5.Transformaciones

Jacobo Hirsch Rodriguez

2024-08-16

```r
# Cargar las librerías necesarias
library(MASS)
library(car)
```

```
## Loading required package: carData
```

```r
library(ggplot2)
library(nortest)
library(moments)

# Cargar el dataset
mcdonalds <- read.csv("./mc-donalds-menu.csv") #leer la base de datos
```

ahora vamos a seleccionar una variable que no sea calorias, en este caso es sugars pero lo vamos a limpiar para utilizar el metodo box cox, como la transformacion box-cox no admite valores iguales a 0 o negativos, vamos a cambiar los 0s a valores muy cercanos

```r
# Reemplazar ceros por un pequeño valor positivo
mcdonalds$Sugars[mcdonalds$Sugars == 0] <- 0.01
```

```r
variable_original <- mcdonalds$Sugars
```

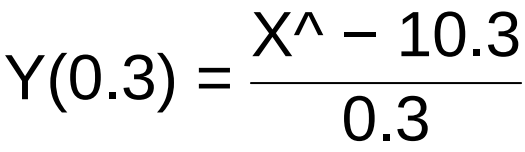
Utiliza la transformación Box-Cox. Utiliza el modelo exacto y el aproximado de acuerdo con las sugerencias de Box y Cox para la transformación

```r
  # Aplicar la transformación de Box-Cox
  boxcox_model <- boxcox(variable_original ~ 1, plotit = FALSE)

  # Encontrar el valor óptimo de lambda
  lambda_opt <- boxcox_model$x[which.max(boxcox_model$y)]

  # Transformación exacta
  variable_bc_exact <- (variable_original^lambda_opt - 1) / lambda_opt

  # Transformación aproximada (logarítmica)
  variable_bc_approx <- log(variable_original)

  # Mostrar el valor de lambda encontrado
  print(paste("Valor óptimo de lambda:", lambda_opt))
```

```
## [1] "Valor óptimo de lambda: 0.3"
```

Escribe las ecuaciones de los modelos encontrados.

```
library(latex2exp)
# Crear la expresión LaTeX
expr <- TeX(r'(Y(0.3) = \frac{X^{0.3} - 1}{0.3})')

# Crear un gráfico vacío
plot.new()

# Mostrar la expresión en el centro del gráfico
text(0.5, 0.5, labels = expr, cex = 2)
```



Analiza la normalidad de las transformaciones obtenidas con los datos originales. Utiliza como argumento de normalidad:

```
# Función para calcular estadísticas descriptivas
estadisticas_descriptivas <- function(x) {
  return(c(
    Min = min(x),
    Max = max(x),
    Media = mean(x),
    Mediana = median(x),
    Q1 = quantile(x, 0.25),
    Q3 = quantile(x, 0.75),
```

