Actividad integradora 1

Jacobo Hirsch Rodriguez

2024-08-20

##Punto 1. Análisis descriptivo de la variabl

Analiza la siguiente variable: Grasas monosaturadas

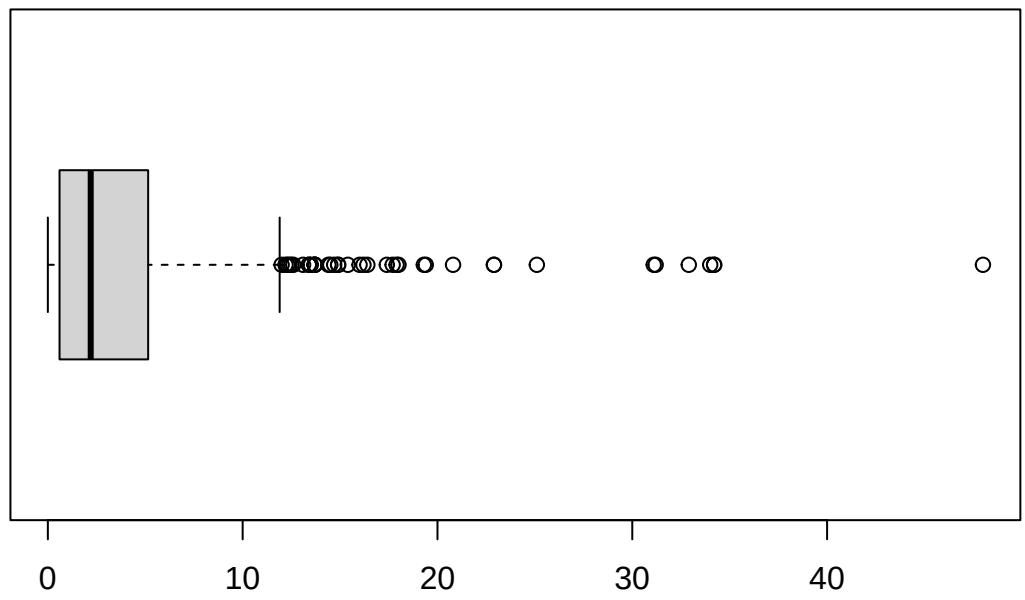
```
F=read.csv("./food_data_g.csv") #leer la base de datos

cabeceras <- colnames(F)
print(cabeceras)
```

```
##  [1] "X"                    "Unnamed..0"           "food"
##  [4] "Caloric.Value"        "Fat"                  "Saturated.Fats"
##  [7] "Monounsaturated.Fats" "Polyunsaturated.Fats" "Carbohydrates"
## [10] "Sugars"               "Protein"              "Dietary.Fiber"
## [13] "Cholesterol"          "Sodium"               "Water"
## [16] "Vitamin.A"            "Vitamin.B1"           "Vitamin.B11"
## [19] "Vitamin.B12"          "Vitamin.B2"           "Vitamin.B3"
## [22] "Vitamin.B5"           "Vitamin.B6"           "Vitamin.C"
## [25] "Vitamin.D"            "Vitamin.E"            "Vitamin.K"
## [28] "Calcium"              "Copper"               "Iron"
## [31] "Magnesium"            "Manganese"            "Phosphorus"
## [34] "Potassium"            "Selenium"             "Zinc"
## [37] "Nutrition.Density"
```

1. Para analizar datos atípicos se te sugiere:

Graficar el diagrama de caja y bigote

```
boxplot(F$Monounsaturated.Fats, main="Diagrama de caja para grasas monosaturadas", horizontal=TRUE)
```



Calcula las principales medidas que te ayuden a identificar datos atípicos (utilizar summary te puede abreviar el cálculo): Cuartil 1, Cuartil 2, Media, Cuartil 3, Rango intercuartílico y Desviación estándar

```
summary(F$Monounsaturated.Fats)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.600   2.200   4.002   5.150  48.000
```

```
#calculo de rango intercuartilico y cuartiles

q1 <- quantile(F$Monounsaturated.Fats, 0.25)

q3 <- quantile(F$Monounsaturated.Fats, 0.75)

ri <- q3 - q1

cat("rango intercuartilico:", ri)
```

```
## rango intercuartilico: 4.55
```

```
desviacion_estandar <- sd(F$Monounsaturated.Fats)

cat("desviacion estandar :", desviacion_estandar)
```

```
## desviacion estandar : 5.540608
```

Identifica la cota de 1.5 rangos intercuartílicos para datos atípicos, ¿hay datos atípicos de acuerdo con este criterio? ¿cuántos son?

```
lim_inf <- q1 - 1.5 * ri
lim_sup <- q3 + 1.5 * ri

outliers_ri <- F$Monounsaturated.Fats[F$Monounsaturated.Fats < lim_inf | F$Monounsaturated.Fats > lim_su


cat("cantidad de elementos en atipicos en 1.5 rangos intercuartilicos:", length(outliers_ri))
```

```
## cantidad de elementos en atipicos en 1.5 rangos intercuartilicos: 40
```

Identifica la cota de 3 desviaciones estándar alrededor de la media, ¿hay datos atípicos de acuerdo con este criterio? ¿cuántos son?

```
mean_monosaturated <- mean(F$Monounsaturated.Fats)
sd_monosaturated <- sd(F$Monounsaturated.Fats)

lim_inf_sd <- mean_monosaturated - 3 * sd_monosaturated
lim_sup_sd <- mean_monosaturated + 3 * sd_monosaturated

outliers_sd <- F$Monounsaturated.Fats[F$Monounsaturated.Fats < lim_inf_sd | F$Monounsaturated.Fats > lim



cat("cantidad de elementos 3 desviaciones estándar alrededor de la media:", length(outliers_sd))
```

```
## cantidad de elementos 3 desviaciones estándar alrededor de la media: 11
```

