Transformaciones

Erika Martínez Meneses

2024-08-14

Lectura de Datos

```
file.choose()
```

```
## [1] "C:\\Users\\erika\\Documents\\Agos-Dic2024\\Estadística\\mc-donald
s-menu.csv"
```

```
library(readr)
data <- read_csv("C:\\Users\\erika\\Documents\\Agos-Dic2024\\Estadística\
\mc-donalds-menu.csv")
```

```
## Rows: 260 Columns: 24
## ── Column specification ────────────────────────────────────────────────
─────────
## Delimiter: ","
## chr  (3): Category, Item, Serving Size
## dbl (21): Calories, Calories from Fat, Total Fat, Total Fat (% Daily V
alue),...
##
## ℹ Use `spec()` to retrieve the full column specification for this data
.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet th
is message.
```

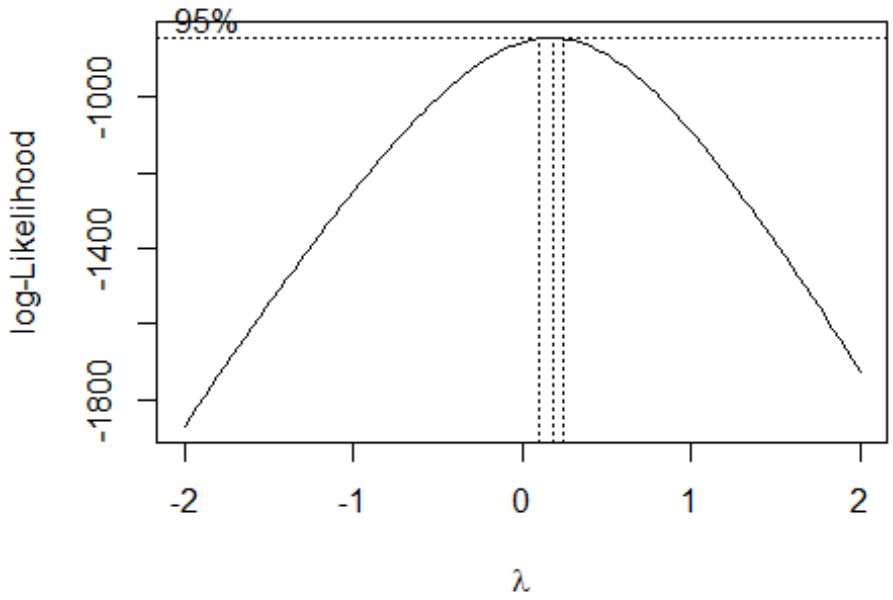
```
head(data)
```

```
## # A tibble: 6 × 24
##   Category  Item         `Serving Size` Calories `Calories from Fat` `
Total Fat`
##   <chr>     <chr>        <chr>             <dbl>               <dbl>
<dbl>
## 1 Breakfast Egg McMuffin 4.8 oz (136 g)      300                 120
13
## 2 Breakfast Egg White D… 4.8 oz (135 g)      250                  70
8
## 3 Breakfast Sausage McM… 3.9 oz (111 g)      370                 200
23
## 4 Breakfast Sausage McM… 5.7 oz (161 g)      450                 250
28
## 5 Breakfast Sausage McM… 5.7 oz (161 g)      400                 210
23
## 6 Breakfast Steak & Egg… 6.5 oz (185 g)      430                 210
```

```
23
## # ℹ 18 more variables: `Total Fat (% Daily Value)` <dbl>,
## #   `Saturated Fat` <dbl>, `Saturated Fat (% Daily Value)` <dbl>,
## #   `Trans Fat` <dbl>, Cholesterol <dbl>, `Cholesterol (% Daily Value)
` <dbl>,
## #   Sodium <dbl>, `Sodium (% Daily Value)` <dbl>, Carbohydrates <dbl>,
## #   `Carbohydrates (% Daily Value)` <dbl>, `Dietary Fiber` <dbl>,
## #   `Dietary Fiber (% Daily Value)` <dbl>, Sugars <dbl>, Protein <dbl>
,
## #   `Vitamin A (% Daily Value)` <dbl>, `Vitamin C (% Daily Value)` <db
l>, …
```

Box-Cox

```
library(MASS)
Cholesterol <- data$Cholesterol
bc<-boxcox((Cholesterol+1)~1)
```




```
l=bc$x[which.max(bc$y)]

l

## [1] 0.1818182
```