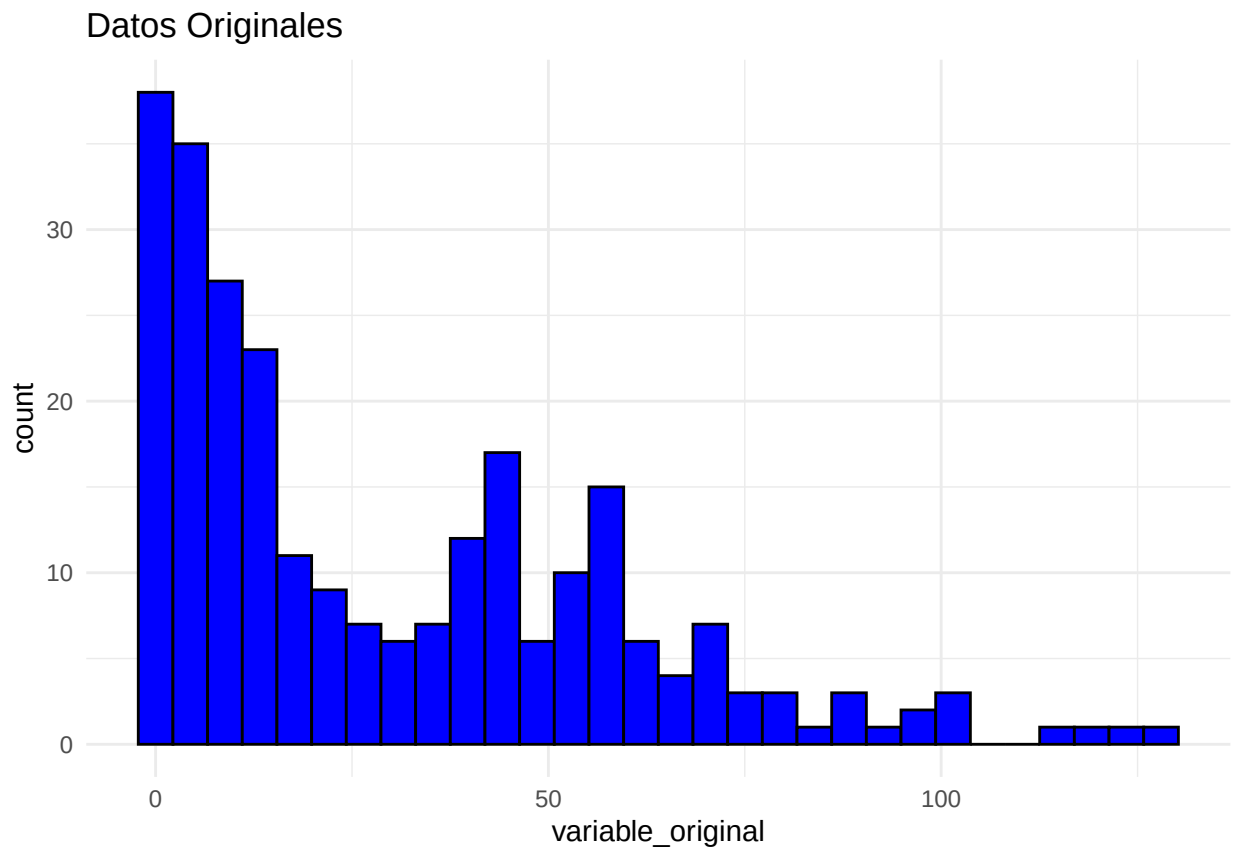
```
    Sesgo = skewness(x),
    Curtosis = kurtosis(x)
  ))
}

# Calcular estadísticas descriptivas
stats_original <- estadisticas_descriptivas(variable_original)
stats_bc_exact <- estadisticas_descriptivas(variable_bc_exact)
stats_bc_approx <- estadisticas_descriptivas(variable_bc_approx)

# Mostrar los resultados
stats_comparacion <- data.frame(Original = stats_original,
                                BoxCox_Exacto = stats_bc_exact,
                                BoxCox_Aproximado = stats_bc_approx)

print(stats_comparacion)
```

```
##            Original BoxCox_Exacto BoxCox_Aproximado
## Min        0.010000     -2.496038         -4.605170
## Max      128.000000     10.956980          4.852030
## Media     29.424038      4.492618          2.228792
## Mediana   17.500000      4.532648          2.861793
## Q1.25%     5.750000      2.296638          1.746179
## Q3.75%    48.000000      7.314252          3.871201
## Sesgo      1.026089     -0.434996         -1.836025
## Curtosis   3.487942      2.401569          5.600293
```

