Actividad Integradora

Adrian Pineda Sanchez

2024-08-20

Trabaja con el set de datos Nutrición Mundial Download Nutrición Mundial, que contiene diversas características del alimentos que se consumen en el mundo. Pueden encontrar más información sobre ella en: Utsav Dey. (2024). Food Nutrition Dataset [Data set]. Kaggle.

Punto 1. Análisis descriptivo de la variable

Analiza una de las siguientes variables en cuanto a sus datos atípicos y normalidad. La variable que te corresponde analizar te será asignada por tu profesora al inicio de la actividad:

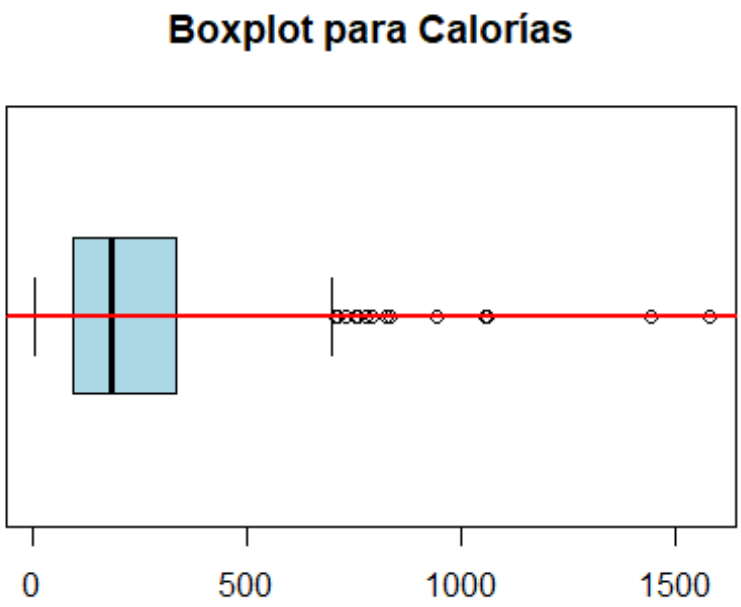
```
M=read.csv("food_data_g.csv") #Leer la base de datos
```

Como me toco la variable 1, ire con calorias el cual tiene como Columna: Caloric Value

```
# Contar el número de datos
Calorias = M$Caloric.Value

# Contar el número de datos
Conteo_datos_calorias <- length(Calorias)

print(Conteo_datos_calorias)

## [1] 551
```

1.1 Graficar el diagrama de caja y bigote

```
# Boxplot para Calorías
boxplot(Calorias, horizontal = TRUE, main = "Boxplot para Calorías", col =
"lightblue")
abline(h=1, col="red", lwd=2)  # Línea de referencia en el límite de los
datos atípicos
```



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

```
summary(Calorias)  # Esto te da los cuartiles y la media
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     3.0    94.5   186.0   237.4   337.0  1578.0
```

```
cat("\n")
```

```
cat("La desviacion estandar es:", sd(Calorias),"\n","\n")       # Desviación estándar
```

```
## La desviacion estandar es: 199.2356
##
```

```
cat("El rango intercuatilico",IQR(Calorias))      # Rango intercuartílico
```

```
## El rango intercuatilico 242.5
```

```
# Calcular cuartiles y rango intercuartílico para Calorías
q1_cal <- quantile(Calorias, 0.25)
q3_cal <- quantile(Calorias, 0.75)
iqr_cal <- IQR(Calorias)

cat("Cuartil 1 para Calorías:", q1_cal, "\n")
```

```
## Cuartil 1 para Calorías: 94.5

cat("Cuartil 3 para Calorías:", q3_cal, "\n")

## Cuartil 3 para Calorías: 337

cat("Rango intercuartílico para Calorías:", iqr_cal, "\n")

## Rango intercuartílico para Calorías: 242.5
```

