis de normalidad

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

```
M = data(cars)
head(cars)
```

```
##   speed dist
## 1     4    2
## 2     4   10
## 3     7    4
## 4     7   22
## 5     8   16
## 6     9   10
```

```
speed = cars$speed
dist = cars$dist
```

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

```
library(nortest)
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

```
ad.test(speed)
```

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

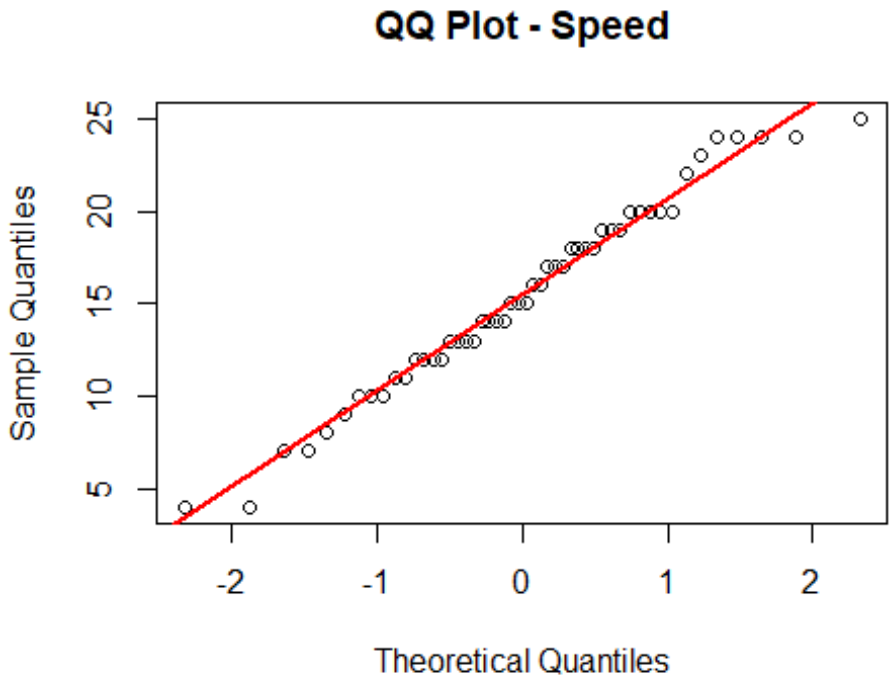
```
ad.test(dist)
```

```
##
##  Anderson-Darling normality test
```

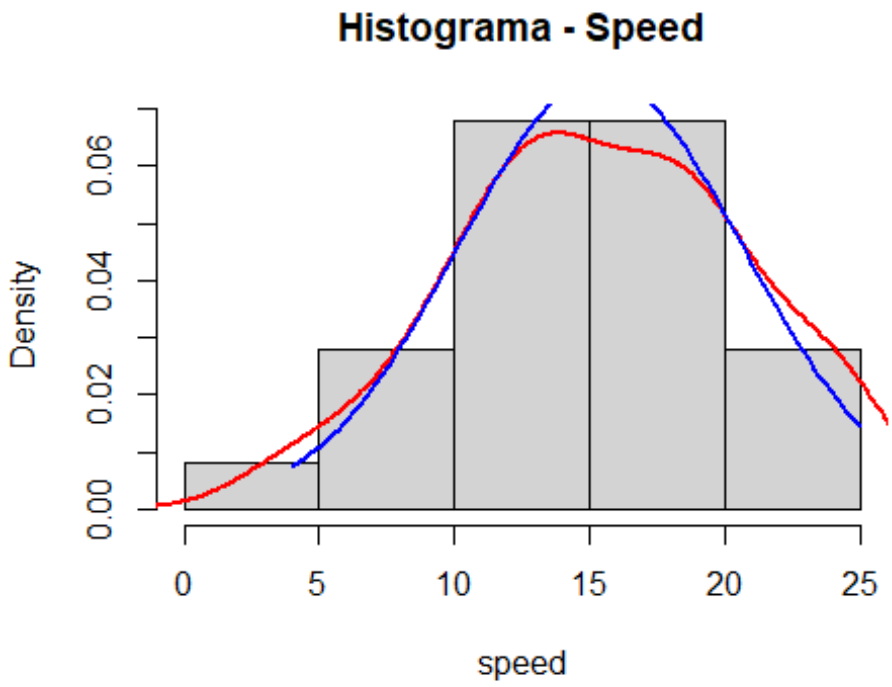
```
##
## data:  dist
## A = 0.74067, p-value = 0.05021
```

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.

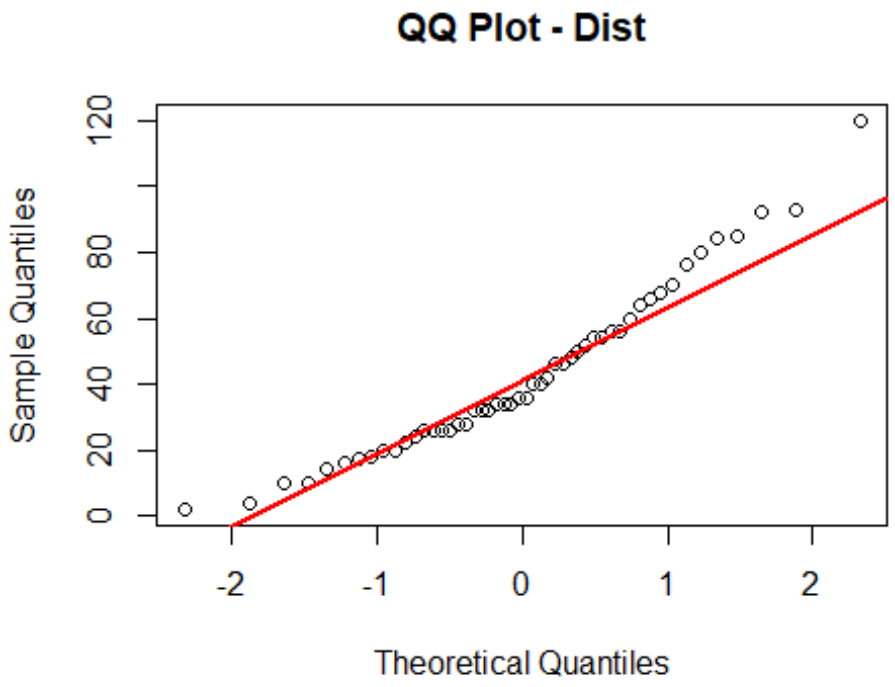
```r
# Gráficos para la variable 'speed'
# 1. QQ Plot para 'speed'
qqnorm(speed, main="QQ Plot - Speed")
qqline(speed, col="red", lwd=2)
```



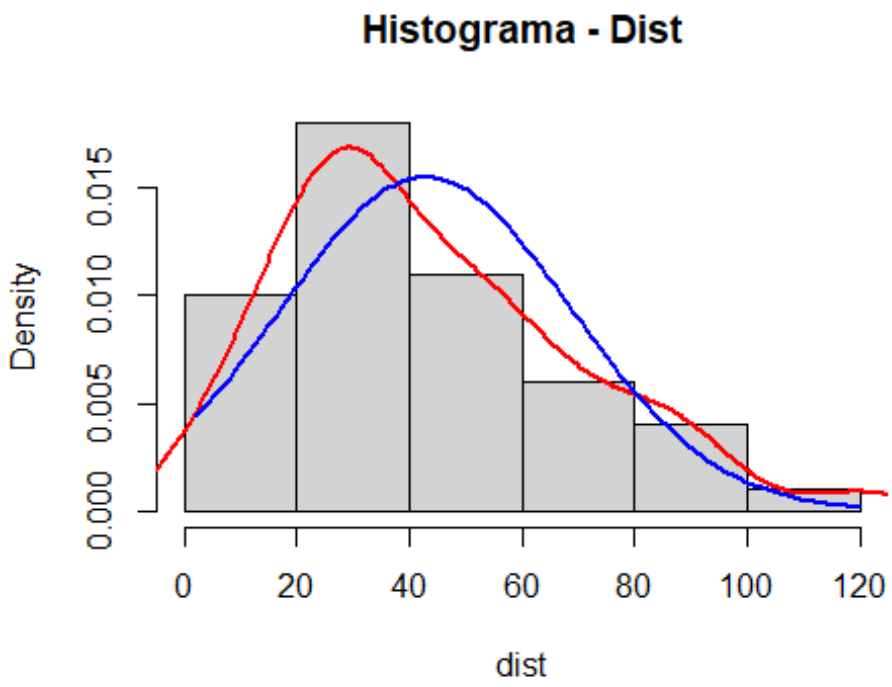