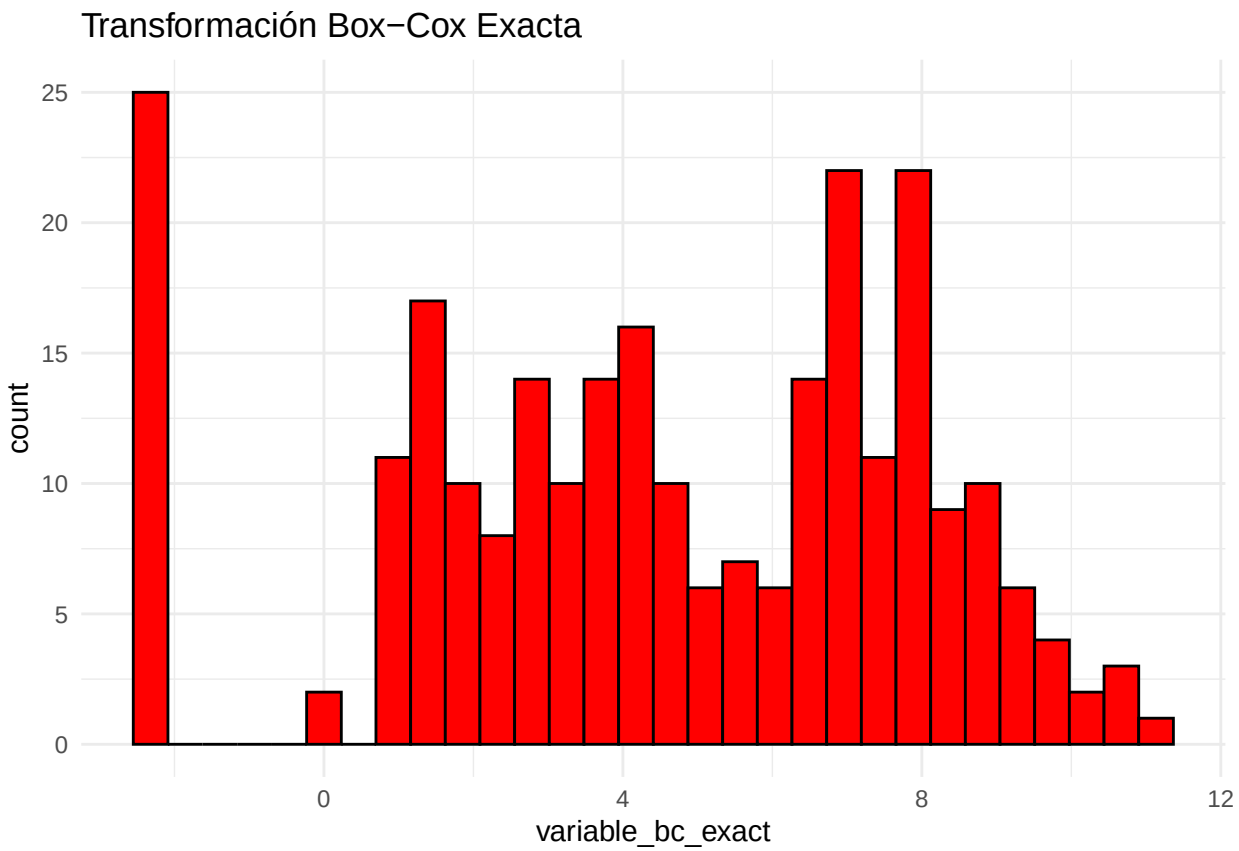
La transformación exacta de Box-Cox (Sesgo = -0.434996) parece mejorar la simetría de los datos, acercándolos más a una distribución normal. La transformación exacta de Box-Cox también mejora la curtosis, acercándola más al valor ideal de 3, lo que sugiere que los datos se comportan más como una distribución normal.