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

```
Q1 <- quantile(Calorias, 0.25)
Q3 <- quantile(Calorias, 0.75)
IQR <- Q3 - Q1
lower_bound <- Q1 - 1.5 * IQR
upper_bound <- Q3 + 1.5 * IQR

outliers_1_5_iqr <- Calorias[Calorias < lower_bound | Calorias> upper_bound]
cat("Los outlieres en base a los 1.5 IQR son:", outliers_1_5_iqr,"\n","\n")

## Los outlieres en base a los 1.5 IQR son: 1058 708 779 1440 1578 776 942
1061 833 790 942 733 709 755 760 824 711 759 753
##

cat("La cantidad de outliers es:",length(outliers_1_5_iqr))  # Cantidad de
datos atípicos

## La cantidad de outliers es: 19

# Para Calorías
cota_inf_cal_1_5iqr <- q1_cal - 1.5 * iqr_cal
cota_sup_cal_1_5iqr <- q3_cal + 1.5 * iqr_cal

# Imprimir las cotas
cat("Cota inferior 1.5 IQR para Calorías:", cota_inf_cal_1_5iqr, "\n")

## Cota inferior 1.5 IQR para Calorías: -269.25

cat("Cota superior 1.5 IQR para Calorías:", cota_sup_cal_1_5iqr, "\n")

## Cota superior 1.5 IQR para Calorías: 700.75

# Identificar los datos atípicos
outliers_inf_1_5iqr_cal <- Calorias[Calorias < cota_inf_cal_1_5iqr]
outliers_sup_1_5iqr_cal <- Calorias[Calorias > cota_sup_cal_1_5iqr]

# Imprimir los datos atípicos
cat("\n")

cat("Datos atípicos inferiores según 1.5 IQR para Calorías:",
outliers_inf_1_5iqr_cal, "\n")
```

```
## Datos atípicos inferiores según 1.5 IQR para Calorías:

cat("Datos atípicos superiores según 1.5 IQR para Calorías:",
outliers_sup_1_5iqr_cal, "\n")

## Datos atípicos superiores según 1.5 IQR para Calorías: 1058 708 779 1440
1578 776 942 1061 833 790 942 733 709 755 760 824 711 759 753
```

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

```
# Para Calorías
cota_sup_cal_3sd <- mean(Calorias) + 3 * sd(Calorias)

# Imprimir la cota
cat("Cota superior 3 desviaciones estándar para Calorías:", cota_sup_cal_3sd,
"\n")

## Cota superior 3 desviaciones estándar para Calorías: 835.0661

# Identificar los datos atípicos
outliers_3sd_cal <- Calorias[Calorias > cota_sup_cal_3sd]

# Imprimir los datos atípicos
cat("Datos atípicos según 3 SD para Calorías:", outliers_3sd_cal, "\n")

## Datos atípicos según 3 SD para Calorías: 1058 1440 1578 942 1061 942

cat("La cantidad de outliers es:",length(outliers_3sd_cal))  # Cantidad de
datos atípicos

## La cantidad de outliers es: 6
```

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

```
lower_bound_extreme <- Q1 - 3 * IQR
upper_bound_extreme <- Q3 + 3 * IQR

extreme_outliers <- Calorias[ Calorias < lower_bound_extreme |  Calorias >
upper_bound_extreme]
extreme_outliers

## [1] 1440 1578

length(extreme_outliers)  # Cantidad de datos extremos

## [1] 2
```

Podemos analizar si hay valores con Calorias 0 que afecten

```r
# Filtrar y contar los valores de Calorías iguales a 0
calorias_cero <- Calorias[Calorias == 0]
length(calorias_cero)
```

```
## [1] 0
```

NO hay

1.6 Interpreta los resultados obtenidos y argumenta sobre el comportamiento de los datos atípicos y extremos en la variable seleccionada

Resultados obtenidos y observaciones:

El diagrama de caja y bigote muestra la distribución de los datos, destacando algunos valores atípicos.

Outliers con 1.5 IQR:

Aunque numerosos y visibles a traves del boxplot en la cual contamos 19 datos, los mismos son aquellos que rondan a traves de superar Cota superior 1.5 IQR para Calorías: 700.75, lo cual para ser Calorias de alimentos, obviamente no suena un dato irreal o un error, en especial porque ninguno si quiera supera las 2,000 calorias, por lo tanto mi recomendacion seria no eliminar ninguno aun, aunque es destacable que tantos datos escapen de este 1.5 IQR

Outliers con 3 desv est:

Aunque un poco mayo, mismo caso con el anterior, son datos que en este caso superan Cota superior 3 desviaciones estándar para Calorías: 835.0661, con una cantidad total de 6, lo que nos deja saber que ademas gran parte de los datos concentrados entre 1.5 IQR Y 3 desv est, no superan las 135 calorias de diferencia, es decia de los 19 anteriores, 13 son menores a 835 cal, en cuestion de estos presentamos 6 que son: 1058 1440 1578 942 1061 942; en los cuales aunque es destacable el 1578, podemos suponer que es un dato de alimento altamente calorico, sin observar mas la base de datos podria pensar que se trata de un alimento de comida rapida con altas cantidades de macronutrientes, posiblemente en especial grasas dbeido a su aporte calorico.

Outliers con 3 IQR:

Parecido a los anteriores aqui solo hay dos, 1440 1578, los cuales por ;a justificacion anterior podemos conservar para evitar perdida innecesaria de datos, ya que como mencionamos podria ser perfectamente razonable, ademas que no queremos afectar la naturaleza de los datos

El análisis revela que la variable Calorías contiene un número considerable de datos atípicos y algunos extremos que podrían influir en análisis subsecuentes. Es importante considerar si estos datos atípicos y extremos son representativos de productos específicos que deben analizarse por separado o si deberían transformarse o excluirse en estudios posteriores para evitar sesgos. Aunque en mi opinion lo mas probable es que sean validos y aceptables.

1.2. Para analizar normalidad se te sugiere:

1.2.1 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.

$$H_0: \bar{X} \sim \mathcal{N}(\mu, \sigma^2)$$

$$H_1: \bar{X} \sim/ \mathcal{N}(\mu, \sigma^2)$$

```
library(nortest)
library(tseries)
```

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

```
# Prueba de Anderson-Darling
ad.test(Calorias)
```

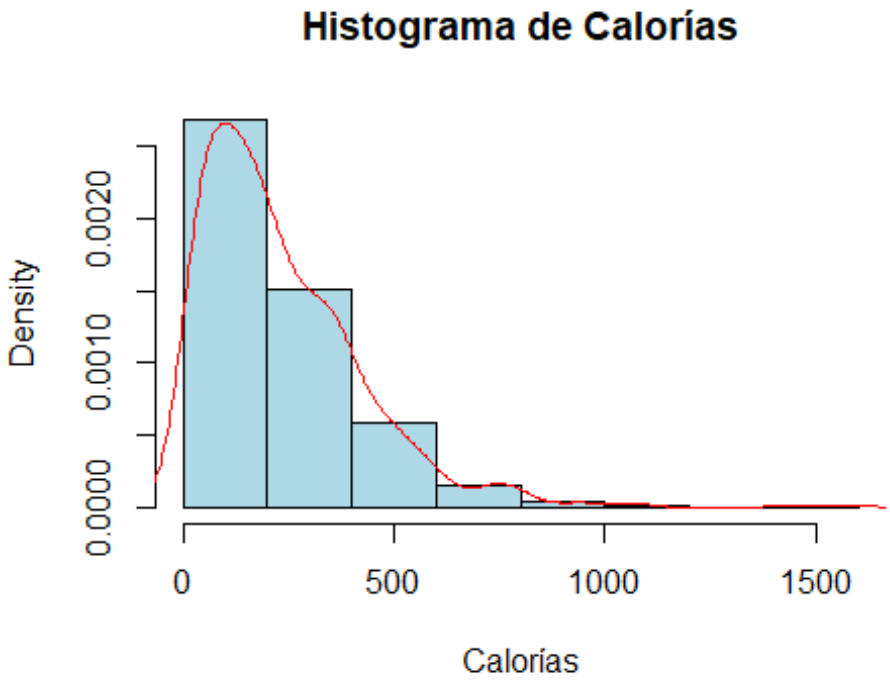
```
##
##  Anderson-Darling normality test
##
## data:  Calorias
## A = 15.326, p-value < 2.2e-16
```

```
# Prueba de Jarque-Bera
jarque.bera.test(Calorias)
```