```r
# 2. Histograma con curva teórica para 'speed'
hist(speed, freq=FALSE, main="Histograma - Speed", col="lightgray")
lines(density(speed), col="red", lwd=2)
curve(dnorm(x, mean=mean(speed), sd=sd(speed)),
      from=min(speed), to=max(speed), add=TRUE, col="blue", lwd=2)
```




```r
# Gráficos para la variable 'dist'
# 3. QQ Plot para 'dist'
qqnorm(dist, main="QQ Plot - Dist")
qqline(dist, col="red", lwd=2)
```




```r
# 4. Histograma con curva teórica para 'dist'
hist(dist, freq=FALSE, main="Histograma - Dist", col="lightgray")
lines(density(dist), col="red", lwd=2)
curve(dnorm(x, mean=mean(dist), sd=sd(dist)),
      from=min(dist), to=max(dist), add=TRUE, col="blue", lwd=2)
```



QQplot de speed: No se alejan mucho los puntos de la linea roja.

Histograma de speed: Se muestra que ambas curvas si siguien los datos pero la curva azul sobre sale haciendo que los datos no siguen una normalidad

QQplot de distancia: Los puntos tienden alejarse más que en qqplot de speed teniendo datos atipicos

Histograma de dist: Igual que en el histograma de speed la line aroja sigue los resultados pero la liena azul sobrela, haciendo que los datos no siguen una normalidad

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

```
library(e1071)
print("Curtosis de speed")

## [1] "Curtosis de speed"

kurtosis(speed)

## [1] -0.6730924

print("Sesgo de speed")

## [1] "Sesgo de speed"

skewness(speed)
```

```
## [1] -0.1105533
```

```
print("Curtosis de dist")
```

```
## [1] "Curtosis de dist"
```

```
kurtosis(dist)
```

```
## [1] 0.1193971
```

```
print("Sesgo de dist")
```

```
## [1] "Sesgo de dist"
```

```
skewness(dist)
```

```
## [1] 0.7591268
```

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 base a los graficos se observa que ambas variables no siguen una distribuciones normales, pero speed presenta una distribucion normal ya que, tiene sesgo positivo indicado una ligera asimetría a la derecha.Y distancia con un sesgo y curtosis indican una asimetría significativa.

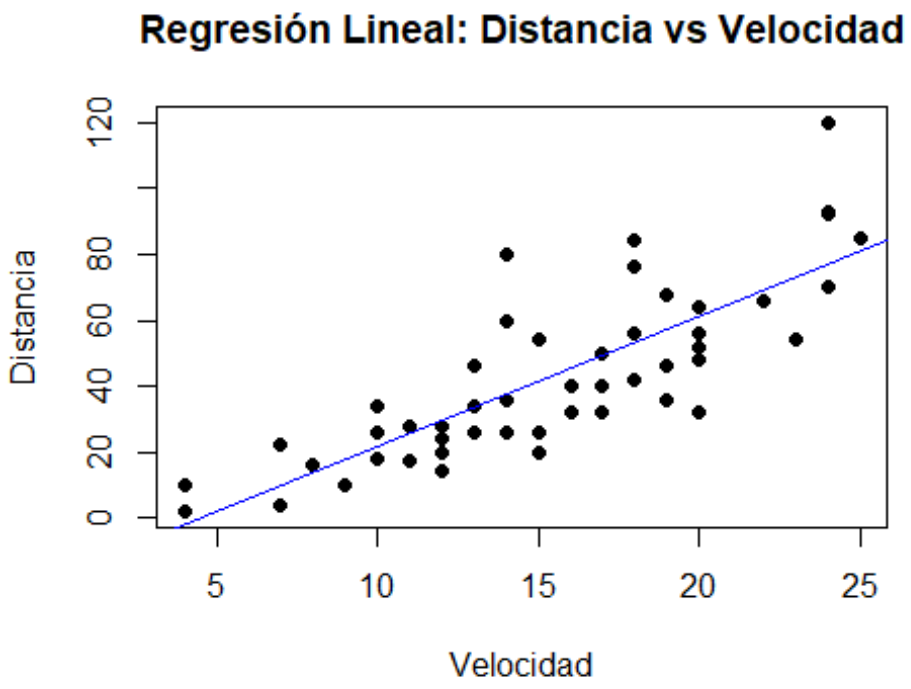
##Parte 2: Regresión lineal

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