Identifica la cota de 3 rangos intercuartílicos para datos extremos, ¿hay datos extremos de acuerdo con este criterio? ¿cuántos son?

```
lim_inf_ext <- q1 - 3 * ri
lim_sup_ext <- q3 + 3 * ri

extremes_ri <- F$Monounsaturated.Fats[F$Monounsaturated.Fats < lim_inf_ext | F$Monounsaturated.Fats > li


cat("cantidad de datos  extremos: ", length(extremes_ri))
```

```
## cantidad de datos  extremos:  13
```

2. Para analizar normalidad se te sugiere:

Realiza pruebas de normalidad univariada para la variable (utiliza las pruebas de Anderson-Darling y de Jarque Bera). No olvides incluir H0 y H1 para la prueba de normalidad.

prueba de Anderson-Darling

```r
library(nortest)

# Prueba de Anderson-Darling para la variable Calorias
resultado_ad <- ad.test(F$Monounsaturated.Fats)

# Mostrar los resultados
print(resultado_ad)
```

```
##
##  Anderson-Darling normality test
##
## data:  F$Monounsaturated.Fats
## A = 46.499, p-value < 2.2e-16
```

pruebas Jarque Bera

```r
library(tseries)
```

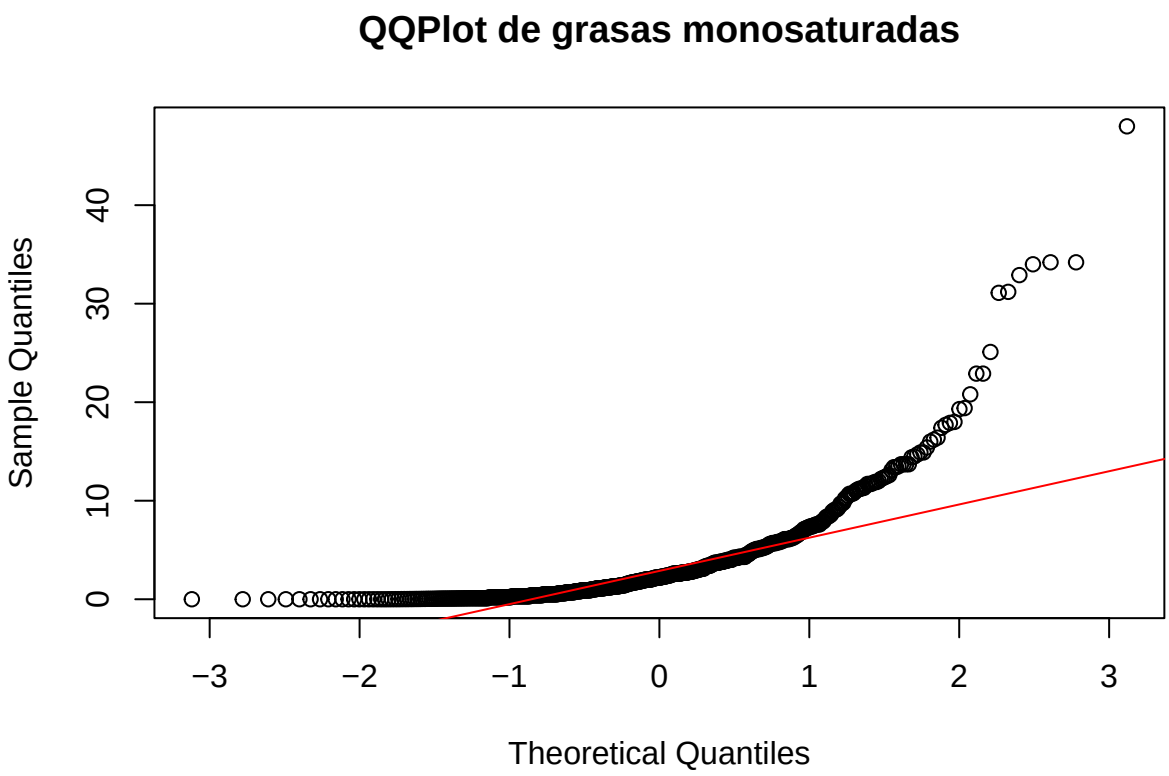
```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
#asegurarnos de que no tenga valores NA
if (any(is.na(F$Monounsaturated.Fats))) {
    # Elimina los NA
    variable_jb <- na.omit(F$Monounsaturated.Fats)
} else {
    variable_jb <- F$Monounsaturated.Fats
}

# Realizar la prueba de Jarque-Bera
jb_test <- jarque.bera.test(variable_jb)
print(jb_test)
```

```
##
##  Jarque Bera Test
##
## data:  variable_jb
## X-squared = 5884, df = 2, p-value < 2.2e-16
```

Grafica los datos y su respectivo QQPlot: qqnorm(datos) y qqline(datos)

```r
qqnorm(F$Monounsaturated.Fats, main="QQPlot de grasas monosaturadas")
qqline(F$Monounsaturated.Fats, col="red")
```



Calcula el coeficiente de sesgo y el coeficiente de curtosis

```
library(moments)

# Calcular sesgo y curtosis para Sodium
sesgo_monosaturated <- skewness(F$Monounsaturated.Fats)
curtosis_monosaturated <- kurtosis(F$Monounsaturated.Fats)

# Mostrar resultados
print(paste("Sesgo:", sesgo_monosaturated))
```

```
## [1] "Sesgo: 3.20798187079513"
```

```
print(paste("Curtosis:", curtosis_monosaturated))
```

```
## [1] "Curtosis: 17.6671218836864"
```