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

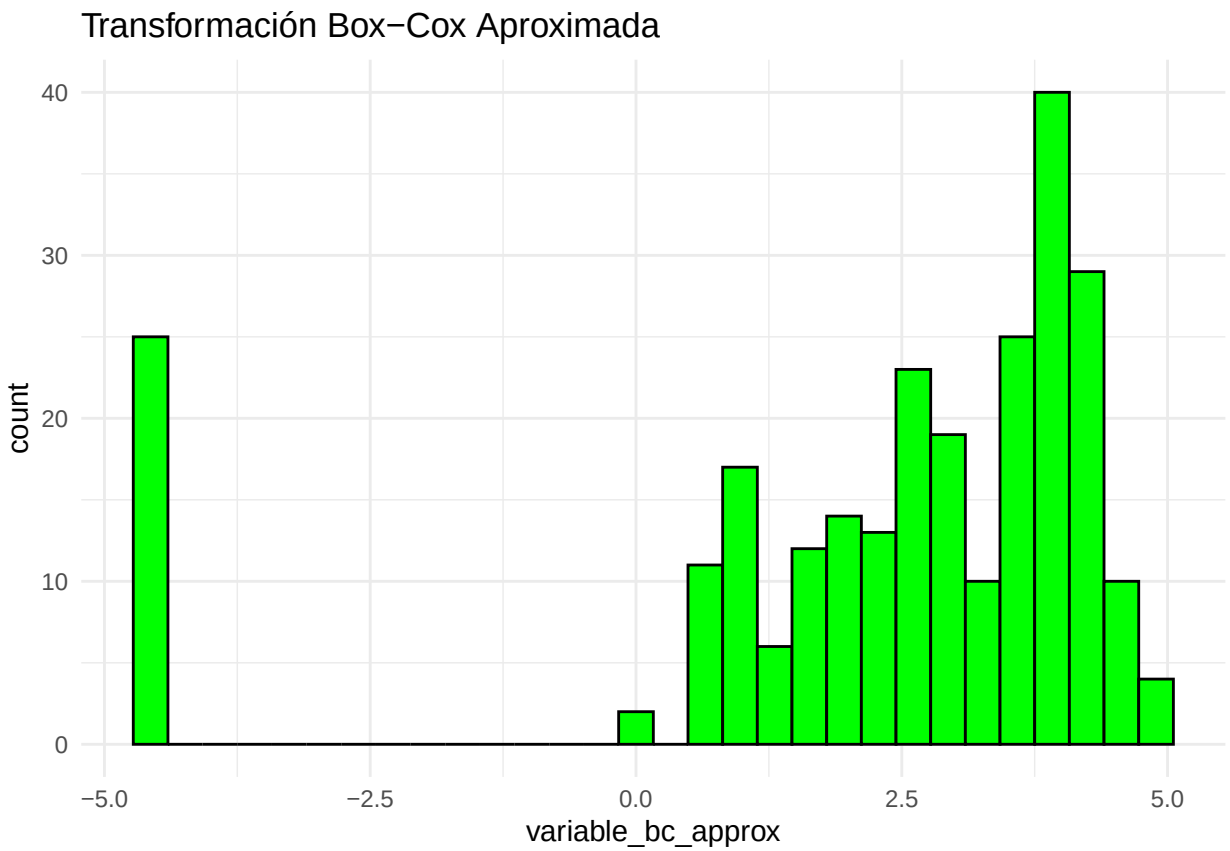
```
# Histograma para los datos originales
ggplot(data.frame(variable_original), aes(x = variable_original)) +
  geom_histogram(bins = 30, color = "black", fill = "blue") +
  ggtitle("Datos Originales") + theme_minimal()
```




```
# Histograma para la transformación de Box-Cox exacta
ggplot(data.frame(variable_bc_exact), aes(x = variable_bc_exact)) +
  geom_histogram(bins = 30, color = "black", fill = "red") +
  ggtitle("Transformación Box-Cox Exacta") + theme_minimal()
```




```r
# Histograma para la transformación de Box-Cox aproximada
ggplot(data.frame(variable_bc_approx), aes(x = variable_bc_approx)) +
  geom_histogram(bins = 30, color = "black", fill = "green") +
  ggtitle("Transformación Box-Cox Aproximada") + theme_minimal()
```



Realiza la prueba de normalidad de Anderson-Darling o de Jarque Bera para los datos transformados y los originales

```
# Prueba de normalidad de Anderson-Darling para los datos originales
ad_original <- ad.test(variable_original)
print(ad_original$p.value)
```

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

```
# Prueba de normalidad de Anderson-Darling para la transformación exacta de Box-Cox
ad_bc_exact <- ad.test(variable_bc_exact)
print(ad_bc_exact$p.value)
```

```
## [1] 1.841224e-08
```