```
modelo <- lm(dist~speed)
print(modelo)
```

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

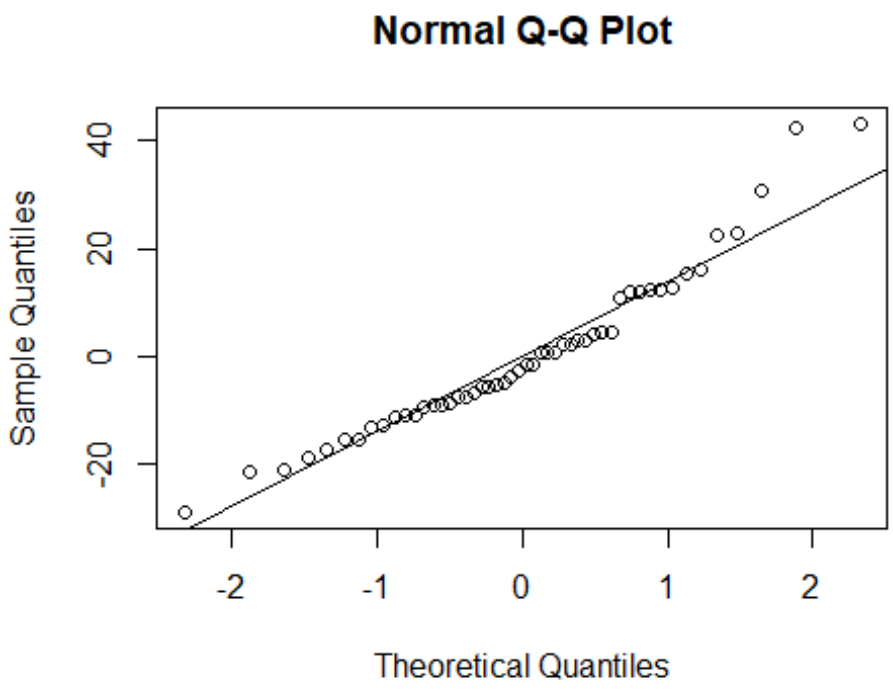
```
plot(speed, dist, main = "Regresión Lineal: Distancia vs Velocidad",
     xlab = "Velocidad", ylab = "Distancia", pch = 19)
abline(modelo, col = "blue")  # Línea de regresión ajustada
```



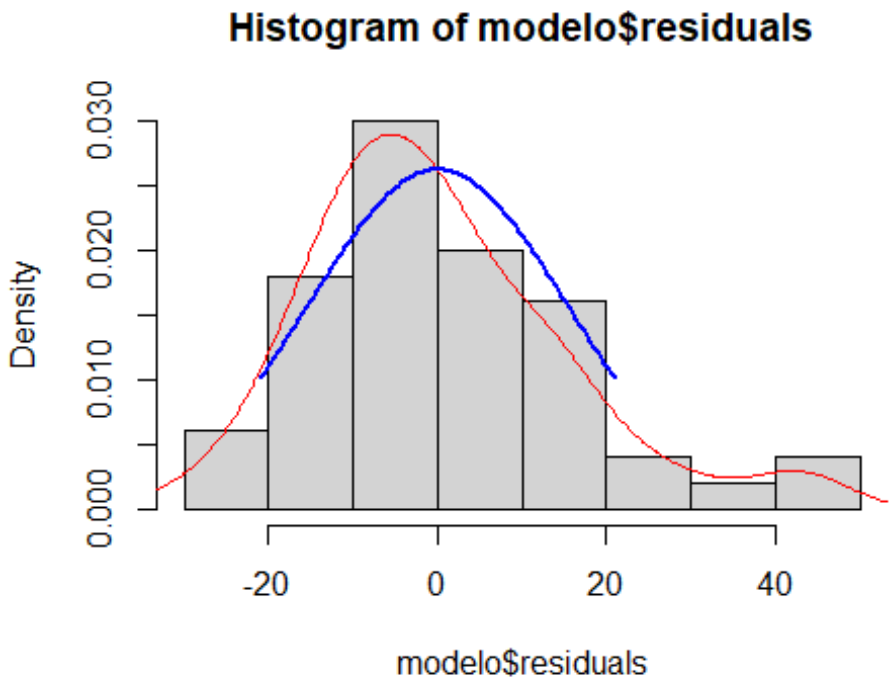

```
summary(modelo)

##
## Call:
## lm(formula = dist ~ speed)
##
## 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. Residuos con media cero Normalidad de los residuos Homocedasticidad, independencia y linealidad. Usa plot(Modelo) para los gráficos y añade pruebas de hipótesis. Grafica los datos y el modelo de la distancia en función de la velocidad.

```r
# Normalidad de los residuos speed
library(nortest)
ad.test(modelo$residuals)

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

qqnorm(modelo$residuals)
qqline(modelo$residuals)
```




```r
hist(modelo$residuals,freq=FALSE)
lines(density(modelo$residual),col="red")
curve(dnorm(x,mean=mean(modelo$residuals),sd=sd(modelo$residuals)),
from=-
21, to=21, add=TRUE, col="blue",lwd=2)
```



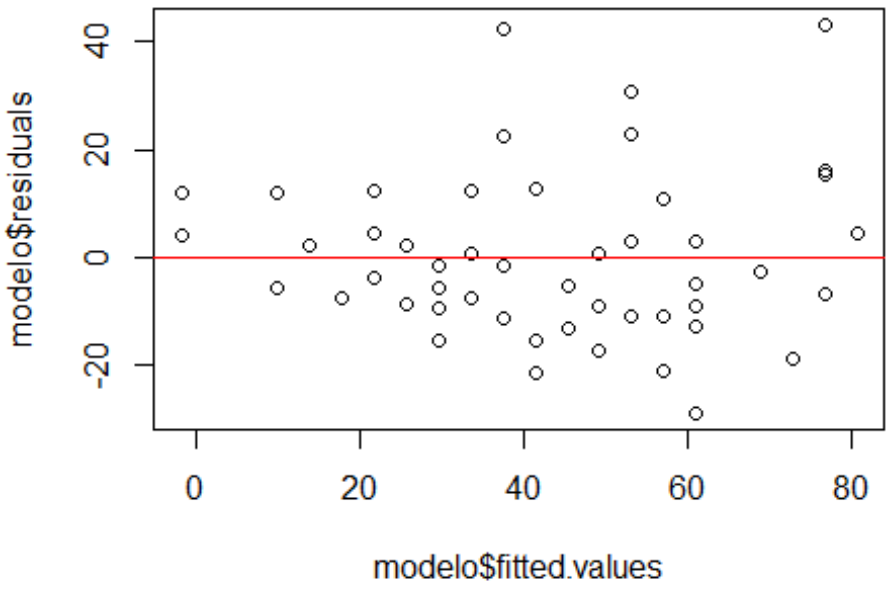

```r
# Residuos con media cero
t.test(modelo$residuals)
```

```
##
##  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
```

```r
# Homocedasticidad, independencia y linealidad
plot(modelo$fitted.values,modelo$residuals)
abline(h=0, col="red")

library(lmtest)
```

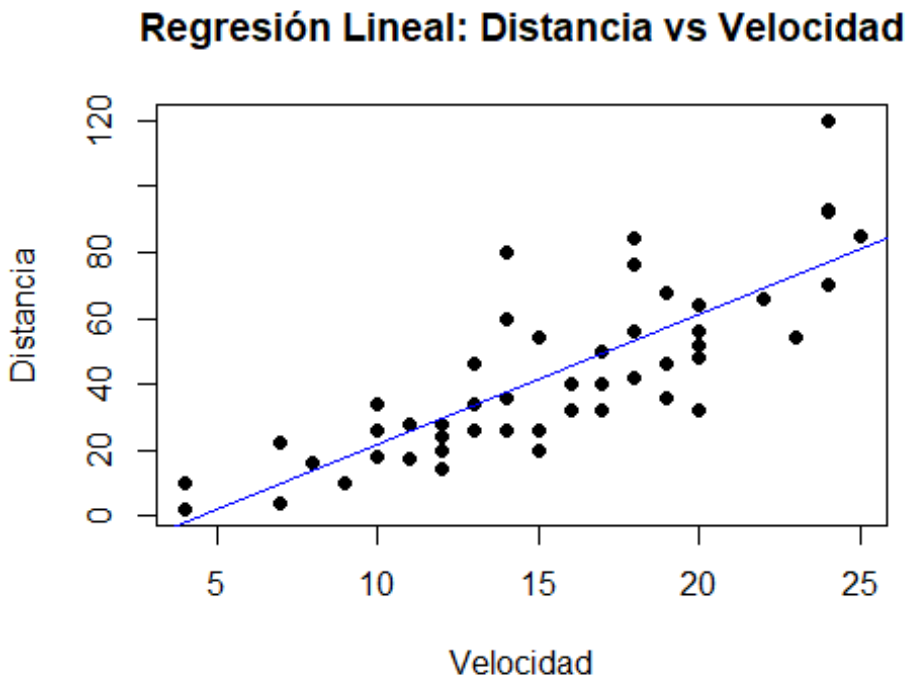
```
## Cargando paquete requerido: zoo

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

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




```
# Homocedasticidad
bptest(modelo)

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

# Independencia
dwtest(modelo)

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

# Linealidad
resettest(modelo)

##
##  RESET test
```

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

plot(speed, dist, main = "Regresión Lineal: Distancia vs Velocidad",
     xlab = "Velocidad", ylab = "Distancia", pch = 19)
abline(modelo, col = "blue")  # Línea de regresión ajustada
```



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

El modelo tiene significancia ya que los coeficientes del modelo significativos al igual que la R cuadrada es significativa, los residuos cumplen con la normalidad, homestacidad, independecia y linealidad. Si hubiera problemas con los residuos solo abria que transformalos.