Compara las medidas de media, mediana y rango medio

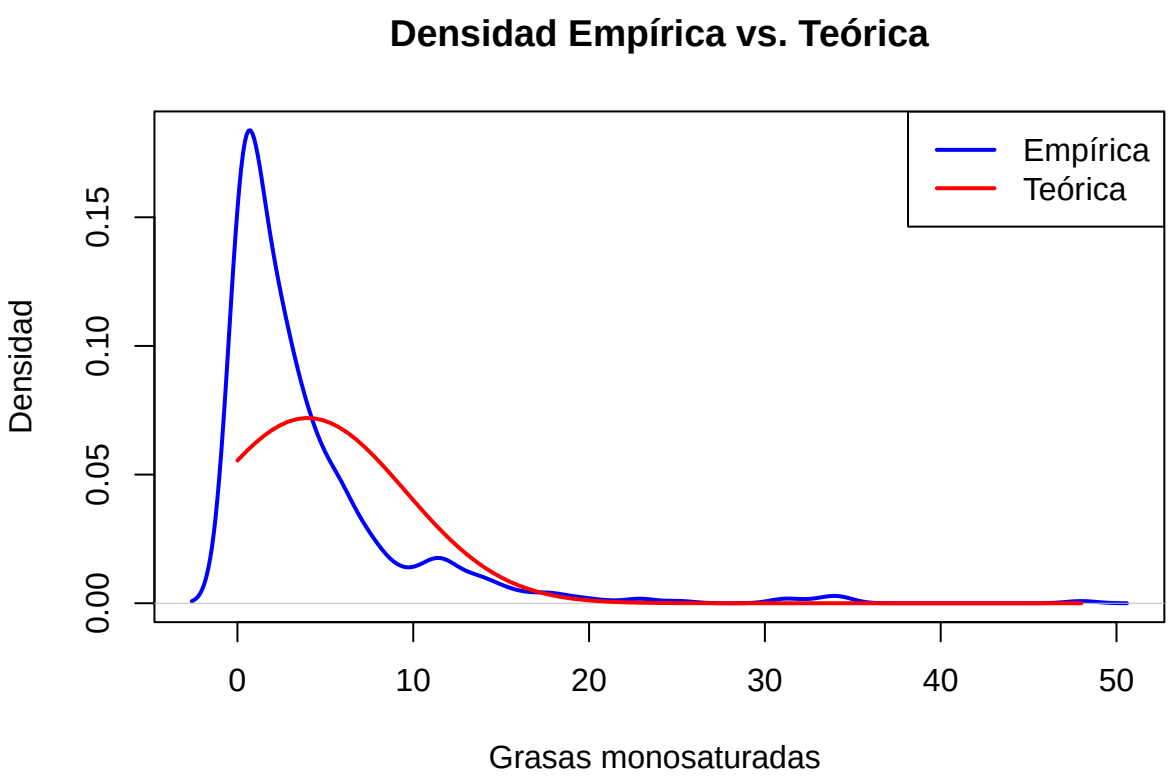
```
summary(F$Monounsaturated.Fats)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.600   2.200   4.002   5.150  48.000
```

Realiza el gráfico de densidad empírica y teórica suponiendo normalidad en la variable. Adapta el código:

hist(datos,freq=FALSE) lines(density(datos),col="red") curve(dnorm(x,mean=mean(datos,sd=sd(datos)), from=-6, to=6, add=TRUE, col="blue",lwd=2)

```r
# Supongamos que ya tienes el dataset F cargado y la variable Monounsaturated.Fats
v_one <- F$Monounsaturated.Fats

# Asegurarse de eliminar valores NA
v_one <- na.omit(v_one)

# Calcular la densidad empírica
densidad_empirica <- density(v_one)

# Calcular la media y la desviación estándar de la variable
media <- mean(v_one)
desviacion_estandar <- sd(v_one)

# Crear una secuencia de valores sobre los cuales se calculará la densidad teórica
x <- seq(min(v_one), max(v_one), length = 100)

# Calcular la densidad teórica bajo el supuesto de normalidad
densidad_teorica <- dnorm(x, mean = media, sd = desviacion_estandar)

# Graficar la densidad empírica
plot(densidad_empirica, main = "Densidad Empírica vs. Teórica", xlab = "Grasas monosaturadas ", ylab =

# Agregar la densidad teórica al gráfico
lines(x, densidad_teorica, col = "red", lwd = 2)

# Agregar una leyenda
legend("topright", legend = c("Empírica", "Teórica"), col = c("blue", "red"), lwd = 2)
```



Interpreta los gráficos y los resultados obtenidos en cada punto con vías a indicar si hay normalidad de los datos Comenta las características encontradas: Considera alejamientos de normalidad por simetría, curtosis Comenta si hay aparente influencia de los datos atípicos en la normalidad de los datos Emite una conclusión sobre la normalidad de los datos. Se debe argumentar en términos de los 3 puntos analizados: las pruebas de normalidad, los gráficos y las medidas.

##Punto 2. Transformación a normalidad

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

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

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

Encuentra la mejor transformación de los datos para lograr normalidad. Puedes hacer uso de la transformación Box-Cox o de Yeo Johnson o el comando powerTransform para encontrar la mejor lambda para la transformación. Utiliza el modelo exacto y el aproximado de acuerdo con las sugerencias de Box y Cox para la transformación.

vamos a utiliza la transformación box-cox, pero primero necesitamos imputar los datos ya que la transoformación de box-cox no admite que los datos sean iguales a cero

```r
# Reemplazar ceros por un pequeño valor positivo
F$Monounsaturated.Fats[F$Monounsaturated.Fats == 0] <- 0.01
```

Utiliza la transformación Box-Cox

```r
  # Aplicar la transformación de Box-Cox
  boxcox_model <- boxcox(F$Monounsaturated.Fats ~ 1, plotit = FALSE)

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

  # Transformación exacta
  variable_bc_exact <- (F$Monounsaturated.Fats^lambda_opt - 1) / lambda_opt

  # Transformación aproximada (logarítmica)
  variable_bc_approx <- log(F$Monounsaturated.Fats)

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

```
## [1] "Valor óptimo de lambda: 0.2"
```

Escribe las ecuaciones de los modelos de transformación encontrados.