Análisis de la normalidad de las transformaciones obtenidas con los datos originales

Original

```
library(e1071)
summary(Cholesterol)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    5.00   35.00   54.94   65.00  575.00
```

```
print("Curtosis")
```

```
## [1] "Curtosis"
```

```
kurtosis(Cholesterol)
```

```
## [1] 16.87947
```

```
print("Sesgo")
```

```
## [1] "Sesgo"
```

```
skewness(Cholesterol)
```

```
## [1] 3.755186
```

Aproximado

$$\sqrt{x+1}$$

```
library(e1071)
col1=sqrt(Cholesterol+1)
summary(col1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.449   6.000   6.107   8.124  24.000
```

```
print("Curtosis")
```

```
## [1] "Curtosis"
```

```
kurtosis(col1)
```

```
## [1] 3.383817
```

```
print("Sesgo")
```

```
## [1] "Sesgo"
```

```
skewness(col1)
```

```
## [1] 1.473229
```

Exacto

$$\frac{(x+1)^{0.1818}-1}{0.1818}$$

```
library(e1071)
col2=((Cholesterol+1)^l-1)/l
summary(col2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   2.118   5.052   4.489   6.281  11.969
```

```
print("Curtosis")
```

```
## [1] "Curtosis"
```

```
kurtosis(col2)
```

```
## [1] -0.21773
```

```
print("Sesgo")
```

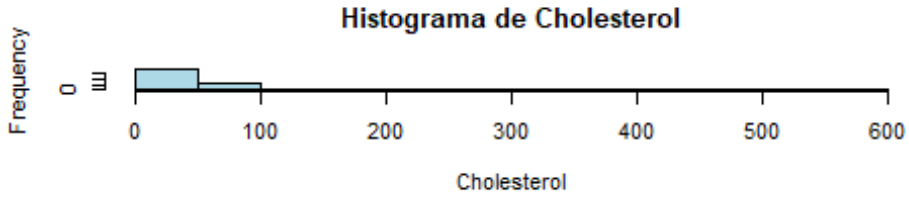
```
## [1] "Sesgo"
```

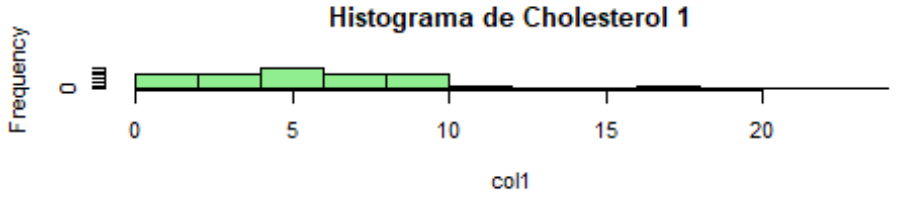
```
skewness(col2)
```

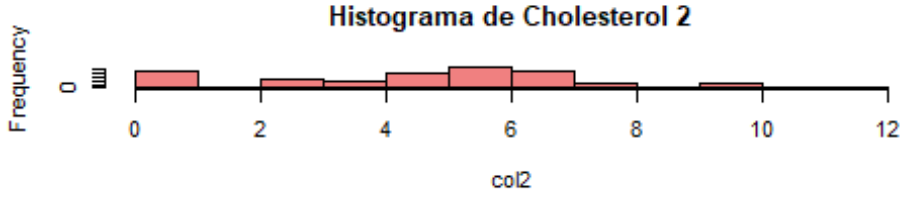
```
## [1] -0.007037008
```

Histogramas

```
par(mfrow=c(3,1))
hist(Cholesterol,col="lightblue",main="Histograma de Cholesterol") # Orig
inal
hist(col1,col="lightgreen",main="Histograma de Cholesterol 1") # Aproxima
do
hist(col2,col="lightcoral",main="Histograma de Cholesterol 2") # Exacto
```







Prueba de Normalidad

Original

```
library(nortest)
D=ad.test(Cholesterol)
D$p.value

## [1] 3.7e-24
```

Aproximado

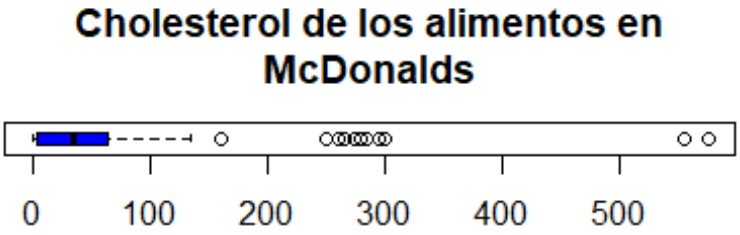
```
library(nortest)
D=ad.test(col1)
D$p.value

## [1] 3.274337e-16
```