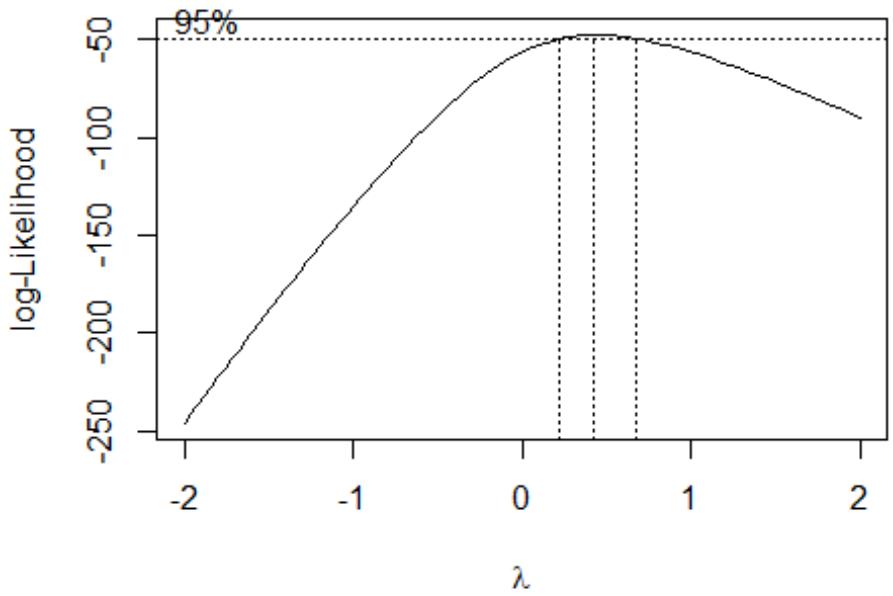
#Parte 3: Regresión no lineal

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).

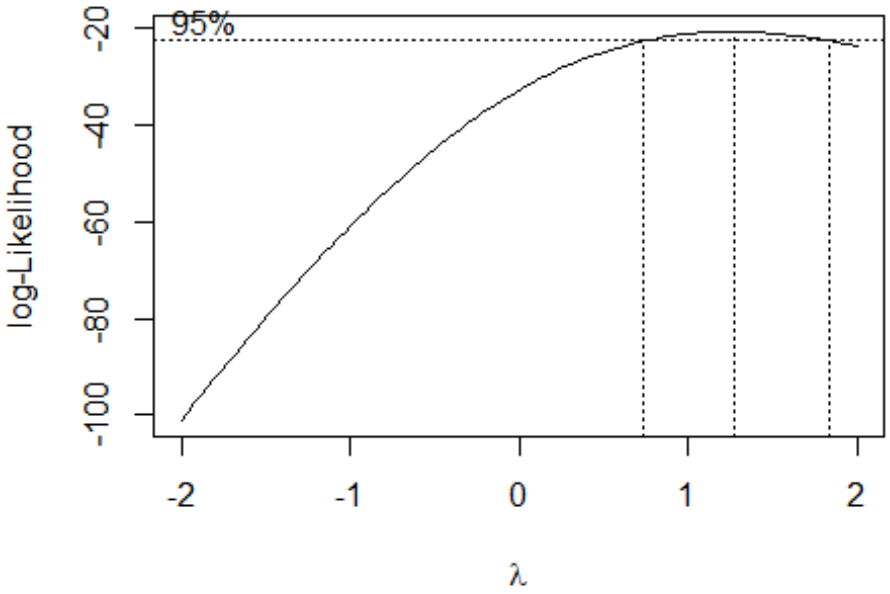
Encuentra el valor de en la transformación Box-Cox para el modelo lineal: donde Y sea la distancia y X la velocidad. Aprovecha que el comando de boxcox en R te da la oportunidad de trabajar con el modelo líneal: Utiliza:
boxcox(lm(Distancia~Velocidad)) si la variable con más alejamiento de normalidad es

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

```
library(MASS)  # Cargar la librería para Box-Cox


#Modelo con distancia
modelo_boxcoxdist <- boxcox(lm(dist ~ speed, data=cars))  # Aplicar
transformación Box-Cox al modelo lineal
```




```
lambda_optimodist <- modelo_boxcoxdist$x[which.max(modelo_boxcoxdist$y)]
# Encontrar el lambda óptimo
lambda_optimodist  # Imprime el valor de lambda

## [1] 0.4242424

#Modelo con velocidad
modelo_boxcoxspeed <- boxcox(lm(speed ~ dist, data=cars))  # Aplicar
transformación Box-Cox al modelo lineal
```




```
lambda_optimospeed <-
modelo_boxcoxspeed$x[which.max(modelo_boxcoxspeed$y)]  # Encontrar el
lambda óptimo
lambda_optimospeed  # Imprime el valor de lambda
```

```
## [1] 1.272727
```

La transformación se hará sobre la variable que usas como dependiente en el comando lm(y~x) 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.

Ecuacion de distancia: $Y_Transformada = log(Y)$ Ecuacion de velocidad: $Y_Transformada = Y(\lambda) - 1/\lambda$

Analiza la normalidad de las transformaciones obtenidas. Utiliza como argumento de normalidad: Compara las medidas: sesgo y curtosis.

```
library(moments)
```

```
##
## Adjuntando el paquete: 'moments'
```

```
## The following objects are masked from 'package:e1071':
##
##     kurtosis, moment, skewness
```

```r
# Transformar la variable dist utilizando el valor de lambda_optimo
if (lambda_optimodist == 0) {
  dist_trans <- log(cars$dist)  # Si lambda es 0, aplica logaritmo
} else {
  dist_trans <- (cars$dist^lambda_optimodist - 1) / lambda_optimodist  # Aplicar la transformación Box-Cox exacta
}

# Transformar la variable speed utilizando el valor de lambda_optimo
if (lambda_optimospeed == 0) {
  speed_trans <- log(cars$speed)  # Si lambda es 0, aplica logaritmo
} else {
  speed_trans <- (cars$speed^lambda_optimospeed - 1) / lambda_optimospeed
# Aplicar la transformación Box-Cox exacta
}


# Calcular sesgo y curtosis antes de la transformación
print("Sesgo y curtosis de distancia")
```

```
## [1] "Sesgo y curtosis de distancia"
```

```r
skewness(cars$dist)
```

```
## [1] 0.7824835
```

```r
kurtosis(cars$dist)
```

```
## [1] 3.248019
```

```r
print("Sesgo y curtosis de velocidad")
```

```
## [1] "Sesgo y curtosis de velocidad"
```

```r
skewness(cars$speed)
```

```
## [1] -0.1139548
```

```r
kurtosis(cars$speed)
```

```
## [1] 2.422853
```

```r
# Calcular sesgo y curtosis después de la transformación
print("Sesgo y curtosis de distancia transformada")
```

```
## [1] "Sesgo y curtosis de distancia transformada"
```

```r
skewness(dist_trans)
```

```
## [1] -0.1753974
```

```r
kurtosis(dist_trans)
```

```
## [1] 2.929109
```

```r
print("Sesgo y curtosis de velocidad transformada")
```

```
## [1] "Sesgo y curtosis de velocidad transformada"
```

```r
skewness(speed_trans)
```