```r
library(latex2exp)
# Crear la expresión LaTeX
expr <- TeX(r'(Y(0.2) = \frac{X^{0.2} - 1}{0.2})')
```

Analiza la normalidad de las transformaciones obtenidas con los datos originales. Utiliza como argumento de normalidad: Compara las medidas: Mínimo, máximo, media, mediana, cuartil 1 y cuartil 3, sesgo y curtosis.

```r
# 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(F$Monounsaturated.Fats)
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,
```

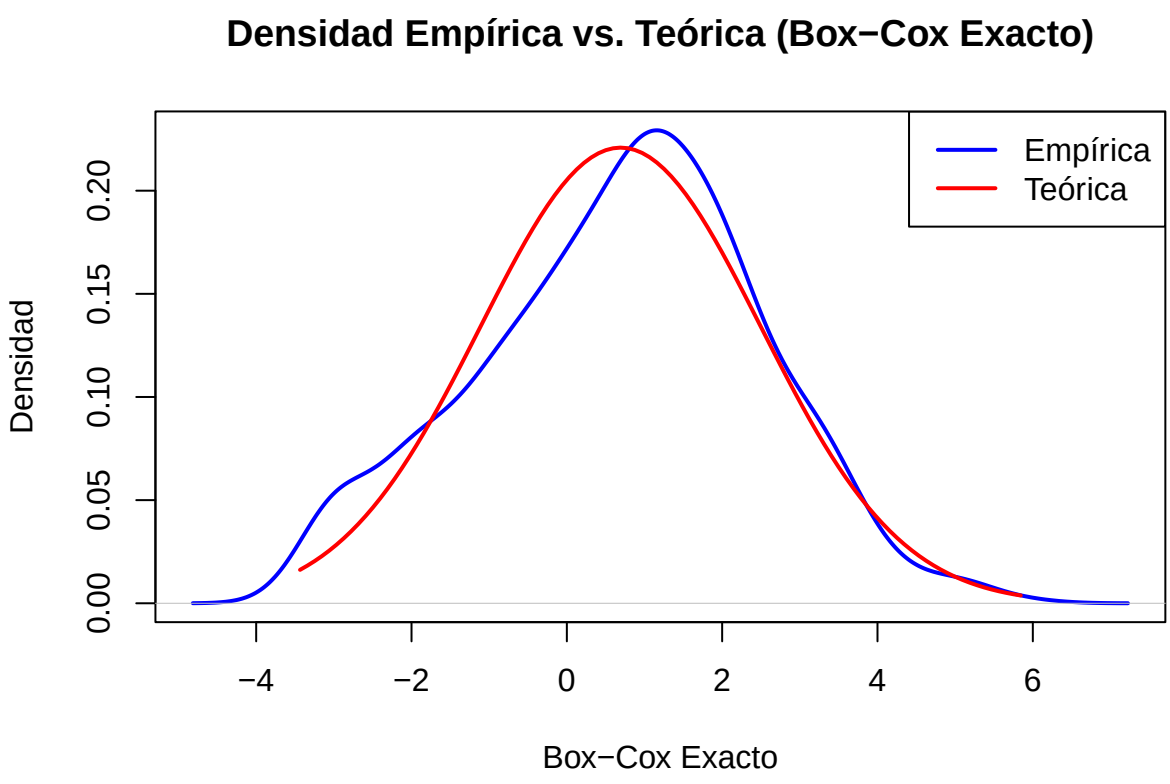
```
                                BoxCox_Exacto = stats_bc_exact,
                                BoxCox_Aproximado = stats_bc_approx)