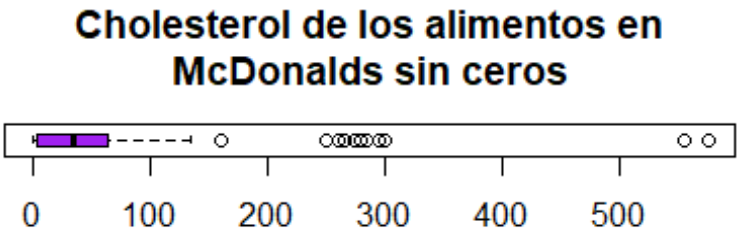
Exacto

```
library(nortest)
D=ad.test(col2)
D$p.value

## [1] 4.087055e-12
```

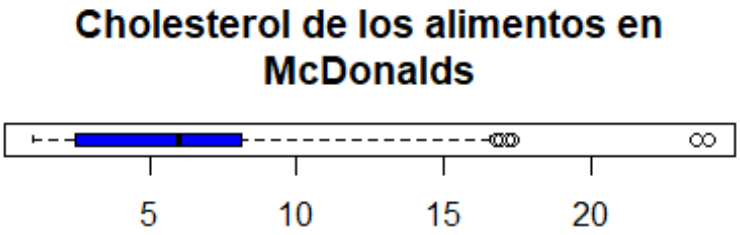
Anomalías

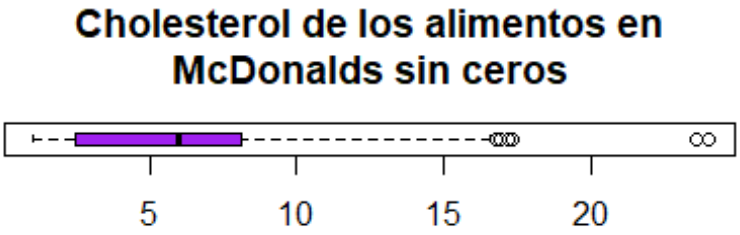
```
data2=subset(data,Cholesterol>0)
par(mfrow=c(2,1))
boxplot(Cholesterol, horizontal = TRUE,col="blue", main="Cholesterol de l
os alimentos en
McDonalds")
boxplot(Cholesterol, horizontal = TRUE,col="purple", main="Cholesterol de
los alimentos en
McDonalds sin ceros")
```



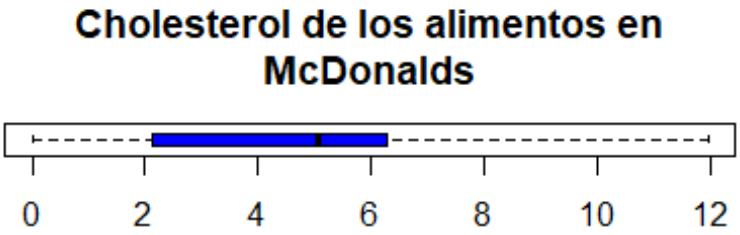


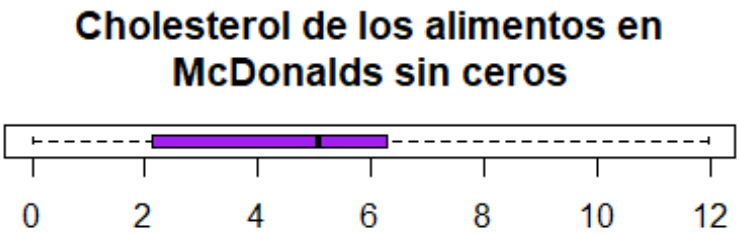

```
data2=subset(data,col1>0)
par(mfrow=c(2,1))
boxplot(col1, horizontal = TRUE,col="blue", main="Cholesterol de los alim
entos en
McDonalds")
boxplot(col1, horizontal = TRUE,col="purple", main="Cholesterol de los al
imentos en
McDonalds sin ceros")
```