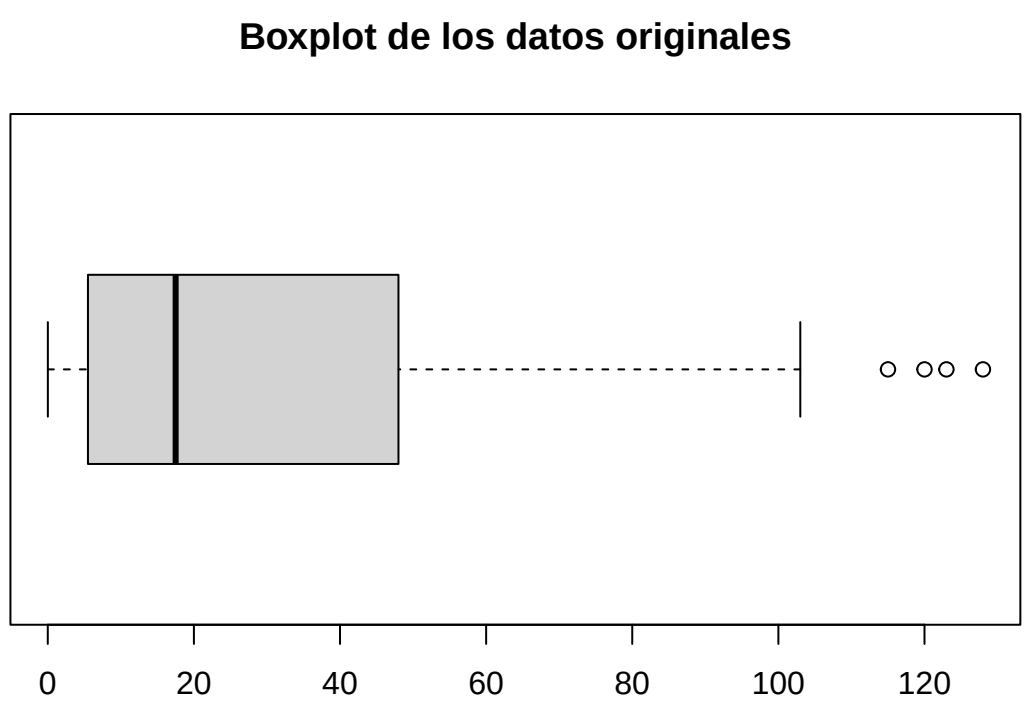
```
# Prueba de normalidad de Anderson-Darling para la transformación aproximada de Box-Cox
ad_bc_approx <- ad.test(variable_bc_approx)
print(ad_bc_approx$p.value)
```

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

Detecta anomalías y corrige tu base de datos (datos atípicos, ceros anámalos, etc).

a) Identificación de Valores Atípicos

```r
# Boxplot para detectar valores atípicos
boxplot(variable_original, main="Boxplot de los datos originales", horizontal = TRUE)
```




```r
# Extraer valores atípicos del boxplot
outliers_original <- boxplot.stats(variable_original)$out
print("Valores atípicos en los datos originales:")
```

```
## [1] "Valores atípicos en los datos originales:"
```

```r
print(outliers_original)
```

```
## [1] 123 120 115 128
```

```r
# Z-Score para detectar valores atípicos
z_scores_original <- scale(variable_original)
outliers_z_original <- variable_original[abs(z_scores_original) > 3]
print("Valores atípicos en los datos originales (Z-Score):")
```

```
## [1] "Valores atípicos en los datos originales (Z-Score):"
```

```r
print(outliers_z_original)
```

```
## [1] 123 120 128
```

b) Detección de Ceros Anómalos

```
# Contar cuántos ceros existen en la variable
ceros_anomalos <- sum(variable_original == 0)
print(paste("Número de ceros en los datos originales:", ceros_anomalos))
```

```
## [1] "Número de ceros en los datos originales: 0"
```

```
# Ver si estos ceros son anómalos revisando el contexto
ceros_anomalos_detalle <- variable_original[variable_original == 0]
print("Detalles de los ceros en los datos originales:")
```

```
## [1] "Detalles de los ceros en los datos originales:"
```

```
print(ceros_anomalos_detalle)
```

```
## numeric(0)
```

ahora vamos a correguir la base de datos:

a) manejo de valores atipicos

```
# Opción 1: Eliminar valores atípicos
variable_original_sin_outliers <- variable_original[!variable_original %in% outliers_original]

# Opción 2: Reemplazar valores atípicos con la mediana
variable_original_corr <- ifelse(variable_original %in% outliers_original, median(variable_original, na
```

b) Manejo de Ceros Anómalos

```
# Ejemplo de eliminación de ceros anómalos
variable_original_sin_ceros <- variable_original[variable_original != 0]

# Ejemplo de imputación de ceros con la mediana
variable_original_corr <- ifelse(variable_original == 0, median(variable_original, na.rm = TRUE), varial
```