print(stats_comparacion)
```

```
##           Original BoxCox_Exacto BoxCox_Aproximado
## Min       0.003000    -3.4354327        -5.8091430
## Max      48.000000     5.8447177         3.8712010
## Media     4.001909     0.6926495         0.3431234
## Mediana   2.200000     0.8540246         0.7884574
## Q1.25%    0.600000    -0.4855977        -0.5108256
## Q3.75%    5.150000     1.9394999         1.6389496
## Sesgo     3.208325    -0.1445613        -1.0286169
## Curtosis 17.669407     2.6581955         3.7400111
```

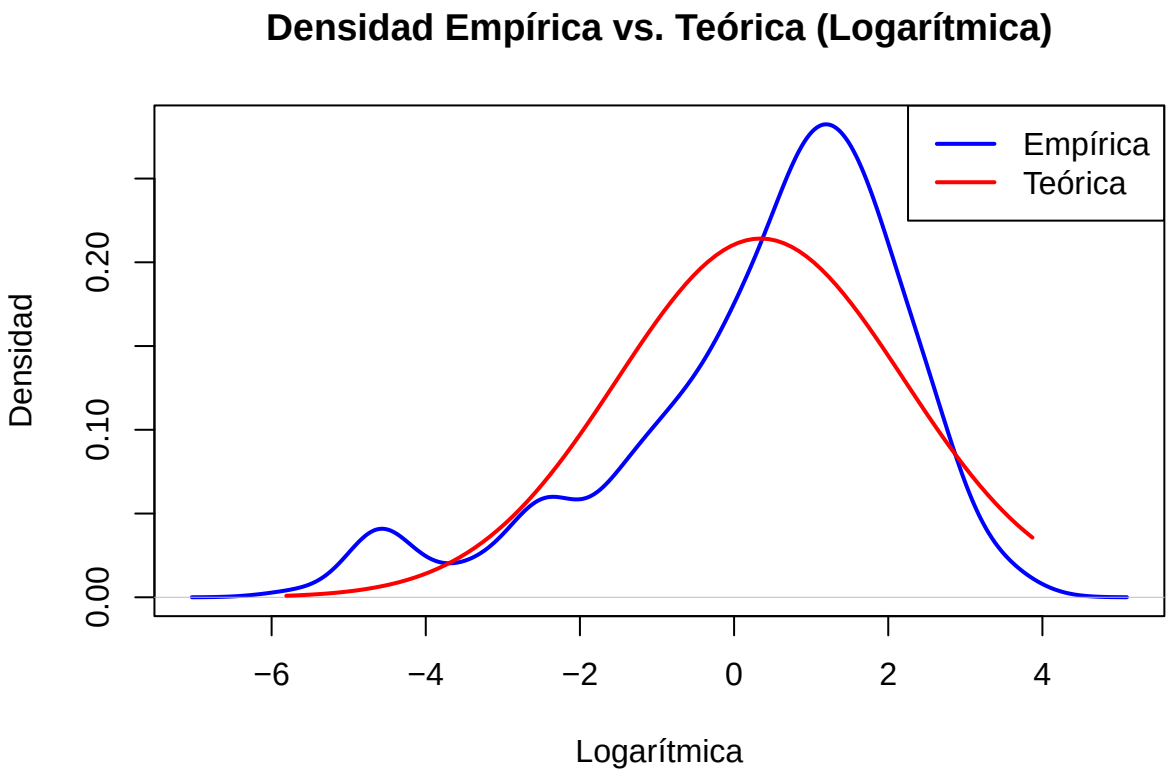
Grafica las funciones de densidad empírica y teórica de los 2 modelos obtenidos (exacto y aproximado) y los datos originales.

Modelo exacto

```
# Calcular la densidad empírica para la transformación exacta
densidad_empirica_exact <- density(variable_bc_exact)

# Calcular la media y la desviación estándar de la transformación exacta
media_exact <- mean(variable_bc_exact)
desviacion_estandar_exact <- sd(variable_bc_exact)

# Crear una secuencia de valores para la densidad teórica de la transformación exacta
x_exact <- seq(min(variable_bc_exact), max(variable_bc_exact), length = 100)

# Calcular la densidad teórica bajo el supuesto de normalidad para la transformación exacta
densidad_teorica_exact <- dnorm(x_exact, mean = media_exact, sd = desviacion_estandar_exact)

# Calcular la densidad empírica para la transformación aproximada
densidad_empirica_approx <- density(variable_bc_approx)

# Calcular la media y la desviación estándar de la transformación aproximada
media_approx <- mean(variable_bc_approx)
desviacion_estandar_approx <- sd(variable_bc_approx)

# Crear una secuencia de valores para la densidad teórica de la transformación aproximada
x_approx <- seq(min(variable_bc_approx), max(variable_bc_approx), length = 100)

# Calcular la densidad teórica bajo el supuesto de normalidad para la transformación aproximada
densidad_teorica_approx <- dnorm(x_approx, mean = media_approx, sd = desviacion_estandar_approx)

# Graficar la densidad empírica y teórica para la transformación exacta
plot(densidad_empirica_exact, main = "Densidad Empírica vs. Teórica (Box-Cox Exacto)", xlab = "Box-Cox
lines(x_exact, densidad_teorica_exact, col = "red", lwd = 2)
legend("topright", legend = c("Empírica", "Teórica"), col = c("blue", "red"), lwd = 2)
```




```r
# Graficar la densidad empírica y teórica para la transformación aproximada
plot(densidad_empirica_approx, main = "Densidad Empírica vs. Teórica (Logarítmica)", xlab = "Logarítmica
lines(x_approx, densidad_teorica_approx, col = "red", lwd = 2)
legend("topright", legend = c("Empírica", "Teórica"), col = c("blue", "red"), lwd = 2)
```



Realiza la prueba de normalidad de Anderson-Darling y de Jarque Bera para los datos transformados y los originales Detecta anomalías y corrige tu base de datos (datos atípicos, ceros anámalos, etc).

Primero haremos la pruebas de Anderson-Darling

```
# Prueba de normalidad de Anderson-Darling para los datos originales
ad_original <- ad.test(F$Monounsaturated.Fats)
print(ad_original$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] 0.0004435765
```

```
# 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
```

Ahora haremos la prueba de Jarque Bera

```r
#asegurarnos de que no tenga valores NA
if (any(is.na(F$Monounsaturated.Fats))) {
    # Elimina los NA
    variable_jb_two <- na.omit(F$Monounsaturated.Fats)
} else {
    variable_jb_two <- F$Monounsaturated.Fats
}

# Realizar la prueba de normalidad  de Jarque-Bera para datos normales
jb_test_two <- jarque.bera.test(variable_jb_two)
print(jb_test_two)
```

```
##
##  Jarque Bera Test
##
## data:  variable_jb_two
## X-squared = 5885.7, df = 2, p-value < 2.2e-16
```

```r
# Prueba de normalidad de Jarque-Bera para la transformación exacta de Box-Cox
jb_exact <- ad.test(variable_bc_exact)
print(jb_exact)
```