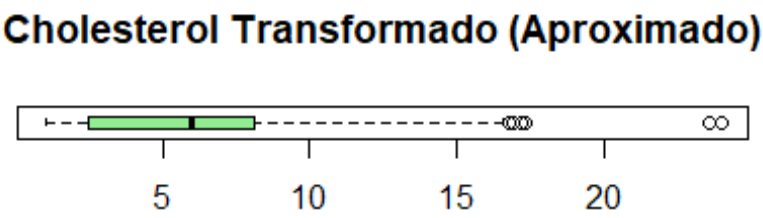

```
data2=subset(data,col2>0)
par(mfrow=c(2,1))
boxplot(col2, horizontal = TRUE,col="blue", main="Cholesterol de los alim
entos en
McDonalds")
boxplot(col2, horizontal = TRUE,col="purple", main="Cholesterol de los al
imentos en
McDonalds sin ceros")
```

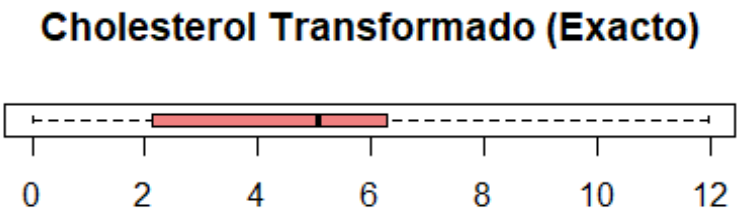




Eliminación de ceros anómalos

```
data_clean <- subset(data, Cholesterol > 0)

# Datos transformados (Aproximado y Exacto)
data_clean_approx <- subset(data, sqrt(Cholesterol + 1) > 0)
data_clean_exact <- subset(data, ((Cholesterol + 1)^l - 1) / l > 0)

# Boxplot para los datos transformados
par(mfrow = c(2, 1))
boxplot(sqrt(Cholesterol + 1), horizontal = TRUE, col = "lightgreen", mai
n = "Cholesterol Transformado (Aproximado)")
boxplot(((Cholesterol + 1)^l - 1) / l, horizontal = TRUE, col = "lightcor
al", main = "Cholesterol Transformado (Exacto)")
```