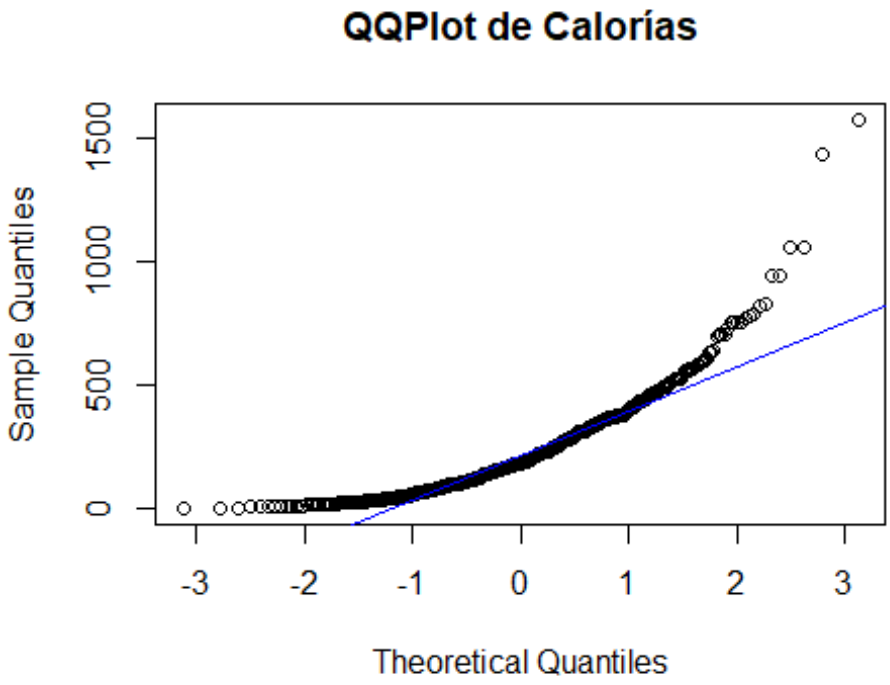
```
##
##  Jarque Bera Test
##
## data:  Calorias
## X-squared = 1388.9, df = 2, p-value < 2.2e-16
```

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

```
# Histograma de los datos
hist(Calorias, main="Histograma de Calorías", xlab="Calorías",
col="lightblue", freq=FALSE)
lines(density(Calorias), col="red")
```




```r
# QQPlot
qqnorm(Calorias, main="QQPlot de Calorías")
qqline(Calorias, col="blue")
```



1.2.3 Calcula el coeficiente de sesgo y el coeficiente de curtosis

```r
library(moments)

# Cálculo de sesgo (skewness) y curtosis (kurtosis)
sesgo <- skewness(Calorias)
curtosis <- kurtosis(Calorias)

cat("Sesgo: ", sesgo, "\n")
```

```
## Sesgo:  1.922735
```

```r
cat("Curtosis: ", curtosis, "\n")
```

```
## Curtosis:  9.760844
```

1.2.4 Compara las medidas de media, mediana y rango medio de cada variable

```r
media <- mean(Calorias)
mediana <- median(Calorias)
rango_medio <- (min(Calorias) + max(Calorias)) / 2

cat("Media: ", media, "\n")
```

```
## Media:  237.3593
```

```r
cat("Mediana: ", mediana, "\n")
```

```
## Mediana:  186
```