```
##
##  Anderson-Darling normality test
##
## data:  variable_bc_exact
## A = 1.585, p-value = 0.0004436
```

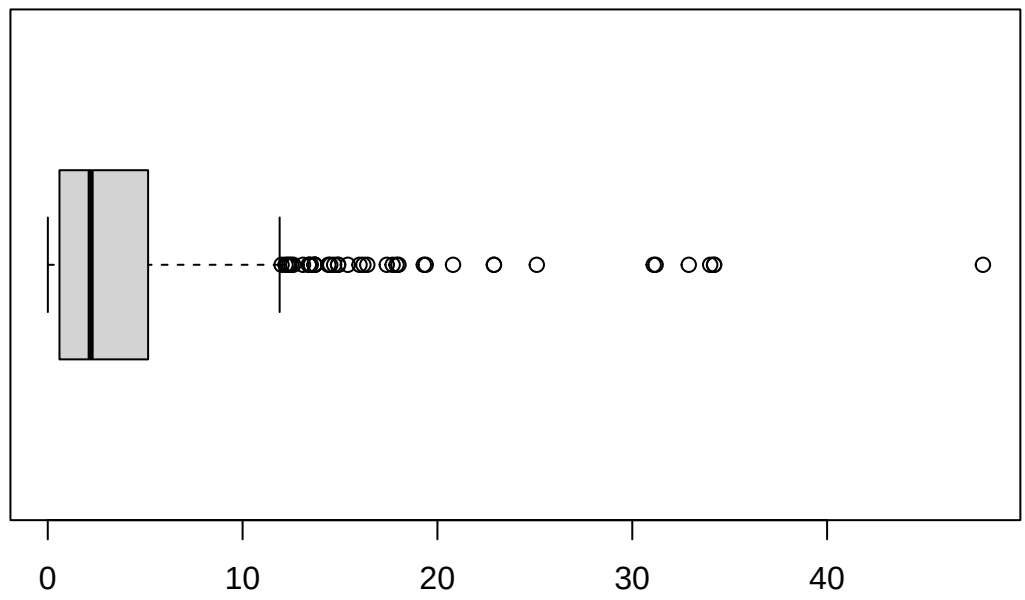
```r
# Prueba de normalidad de Jarque-Bera para la transformación aproximada de Box-Cox
jb_aprox <- ad.test(variable_bc_approx)
print(jb_aprox)
```

```
##
##  Anderson-Darling normality test
##
## data:  variable_bc_approx
## A = 12.888, p-value < 2.2e-16
```

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

a) identificación de valores atipicos

```r
# Boxplot para detectar valores atípicos
boxplot(F$Monounsaturated.Fats, main="Boxplot de los datos originales", horizontal = TRUE)
```




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

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

```r
print(outliers_original)
```

```
##  [1] 12.3 17.9 12.5 19.4 12.4 13.1 16.4 12.0 13.5 13.7 14.4 13.4 13.7 32.9 20.8
## [16] 16.2 12.2 15.4 18.0 13.7 12.6 19.3 13.7 22.9 25.1 14.5 34.0 16.0 13.4 22.9
## [31] 14.9 17.4 31.2 34.2 14.9 17.7 34.2 14.7 31.1 48.0
```

```r
# Z-Score para detectar valores atípicos
z_scores_original <- scale(F$Monounsaturated.Fats)
outliers_z_original <- F$Monounsaturated.Fats[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] 32.9 20.8 22.9 25.1 34.0 22.9 31.2 34.2 34.2 31.1 48.0
```

ahora que visualizamos los valores atipicos y utilizamos el z score, podemos eliminarlos de la base

```
# Crear un nuevo dataset sin los valores atípicos
F_new <- F[!F$Monounsaturated.Fats %in% outliers_original, ]

# Verificar el tamaño del dataset original y el nuevo
print(dim(F))
```

```
## [1] 551  37
```

```
print(dim(F_new))
```

```
## [1] 511  37
```

b) ahora vamos a detectar los ceros anómalos

```
# Contar cuántos ceros existen en la variable
ceros_anomalos <- sum(F_new$Monounsaturated.Fats == 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 <- F_new$Monounsaturated.Fats[F$Monounsaturated.Fats == 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)
```

como podemos ver, no tenemos ceros anomalos ya que al principio los eliminamos debido a que la transformación de boxcox no permite que los valores sean iguales a 0

vamos a añadir el sufijo new a las variables para indicar que los datos fueron limpiados para valores atipicos y verificado que no existían ceros anomalos

tambien necesitamos volver a aplicar las transformaciones para ver como se comportan:

```
  # Aplicar la transformación de Box-Cox
  boxcox_model_new <- boxcox(F_new$Monounsaturated.Fats ~ 1, plotit = FALSE)