Transformación de Yeo Johnson

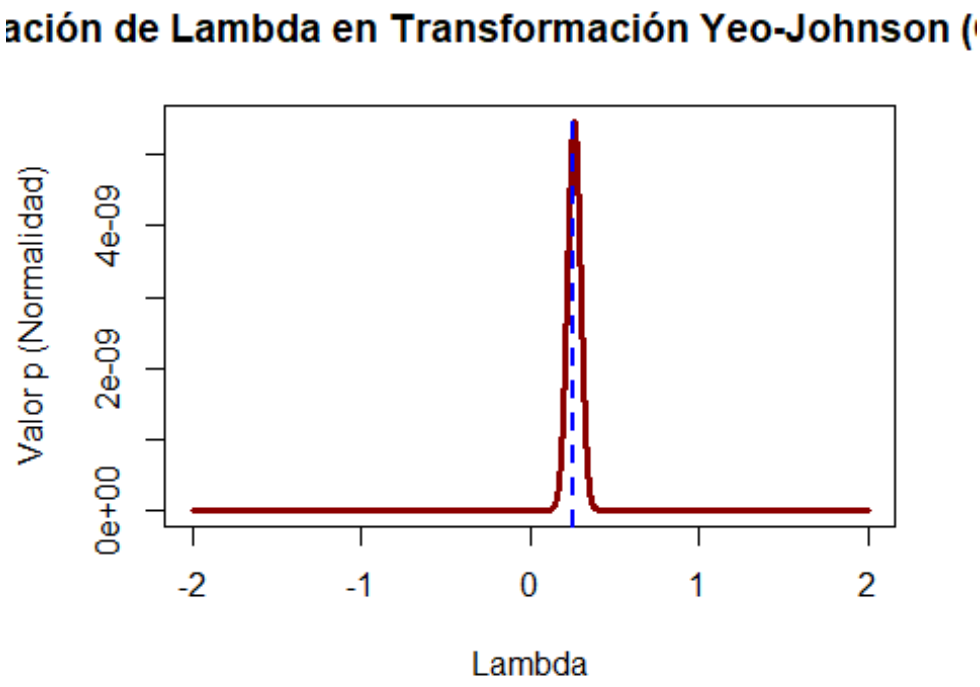
```
library(VGAM)

## Loading required package: stats4

## Loading required package: splines

library(nortest)

# Función para optimizar lambda y aplicar la transformación Yeo-Johnson
optimize_yeojohnson <- function(data) {
  # Definir el rango de lambda propuesto
  lp <- seq(-2, 2, 0.001)  # Puedes ajustar el rango según sea necesario
  nlp <- length(lp)

  # Inicializar una matriz para almacenar los valores de lambda y los val
ores p correspondientes
  D <- matrix(NA, ncol = 2, nrow = nlp)
  colnames(D) <- c("Lambda", "p-value")

  # Calcular la transformación Yeo-Johnson para cada lambda y realizar la
prueba de Anderson-Darling
  for (i in 1:nlp) {
    # Aplicar la transformación Yeo-Johnson para el valor actual de lambd
a
```

```
    transformed <- yeo.johnson(data, lambda = lp[i])

    # Realizar la prueba de Anderson-Darling en los datos transformados
    p_value <- ad.test(transformed)$p.value

    # Almacenar el valor de lambda y el valor p correspondiente
    D[i,] <- c(lp[i], p_value)
  }

  # Convertir la matriz en un data frame para facilitar su manejo
  D <- as.data.frame(D)

  # Encontrar el valor de lambda que maximiza el valor p
  best_lambda <- D[which.max(D$`p-value`), "Lambda"]
  max_p_value <- max(D$`p-value`)

  # Aplicar la transformación Yeo-Johnson con el mejor lambda encontrado
  best_transformed <- yeo.johnson(data, lambda = best_lambda)

  # Devolver los resultados
  list(transformed = best_transformed, lambda = best_lambda, p_value = ma
x_p_value, D = D)
}

# Aplicar la función a Cholesterol, col1, y col2
result_cholesterol <- optimize_yeojohnson(Cholesterol + 1)
result_col1 <- optimize_yeojohnson(col1 + 1)
result_col2 <- optimize_yeojohnson(col2 + 1)

# Mostrar los mejores lambda y los valores p máximos
cat("Cholesterol: Mejor Lambda =", result_cholesterol$lambda, ", Valor p
máximo =", result_cholesterol$p_value, "\n")

## Cholesterol: Mejor Lambda = 0.26 , Valor p máximo = 5.469079e-09

cat("col1: Mejor Lambda =", result_col1$lambda, ", Valor p máximo =", res
ult_col1$p_value, "\n")

## col1: Mejor Lambda = 0.44 , Valor p máximo = 5.738742e-09

cat("col2: Mejor Lambda =", result_col2$lambda, ", Valor p máximo =", res
ult_col2$p_value, "\n")

## col2: Mejor Lambda = 1.444 , Valor p máximo = 3.756036e-09
```

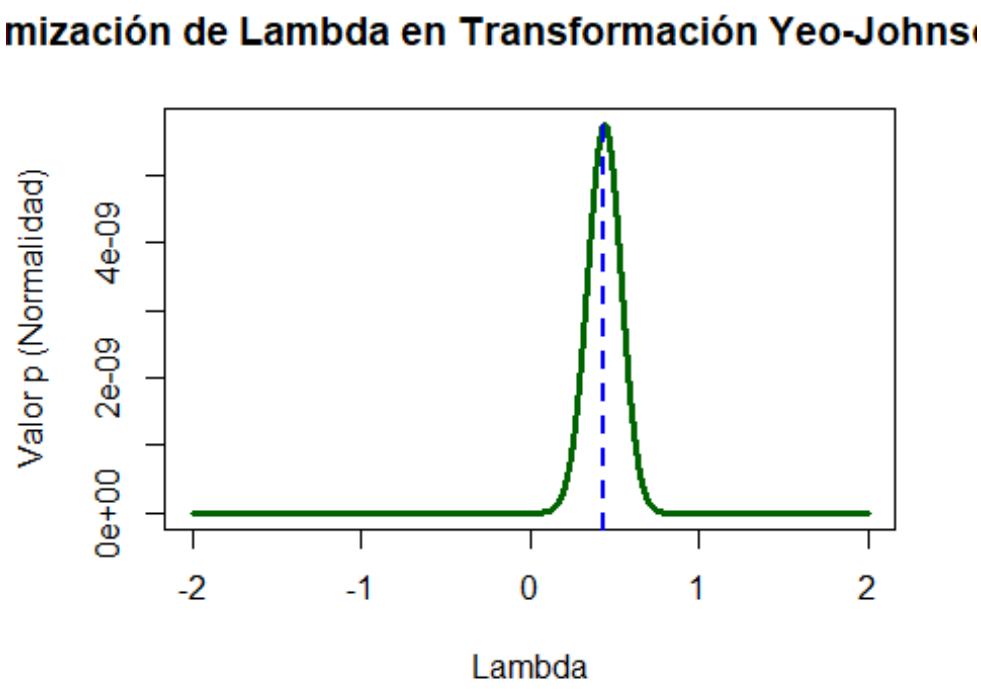
Visualización

```
# Graficar Lambda vs. Valor p para Cholesterol
plot(result_cholesterol$D$Lambda, result_cholesterol$D$`p-value`, type =
"l", col = "darkred", lwd = 3,
     xlab = "Lambda", ylab = "Valor p (Normalidad)", main = "Optimización
```