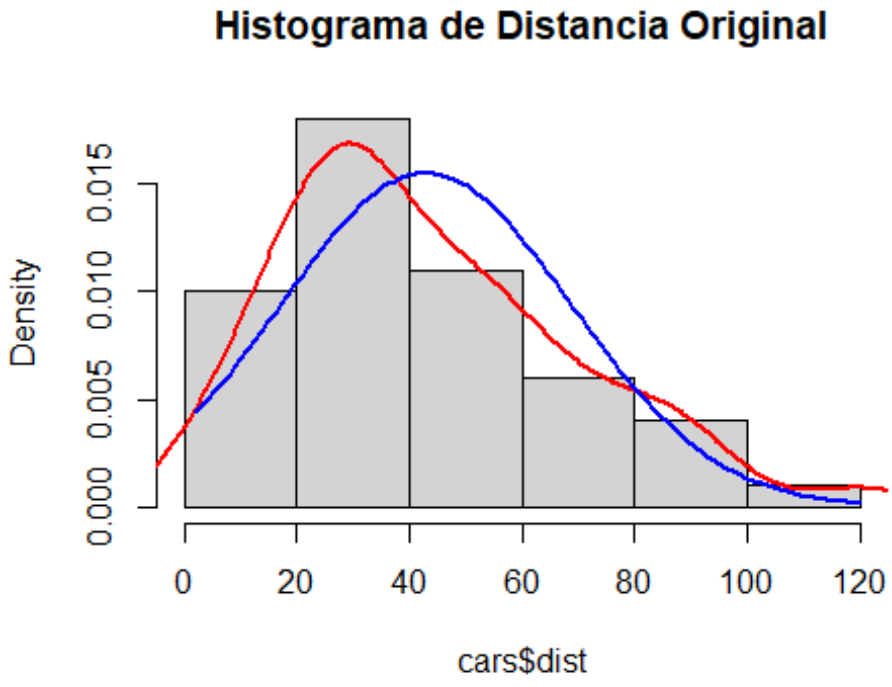
```
## [1] 0.09893401
```

```r
kurtosis(speed_trans)
```

```
## [1] 2.303761
```

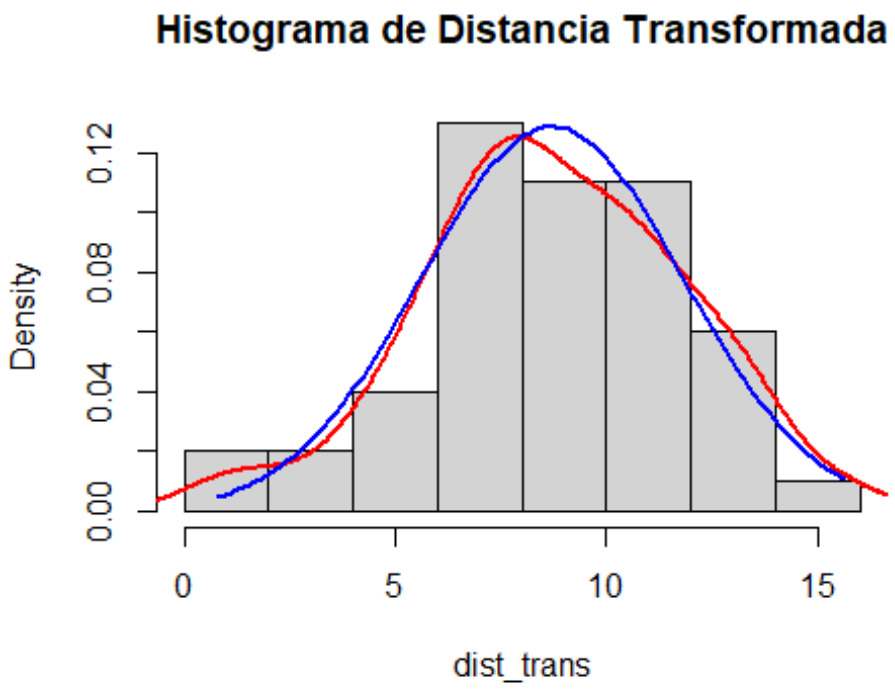
Obten el histograma de los 2 modelos obtenidos (exacto y aproximado) y los datos originales. Realiza algunas pruebas de normalidad para los datos transformados.

```r
library(moments)
library(nortest)  # Para pruebas de normalidad

# Histograma de la distancia original
hist(cars$dist, freq=FALSE, main="Histograma de Distancia Original",
col="lightgray")
lines(density(cars$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)
```




```r
# Histograma de la distancia transformada
hist(dist_trans, freq=FALSE, main="Histograma de Distancia Transformada",
```

```
col="lightgray")
lines(density(dist_trans), col="red", lwd=2)
curve(dnorm(x, mean=mean(dist_trans), sd=sd(dist_trans)),
      from=min(dist_trans), to=max(dist_trans), add=TRUE, col="blue",
lwd=2)
```




```
# Pruebas de normalidad para los datos transformados
print("Prueba de normalidad para distancia transformada")

## [1] "Prueba de normalidad para distancia transformada"

ad.test(dist_trans)

##
##  Anderson-Darling normality test
##
## data:  dist_trans
## A = 0.14027, p-value = 0.9717
```

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 para tus nuevos datos.

```
# Detectar y corregir anomalías en la distancia transformada
# Calcular el IQR para la distancia transformada
Q1_dist <- quantile(dist_trans, 0.25)
Q3_dist <- quantile(dist_trans, 0.75)
IQR_dist <- Q3_dist - Q1_dist
```

```r
# Definir límites para los valores atípicos
lower_bound_dist <- Q1_dist - 1.5 * IQR_dist
upper_bound_dist <- Q3_dist + 1.5 * IQR_dist

# Detectar valores atípicos
outliers_dist <- dist_trans < lower_bound_dist | dist_trans >
upper_bound_dist

# Imprimir cantidad de valores atípicos detectados
cat("Número de valores atípicos en distancia transformada:",
sum(outliers_dist), "\n")
```

```
## Número de valores atípicos en distancia transformada: 1
```