Utiliza la transformación de Yeo Johnson y encuentra el valor de lambda que maximiza el valor p de la prueba de normalidad que hayas utilizado (Anderson-Darling o Jarque Bera).

```
# Asegurarse de que el paquete 'car' está cargado
library(car)

# Aplicar la transformación de Yeo-Johnson utilizando powerTransform
yeo_johnson_model <- powerTransform(variable_original, family = "yjPower")

# Extraer el valor óptimo de lambda
lambda_yj <- yeo_johnson_model$lambda

# Aplicar la transformación Yeo-Johnson a los datos
variable_yj <- yjPower(variable_original, lambda_yj)

# Mostrar el valor de lambda encontrado
print(paste("Valor óptimo de lambda para Yeo-Johnson:", lambda_yj))
```

```
## [1] "Valor óptimo de lambda para Yeo-Johnson: 0.248690064213632"
```

Escribe la ecuación del modelo encontrado.

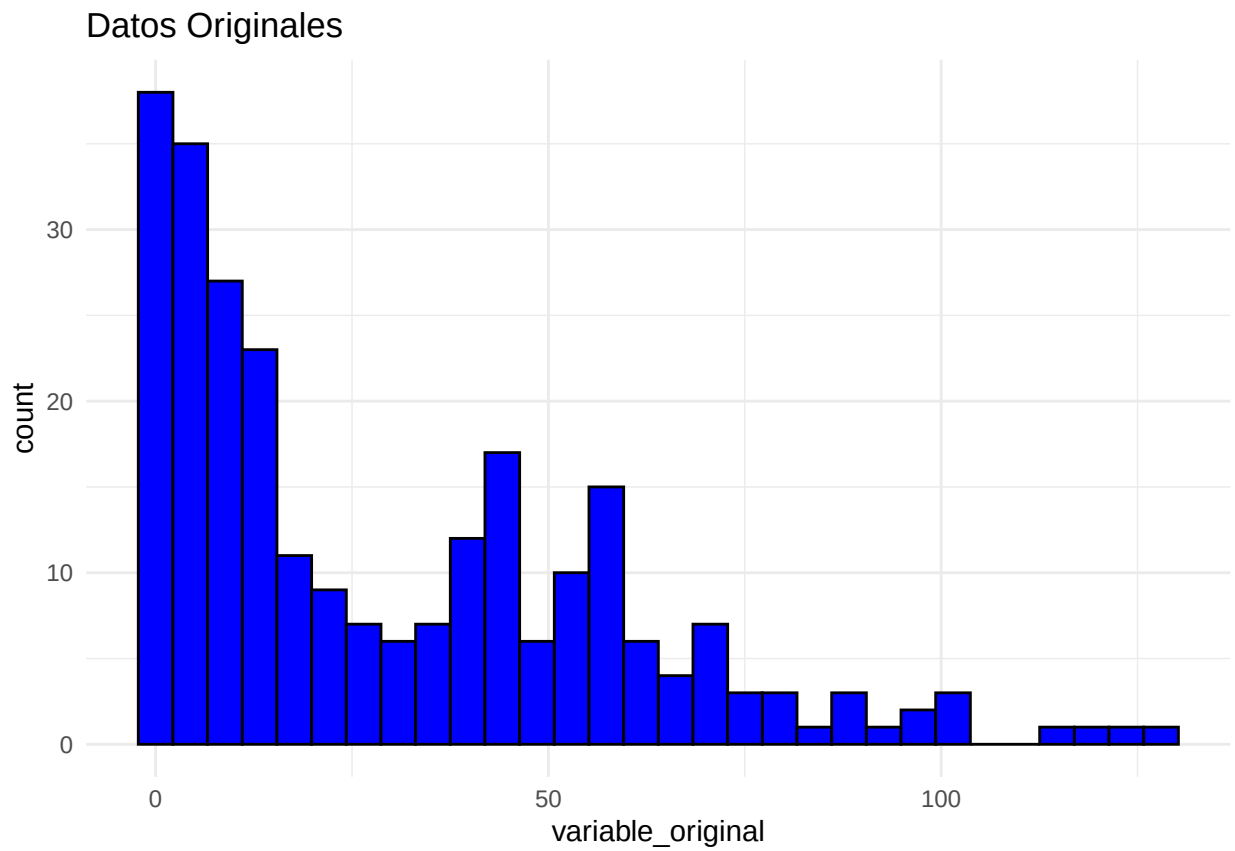
Analiza la normalidad de las transformaciones obtenidas con los datos originales. Utiliza como argumento de normalidad, para ellos vamos a utilizar la funcion que se creo anteriormente.

```
library(moments)
# Calcular estadísticas descriptivas
stats_original <- estadisticas_descriptivas(variable_original)
stats_yj <- estadisticas_descriptivas(variable_yj)

# Mostrar las estadísticas comparativas
stats_comparacion_yj <- data.frame(
  Original = stats_original,
  YeoJohnson = stats_yj
)
print(stats_comparacion_yj)
```