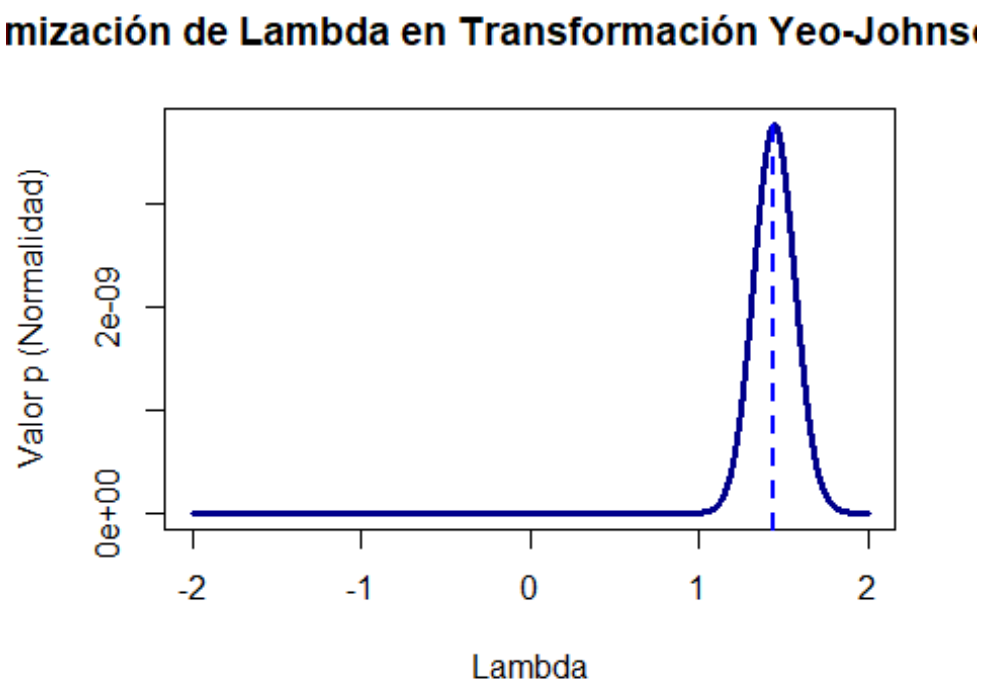
```
de Lambda en Transformación Yeo-Johnson (Cholesterol)")
abline(v = result_cholesterol$lambda, col = "blue", lwd = 2, lty = 2)  # Línea para el mejor lambda
```




```
# Graficar Lambda vs. Valor p para col1
plot(result_col1$D$Lambda, result_col1$D$`p-value`, type = "l", col = "darkgreen", lwd = 3,
     xlab = "Lambda", ylab = "Valor p (Normalidad)", main = "Optimización de Lambda en Transformación Yeo-Johnson (col1)")
abline(v = result_col1$lambda, col = "blue", lwd = 2, lty = 2)  # Línea para el mejor lambda
```




```r
# Graficar Lambda vs. Valor p para col2
plot(result_col2$D$Lambda, result_col2$D$`p-value`, type = "l", col = "da
rkblue", lwd = 3,
     xlab = "Lambda", ylab = "Valor p (Normalidad)", main = "Optimización
de Lambda en Transformación Yeo-Johnson (col2)")
abline(v = result_col2$lambda, col = "blue", lwd = 2, lty = 2)  # Línea p
ara el mejor Lambda
```