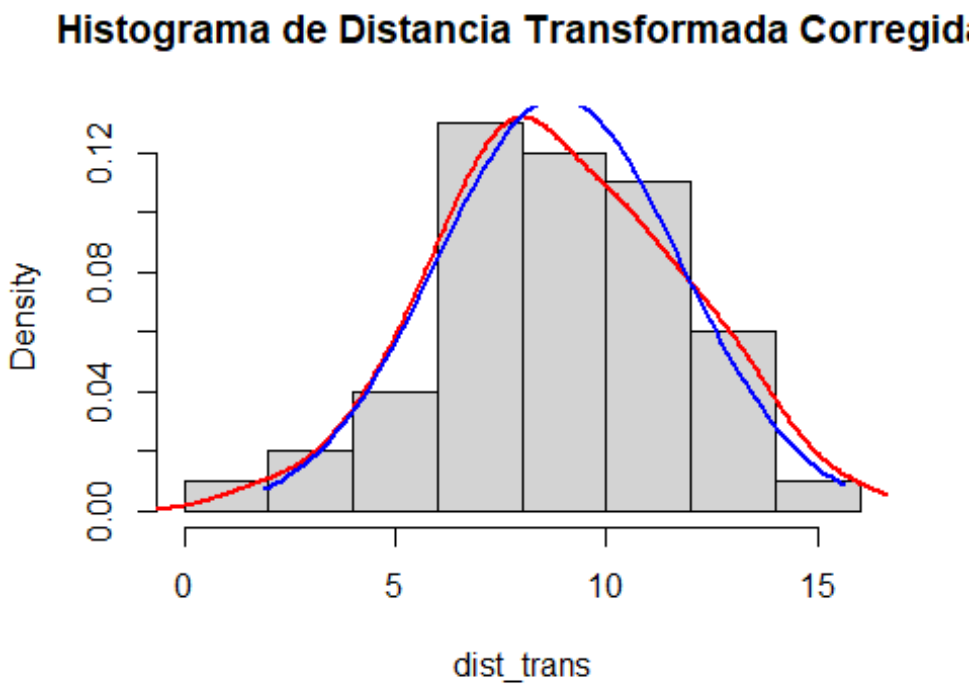
```r
# Reemplazar valores atípicos con NA
dist_trans[outliers_dist] <- NA

# Imputar los valores NA con la mediana
dist_trans[is.na(dist_trans)] <- median(dist_trans, na.rm = TRUE)

# Pruebas de normalidad para los datos corregidos
library(nortest)

cat("Prueba de normalidad para distancia transformada corregida\n")
```

```
## Prueba de normalidad para distancia transformada corregida
```

```r
ad.test(dist_trans)
```

```
##
##  Anderson-Darling normality test
##
## data:  dist_trans
## A = 0.14905, p-value = 0.9613
```

```r
# Histograma de la distancia transformada corregida
hist(dist_trans, freq=FALSE, main="Histograma de Distancia Transformada
Corregida", col="lightgray")
lines(density(dist_trans), col="red", lwd=2)
curve(dnorm(x, mean=mean(dist_trans), sd=sd(dist_trans)),
      from=min(dist_trans), to=max(dist_trans), add=TRUE, col="blue",
lwd=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. Con la mejor transformación (punto 2), realiza la regresión lineal simple entre la mejor transformación (exacta o aproximada) y la variable velocidad:

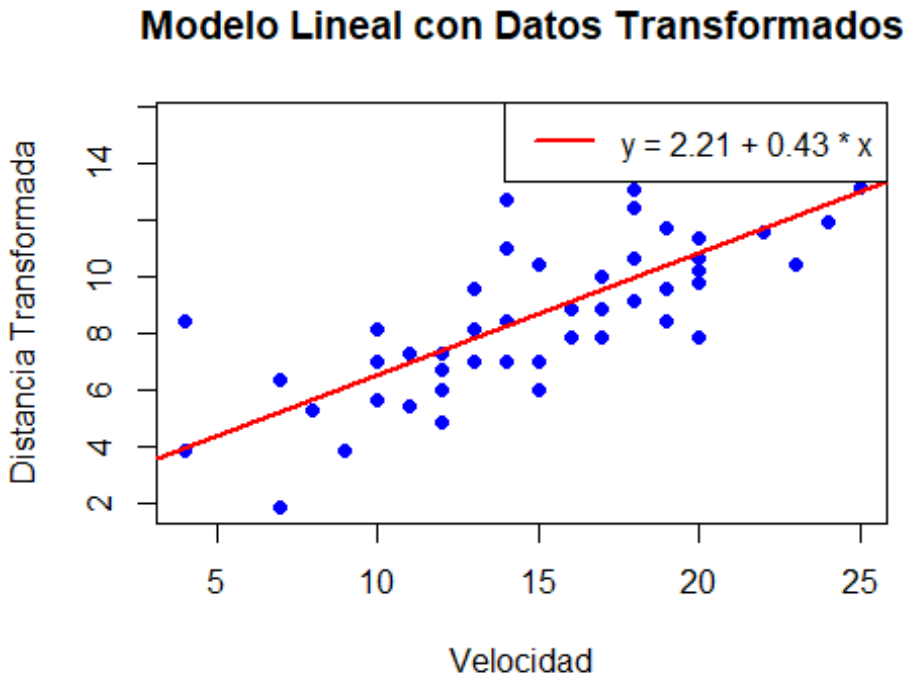
Pues la variable que si necesita una transformacion es la de distancia ya que desde el inicio su qqplt mostraba datos atipicos que podrian dañar los resultados, alterando las graficas, la mejor transformacion fue la de box-cox de distancia ya que fue la más efectiva para mejorar la normalida.

Escribe el modelo lineal para la transformación. Grafica los datos y el modelo lineal (ecuación) de la transformación elegida vs velocidad. Analiza significancia del modelo (individual, conjunta y coeficiente de correlación)

```
# Ajustar el modelo lineal usando la distancia transformada
modelo_transformado <- lm(dist_trans ~ speed)

# Mostrar el resumen del modelo
summary(modelo_transformado)

##
## Call:
## lm(formula = dist_trans ~ speed)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -3.3478 -1.1761 -0.1681  1.0007  4.5109
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.21077    0.78339   2.822  0.00692 **
## speed        0.43202    0.04816   8.970 7.85e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.783 on 48 degrees of freedom
## Multiple R-squared:  0.6263, Adjusted R-squared:  0.6186
## F-statistic: 80.46 on 1 and 48 DF,  p-value: 7.853e-12
```

```r
# Graficar los datos y el modelo lineal
plot(cars$speed, dist_trans, main = "Modelo Lineal con Datos
Transformados",
     xlab = "Velocidad", ylab = "Distancia Transformada", pch = 19, col =
"blue")

# Añadir la línea de regresión ajustada
abline(modelo_transformado, col = "red", lwd = 2)

# Mostrar la ecuación del modelo en la gráfica
coef_modelo <- coef(modelo_transformado)
eq <- paste("y =", round(coef_modelo[1], 2), "+", round(coef_modelo[2],
2), "* x")
legend("topright", legend = eq, col = "red", lwd = 2)
```

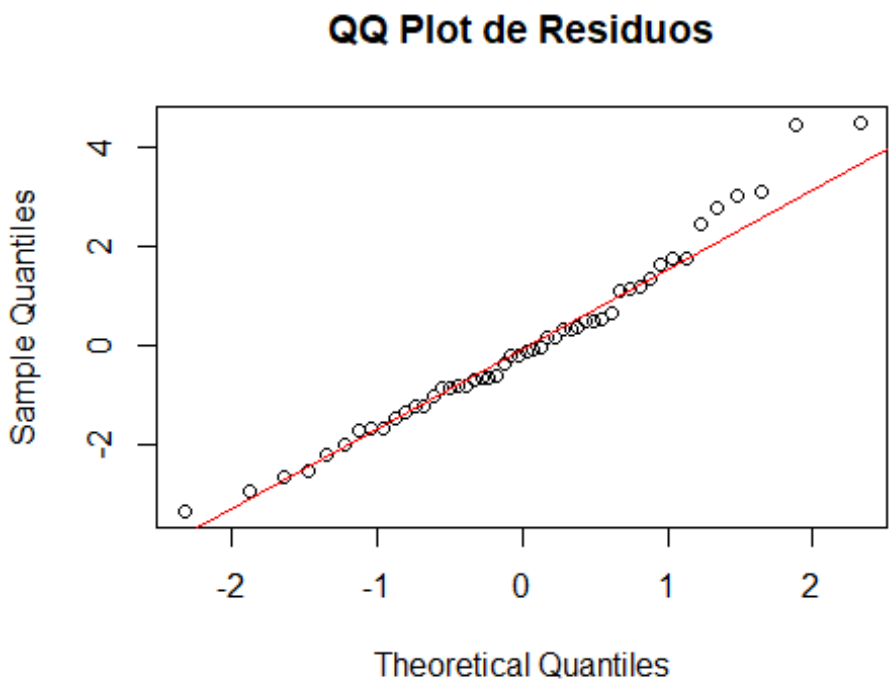