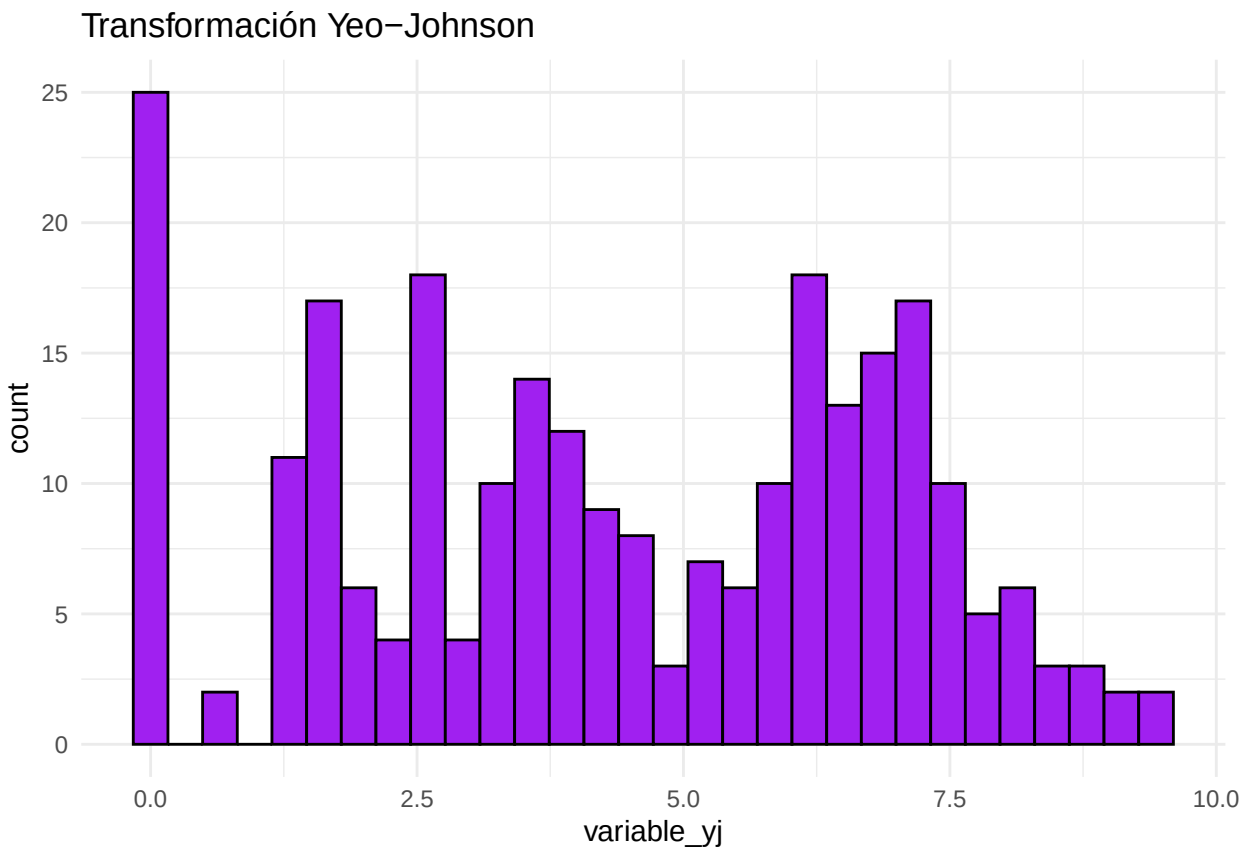
```
##            Original   YeoJohnson
## Min        0.010000  0.009962652
## Max      128.000000  9.444485567
## Media     29.424038  4.392636012
## Mediana   17.500000  4.285944540
## Q1.25%     5.750000  2.441515416
## Q3.75%    48.000000  6.563581490
## Sesgo      1.026089 -0.128676459
## Curtosis   3.487942  1.906949435
```