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

  # Transformación exacta
  variable_bc_exact_new <- (F_new$Monounsaturated.Fats^lambda_opt_new - 1) / lambda_opt_new

  # Transformación aproximada (logarítmica)
  variable_bc_approx_new <- log(F_new$Monounsaturated.Fats)

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

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

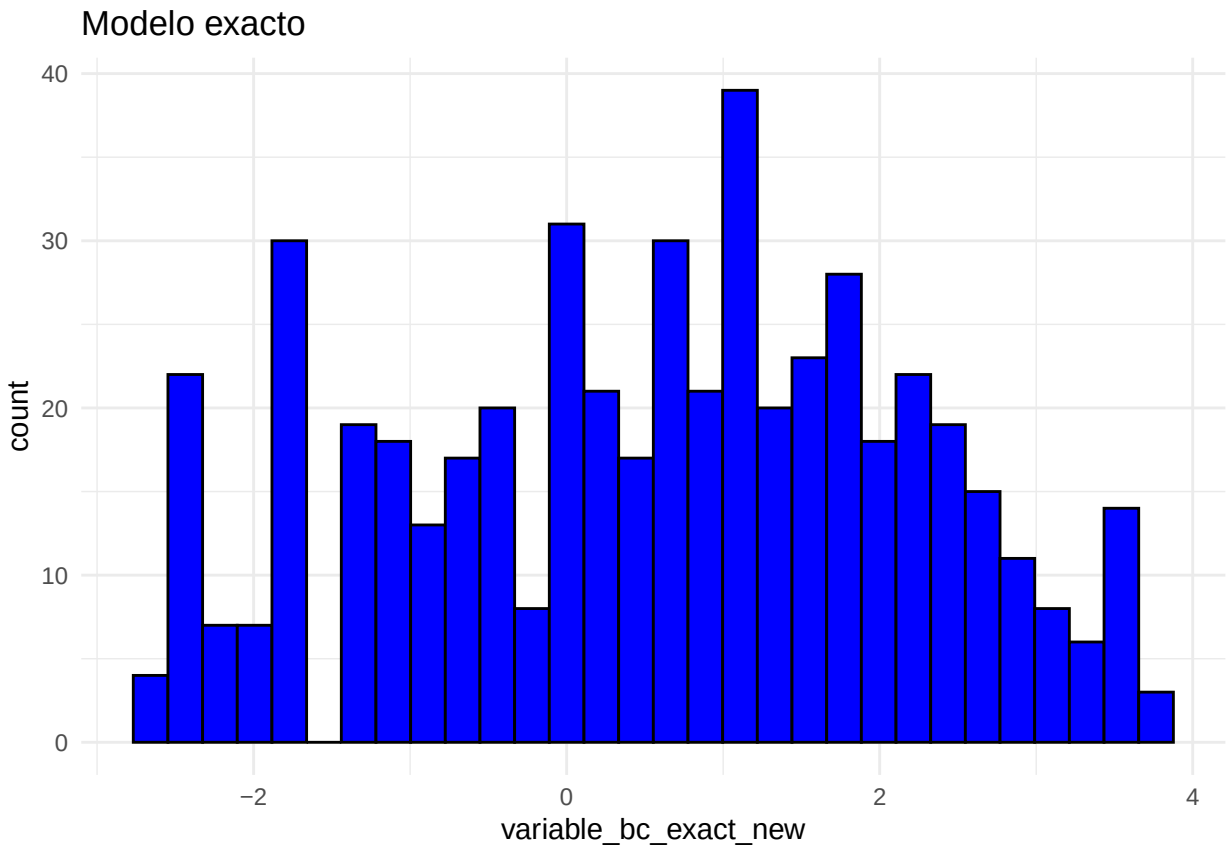
Compara las medidas: Mínimo, máximo, media, mediana, cuartil 1 y cuartil 3, sesgo y curtosis.

```
# Calcular estadísticas descriptivas
stats_original_new <- estadisticas_descriptivas(F_new$Monounsaturated.Fats)
stats_bc_exact_new <- estadisticas_descriptivas(variable_bc_exact_new)
stats_bc_approx_new <- estadisticas_descriptivas(variable_bc_approx_new)

# Mostrar los resultados
stats_comparacion_new <- data.frame(Original_new = stats_original_new,
                                    BoxCox_Exacto_new = stats_bc_exact_new,
                                    BoxCox_Aproximado_new = stats_bc_approx_new)

print(stats_comparacion_new)
```

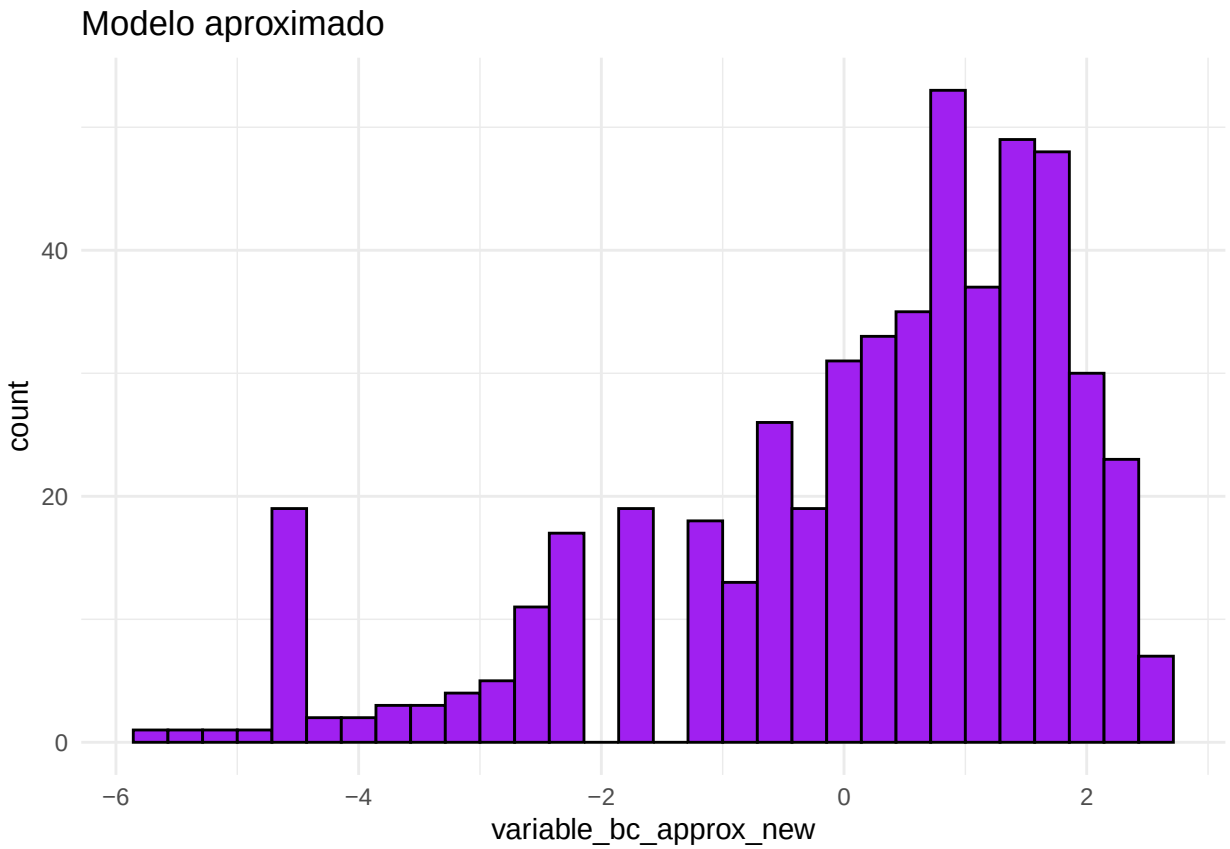
```
##          Original_new BoxCox_Exacto_new BoxCox_Aproximado_new
## Min          0.003000        -2.7498679            -5.8091430
## Max         11.900000         3.6738396             2.4765384
## Media        2.814975         0.5791339             0.1445157
## Mediana      2.000000         0.7704814             0.6931472
## Q1.25%       0.500000        -0.6258253            -0.6931472
## Q3.75%       4.200000         1.7935512             1.4350845
## Sesgo        1.285839        -0.1649129            -1.1643706
## Curtosis     4.070500         2.1933006             3.8239187
```