```r
# Mostrar el resumen del modelo
summary(modelo_transformado)
```

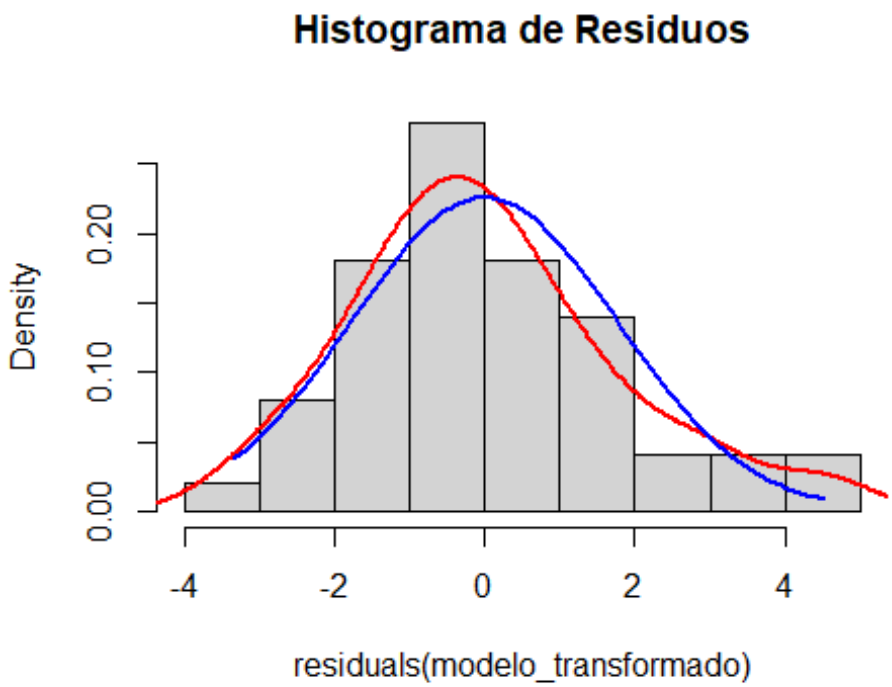
```
##
## Call:
## lm(formula = dist_trans ~ speed)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3478 -1.1761 -0.1681  1.0007  4.5109
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.21077    0.78339   2.822  0.00692 **
## speed        0.43202    0.04816   8.970 7.85e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.783 on 48 degrees of freedom
## Multiple R-squared:  0.6263, Adjusted R-squared:  0.6186
## F-statistic: 80.46 on 1 and 48 DF,  p-value: 7.853e-12
```

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.

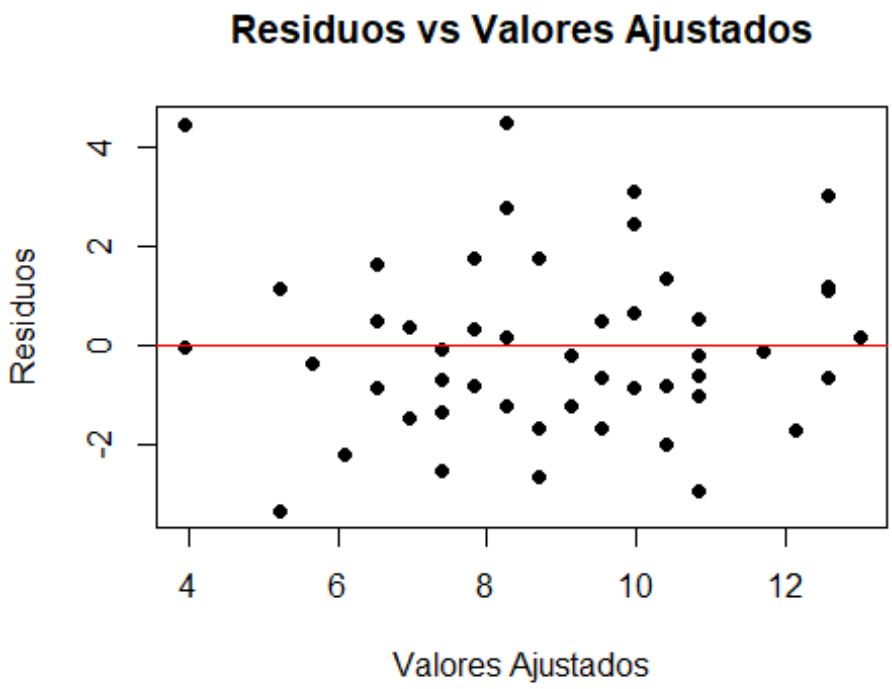
```r
# QQ Plot de los residuos
qqnorm(residuals(modelo_transformado), main="QQ Plot de Residuos")
qqline(residuals(modelo_transformado), col="red")
```



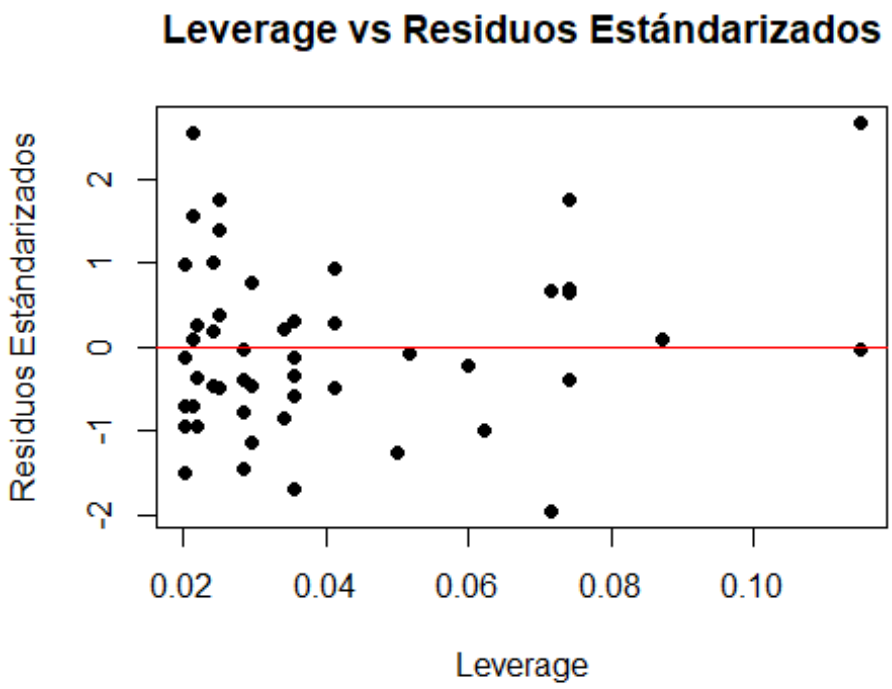