Ecuación del Modelo Yeo-Johnson

$$T_{original}(x) = \frac{(x+1)^{0.26} - 1}{0.26}$$

$$T_{aproximado}(x) = \frac{(x)^{0.44} - 1}{0.44}$$

$$T_{exacto}(x) = \frac{(x)^{1.444} - 1}{1.444}$$

Análisis de Normalidad para Yeo-Johnson

Usando los valores óptimos de lambda obtenidos para cada variable

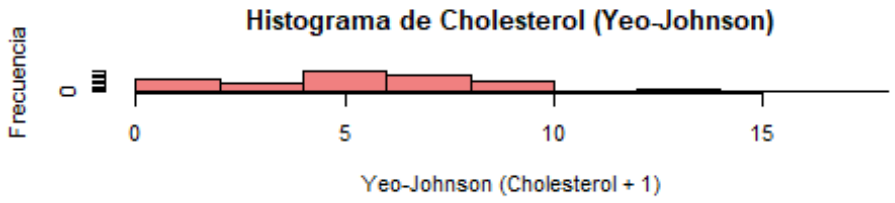
```
library(e1071)

# Función para obtener medidas estadísticas
get_measures <- function(data) {
  data.frame(
    Min = min(data),
    Max = max(data),
    Mean = mean(data),
    Median = median(data),
```

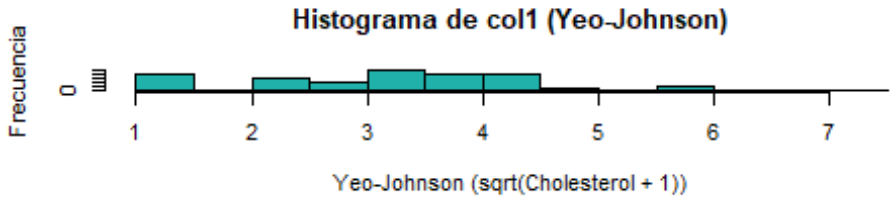
```
    Q1 = quantile(data, 0.25),
    Q3 = quantile(data, 0.75),
    Skewness = skewness(data),
    Kurtosis = kurtosis(data)
  )
}

# Medidas estadísticas
measures_cholesterol <- get_measures(result_cholesterol$transformed)
measures_col1 <- get_measures(result_col1$transformed)
measures_col2 <- get_measures(result_col2$transformed)

print("Medidas estadísticas para Cholesterol (Yeo-Johnson):")

## [1] "Medidas estadísticas para Cholesterol (Yeo-Johnson):"

print(measures_cholesterol)

##           Min      Max     Mean Median       Q1       Q3  Skewness  Ku
rtosis
## 25% 0.7595335 16.24165 5.597797 5.9885 2.532839 7.630257 0.4950605 0.4
358535

print("Medidas estadísticas para col1 (Yeo-Johnson):")

## [1] "Medidas estadísticas para col1 (Yeo-Johnson):"

print(measures_col1)

##          Min     Max     Mean   Median       Q1       Q3 Skewness  Kur
tosis
## 25% 1.412639 7.25821 3.253243 3.401502 2.110599 4.020929 0.480642 0.42
09133

print("Medidas estadísticas para col2 (Yeo-Johnson):")

## [1] "Medidas estadísticas para col2 (Yeo-Johnson):"

print(measures_col2)

##          Min      Max     Mean   Median       Q1       Q3 Skewness  Ku
rtosis
## 25% 1.191649 30.49873 10.23973 10.93209 4.653738 13.96829 0.530801 0.5
672009
```

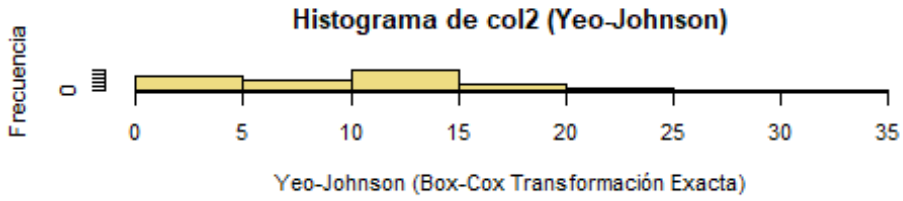
Histogramas

```
# Histogramas de las transformaciones Yeo-Johnson
par(mfrow = c(3, 1))  # Organizar en una matriz de 3x1

# Histograma de Yeo-Johnson aplicado a Cholesterol
hist(result_cholesterol$transformed, col = "lightcoral",
```

```
     main = "Histograma de Cholesterol (Yeo-Johnson)",
     xlab = "Yeo-Johnson (Cholesterol + 1)", ylab = "Frecuencia")

# Histograma de Yeo-Johnson aplicado a col1
hist(result_col1$transformed, col = "lightseagreen",
     main = "Histograma de col1 (Yeo-Johnson)",
     xlab = "Yeo-Johnson (sqrt(Cholesterol + 1))", ylab = "Frecuencia")

# Histograma de Yeo-Johnson aplicado a col2
hist(result_col2$transformed, col = "lightgoldenrod",
     main = "Histograma de col2 (Yeo-Johnson)",
     xlab = "Yeo-Johnson (Box-Cox Transformación Exacta)", ylab = "Frecue
ncia")
```