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

```
library(ggplot2)
# Histograma para los datos originales
ggplot(data.frame(variable_original), aes(x = variable_original)) +
  geom_histogram(bins = 30, color = "black", fill = "blue") +
  ggtitle("Datos Originales") + theme_minimal()
```




```r
# Histograma para la transformación Yeo-Johnson
ggplot(data.frame(variable_yj), aes(x = variable_yj)) +
  geom_histogram(bins = 30, color = "black", fill = "purple") +
  ggtitle("Transformación Yeo-Johnson") + theme_minimal()
```



Realiza la prueba de normalidad de Anderson-Darling para los datos transformados y los originales

```
library(nortest)
# Prueba de normalidad de Anderson-Darling para los datos originales
ad_original <- ad.test(variable_original)
print(ad_original$p.value)
```

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

```
# Prueba de normalidad de Anderson-Darling para la transformación Yeo-Johnson
ad_yj <- ad.test(variable_yj)
print(ad_yj$p.value)
```

```
## [1] 1.498432e-08
```

La transformación de Yeo-Johnson es la mejor opción para normalizar los datos, ya que logra un sesgo casi nulo y una curtosis más baja, lo que indica una distribución más simétrica y con colas menos pesadas. Aunque la transformación Box-Cox Exacta también es efectiva, reduciendo tanto el sesgo como la curtosis a valores cercanos a los ideales para una distribución normal, la ligera negatividad del sesgo y la mayor curtosis sugieren que Yeo-Johnson ofrece una distribución más adecuada para análisis que requieren mayor simetría y menor "puntiagudez". Por lo tanto, Yeo-Johnson se destaca por proporcionar un equilibrio óptimo entre simetría y control de colas en la distribución.

Los modelos de transformación de datos como Box-Cox y Yeo-Johnson son útiles para ajustar la distribución de los datos, mejorando su normalidad y estabilizando la varianza, lo que es especialmente importante en

análisis estadísticos y modelos de regresión. Box-Cox es ideal para datos positivos, mientras que Yeo-Johnson puede manejar tanto datos positivos como negativos, lo que lo hace más versátil. Sin embargo, ambas transformaciones pueden complicar la interpretación de los resultados y requieren un ajuste cuidadoso de sus parámetros para evitar problemas como el sobreajuste.

Por otro lado, la transformación y el escalamiento de datos tienen objetivos distintos. La transformación modifica la forma de la distribución de los datos, haciéndolos más normales o lineales, lo cual es crucial cuando se observan problemas de sesgo o no normalidad en los datos. En cambio, el escalamiento ajusta el rango de los valores de los datos sin cambiar su distribución, y es necesario en algoritmos de aprendizaje automático que son sensibles a las magnitudes de los datos, como SVM o redes neuronales. Mientras que la transformación se utiliza para mejorar la forma y la varianza de los datos, el escalamiento es esencial para garantizar que diferentes características en los modelos de machine learning estén en el mismo rango, facilitando una mejor convergencia y resultados precisos.