```r
# Histograma de los residuos con curva teórica
hist(residuals(modelo_transformado), freq=FALSE, main="Histograma de
Residuos", col="lightgray")
lines(density(residuals(modelo_transformado)), col="red", lwd=2)
curve(dnorm(x, mean=mean(residuals(modelo_transformado)),
sd=sd(residuals(modelo_transformado))),
      from=min(residuals(modelo_transformado)),
to=max(residuals(modelo_transformado)), add=TRUE, col="blue", lwd=2)
```



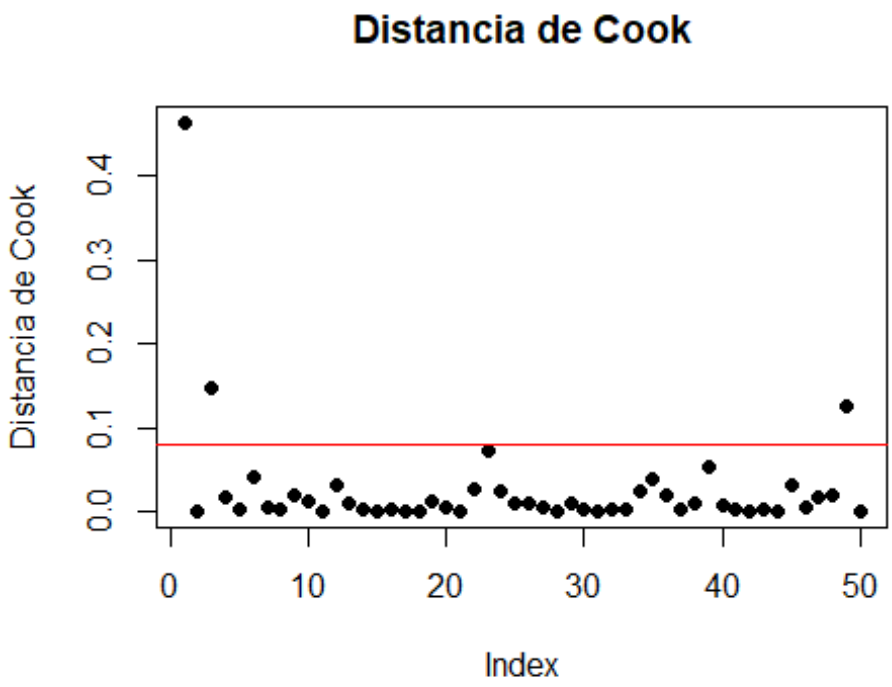

```r
# Prueba de normalidad (Anderson-Darling)
library(nortest)
ad.test(residuals(modelo_transformado))

##
##  Anderson-Darling normality test
##
## data:  residuals(modelo_transformado)
## A = 0.44371, p-value = 0.2749

# Gráfico de dispersión de los residuos frente a los valores ajustados
plot(fitted(modelo_transformado), residuals(modelo_transformado),
     main="Residuos vs Valores Ajustados",
     xlab="Valores Ajustados", ylab="Residuos", pch=19)
abline(h=0, col="red")
```




```r
# Prueba de Breusch-Pagan para homocedasticidad
library(lmtest)
bptest(modelo_transformado)

##
##  studentized Breusch-Pagan test
##
## data:  modelo_transformado
## BP = 1.9223, df = 1, p-value = 0.1656

# Prueba de Durbin-Watson para independencia de los residuos
dwtest(modelo_transformado)

##
##  Durbin-Watson test
##
## data:  modelo_transformado
## DW = 1.7724, p-value = 0.1681
## alternative hypothesis: true autocorrelation is greater than 0

# Gráfico de Leverage vs Residuos Estándarizados
plot(hatvalues(modelo_transformado), rstandard(modelo_transformado),
     main="Leverage vs Residuos Estándarizados",
     xlab="Leverage", ylab="Residuos Estándarizados", pch=19)
abline(h=0, col="red")
```




```r
# Gráfico de la distancia de Cook
plot(cooks.distance(modelo_transformado),
     main="Distancia de Cook",
     ylab="Distancia de Cook", pch=19)
abline(h = 4 / length(cars$speed), col="red")  # Línea de referencia para datos influyentes
```




```r
# Resumen de las métricas
influential_points <- which(cooks.distance(modelo_transformado) > 4 /
length(cars$speed))
print("Puntos influyentes:")
```

```
## [1] "Puntos influyentes:"
```

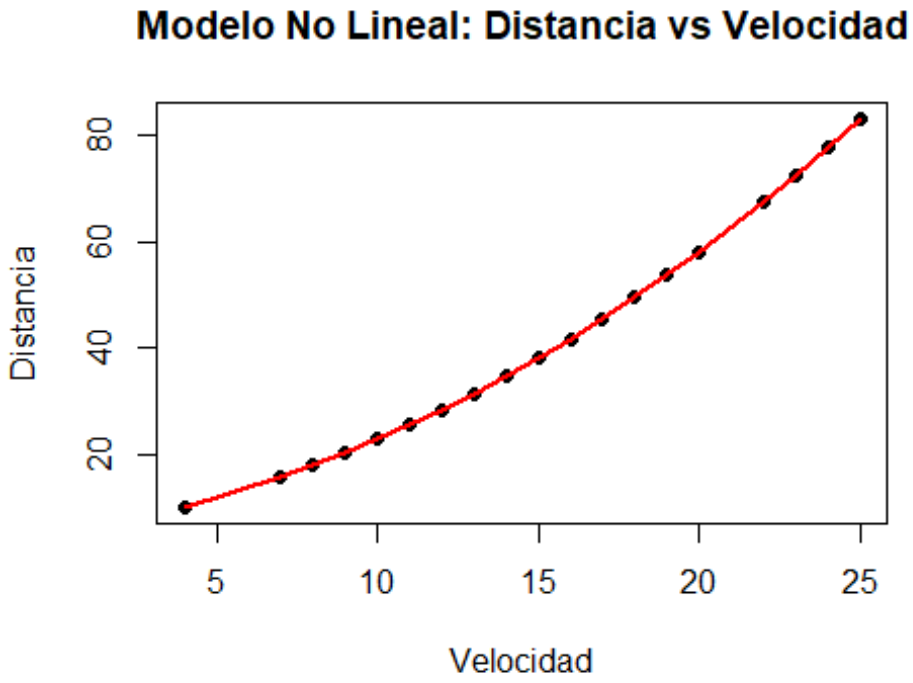
```r
print(influential_points)
```

```
##  1  3 49
##  1  3 49
```

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).

```r
# Despejar la transformación Box-Cox
lambda <- lambda_optimodist
inverse_transformation <- function(y_trans, lambda) {
  if (lambda == 0) {
    return(exp(y_trans))  # Para logaritmo
  } else {
    return((lambda * y_trans + 1)^(1 / lambda))  # Para Box-Cox
  }
}

# Obtener los valores ajustados transformados
fitted_values_trans <- fitted(modelo_transformado)
```

```r
# Despejar los valores ajustados para obtener la distancia
fitted_values <- inverse_transformation(fitted_values_trans, lambda)

plot(cars$speed, fitted_values, main="Modelo No Lineal: Distancia vs
Velocidad",
     xlab="Velocidad", ylab="Distancia", pch=19)

# Añadir la línea del modelo ajustado
lines(cars$speed, fitted_values, col="red", lwd=2)  # Líneas del modelo
ajustado en rojo
```



Grafica los datos y el modelo de la distancia en función de la velocidad. Gracias a la transformacion de box-cox mejora la normalidad y ayuda a reducir el sesgo y la curtosis, ahora el modelo si significativo ya que el p valor es bajo al igual que la R cuadrada muestra la variabilidad de la distancia transformada. Y con la validez del modelo el analisis de los residuos confirma que si se cumple con la normalidad, homocedasticidad e independencia.

##Parte 4: Conclusión

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

El mejor modelo es el de la transformacion con box-cox ya que este es el mejor en terminos de ajuste de datos y cumplimientos del modelo de regresion como la normalidad de los residuos, homocedasticidad e independencia. Siendo este más dificl de interpreta, ofreciendo una mejor descripcion de la relacion no lineal entre la velocidad y la distancia. Los probelmas que este modelo puede enfrentar serian la complejidad de su interpretacion y una posible influencia de los datos atipicos.