Prueba de normalidad

```
# Prueba de normalidad Anderson-Darling para las transformaciones Yeo-Joh
nson
ad_cholesterol <- ad.test(result_cholesterol$transformed)
ad_col1 <- ad.test(result_col1$transformed)
ad_col2 <- ad.test(result_col2$transformed)

cat("Cholesterol (Yeo-Johnson): p-value =", ad_cholesterol$p.value, "\n")

## Cholesterol (Yeo-Johnson): p-value = 5.469079e-09
```

```r
cat("col1 (Yeo-Johnson): p-value =", ad_col1$p.value, "\n")
```

```
## col1 (Yeo-Johnson): p-value = 5.738742e-09
```

```r
cat("col2 (Yeo-Johnson): p-value =", ad_col2$p.value, "\n")
```

```
## col2 (Yeo-Johnson): p-value = 3.756036e-09
```

Conclusiones

La mejor transformación de los datos...

Se prefiere una transformación que minimice la complejidad del modelo y mantenga la interpretación de los resultados. Entre Box-Cox y Yeo-Johnson, Yeo-Johnson es más flexible ya que puede manejar datos negativos y cero. Considerando lo anterior, la transformación de Yeo-Johnson podría ser la mejor opción debido a su flexibilidad y su capacidad para normalizar la distribución de los datos. Así mismo, centrandonos en los valores p obtenidos en ambas transformaciones; aunque ninguno de los valores p es suficientemente alto para aceptar la hipótesis nula (H_0) de normalidad, la transformación de Yeo-Johnson logra acercar más los datos a una distribución normal en comparación con Box-Cox, ya que todos los valores p de Yeo-Johnson son más altos que los obtenidos con Box-Cox.

Ventajas y desventajas de los modelos de Box Cox y de Yeo Johnson

Box-Cox:

Ventajas:

- Puede transformar datos positivos en normalidad, lo que es útil para muchos modelos estadísticos.
- Es ampliamente reconocido y utilizado en análisis estadístico.

Desventajas:

- No puede manejar valores negativos o ceros, lo que limita su aplicabilidad.
- Requiere que los datos sean estrictamente positivos.

Yeo-Johnson:

Ventajas:

- Puede manejar datos con valores negativos y ceros, proporcionando mayor flexibilidad.
- Similar al Box-Cox en su capacidad para normalizar los datos.

Desventajas:

- Puede ser menos intuitivo en su interpretación que el Box-Cox.

- Menos conocido y utilizado en comparación con el Box-Cox.

Diferencias entre la transformación y el escalamiento de los datos

1) Su propósito: Transformación: Se utiliza para modificar la distribución de los datos, haciendo que se asemejen más a una distribución normal o para estabilizar la varianza. Escalamiento: Se utiliza para cambiar la escala de los datos, típicamente para que todas las características tengan un rango similar, sin alterar la forma de la distribución.

2) Aplicación: Transformación: Común en métodos de regresión y cuando los supuestos de normalidad son importantes. Escalamiento: Es fundamental en métodos que son sensibles a las magnitudes de los datos, como en algoritmos de machine learning como KNN o SVM.

3) Afecto a la Distribución: Transformación: Cambia la distribución de los datos. Escalamiento: Mantiene la forma de la distribución de los datos, solo ajusta la escala.

Transformación: Se utiliza cuando se necesita normalizar los datos, mejorar la linearidad o estabilizar la varianza. Es útil antes de aplicar técnicas que asumen normalidad.

Escalamiento: Se utiliza cuando los modelos son sensibles a la escala de las variables, como en métodos de machine learning que calculan distancias (KNN, SVM) o cuando se integran variables en diferentes unidades.