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

```
# Histograma para el modelo exacto
ggplot(data.frame(variable_bc_exact_new), aes(x = variable_bc_exact_new)) +
  geom_histogram(bins = 30, color = "black", fill = "blue") +
  ggtitle("Modelo exacto") + theme_minimal()
```



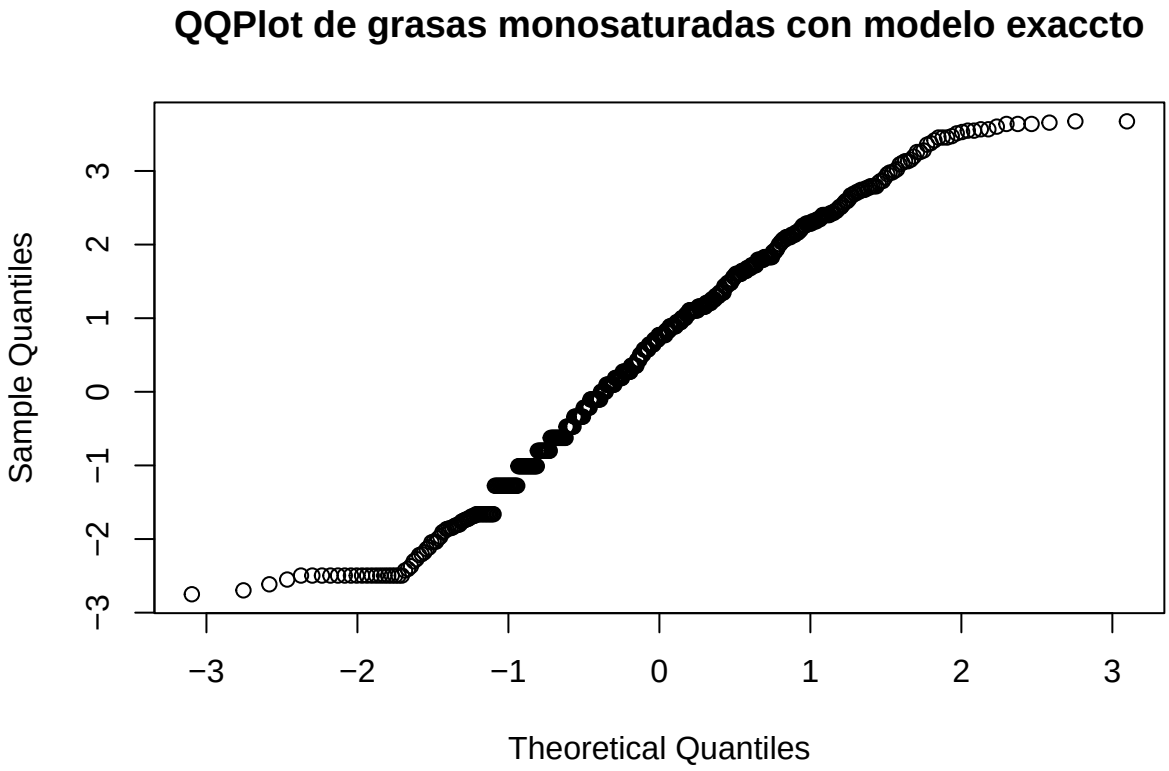

```
# Histograma para el modelo aproximado
ggplot(data.frame(variable_bc_approx_new), aes(x = variable_bc_approx_new)) +
  geom_histogram(bins = 30, color = "black", fill = "purple") +
  ggtitle("Modelo aproximado") + theme_minimal()
```



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

Define la mejor transformación de los datos de acuerdo a las características de los modelos que encontraste. Toma en cuenta los criterios del inciso anterior para analizar normalidad y la economía del modelo.

```
qqnorm(variable_bc_exact_new, main="QQPlot de grasas monosaturadas con modelo exaccto")
```



Conclusiones: Antes de la imputacion de los datos e incluso despues, podemos observar que la mejor transformación es la que te da el modelo exacto ya que logra un sesgo casi nulo y una curtosis más baja, lo que indica una distribución más simétrica y con colas menos pesadas.