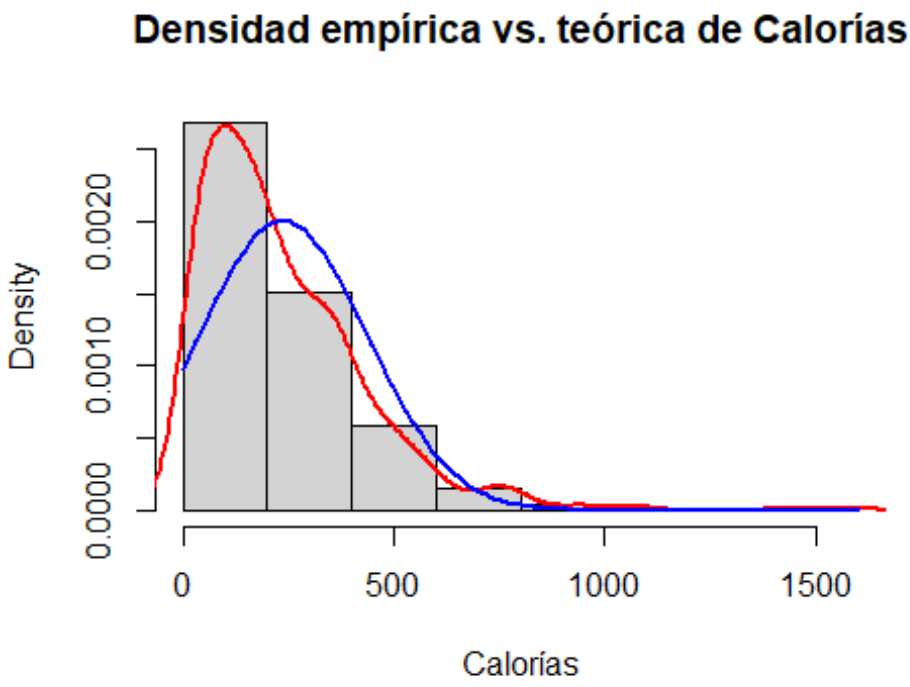
```r
cat("Rango medio: ", rango_medio, "\n")
```

```
## Rango medio:  790.5
```

1.2.5 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
# Gráfico de densidad empírica y teórica
hist(Calorias, freq=FALSE, main="Densidad empírica vs. teórica de Calorías",
xlab="Calorías")
lines(density(Calorias), col="red", lwd=2)
curve(dnorm(x, mean=mean(Calorias), sd=sd(Calorias)), col="blue", lwd=2,
add=TRUE)
```



1.2.6 Interpreta los gráficos y los resultados obtenidos en cada punto con vías a indicar si hay normalidad de los datos

En primer lugar por el Gráfico de densidad empírica y teórica, podemos observar como los datos de Calorias estan lejos de seguir una distribucion normal, con un sesgo importante a la derecha, ya que la mayoria de los datos parece conservarse en la parte izquierda del grafico.

Asimismo podemos ver por el histograma y el Q-Q plot que por ejemplo, aparece muy similar al grafico de densidad, y en el Q-Qplot, estamos observando que parece desviarse en sus colas de la linea de tendencia, esto nos apunta que no sigue una normalidad aparente en especial en sus datos extremos.

1.2.7 Comenta las características encontradas:

1.2.7.1 Considera alejamientos de normalidad por simetría, curtosis

Sesgo:

Un sesgo positivo indica que la distribución tiene una cola larga a la derecha, lo que significa que hay valores extremos altos. El sesgo de 1.92 es considerable, lo que sugiere que la distribución de Calorías está sesgada hacia la derecha. En una distribución normal, el sesgo es cercano a 0. Un valor tan alejado de 0 confirma que la distribución no es simétrica.

Curtosis:

Un valor de 9.76 sugiere que la distribución tiene colas pesadas y es más apuntada que una distribución normal, lo que se traduce en la presencia de datos extremos más frecuente

1.2.7.2 Comenta si hay aparente influencia de los datos atípicos en la normalidad de los datos

Podria ser posible, es decir evidentemente el sesgo a la derecha puede ser ocasionado por la acumulacion de datos que escapan de 1.5 IQR, 3 IQR y 3 desv estandar, sin embargo, debido a que incluso analizamos si habia 0s en Calorias y observamos los datos atipicos, podemos ver que no es que sean un error en si, sino que estos mismos datos podrian explicarnos mas de la verdadera naturaleza de los datos.

1.2.7.3 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.

Pruebas de normalidad:

Prueba de Anderson-Darling: p-value < 2.2e-16 Prueba de Jarque-Bera: p-value < 2.2e-16 Interpretación: Ambas pruebas de normalidad arrojan un valor p extremadamente bajo (menor a 0.05), P value < 0.05 lo que lleva a rechazar la hipótesis nula de que los datos siguen una distribución normal. Esto confirma que los datos de Calorías no son normales en base a las pruebas de normalidad.

Graficos:

Como ya mencionamos en los graficos anteriores en el histograma y grafico de densidad podemos observar un aparente sesgo a la derecha con una linea donde la distribución empírica está sesgada a la derecha, lo cual es consistente con una distribución no normal. Asimismo ya mencionamos que los valores en las colas de Q-Q plot escapan de normalidad aparente.

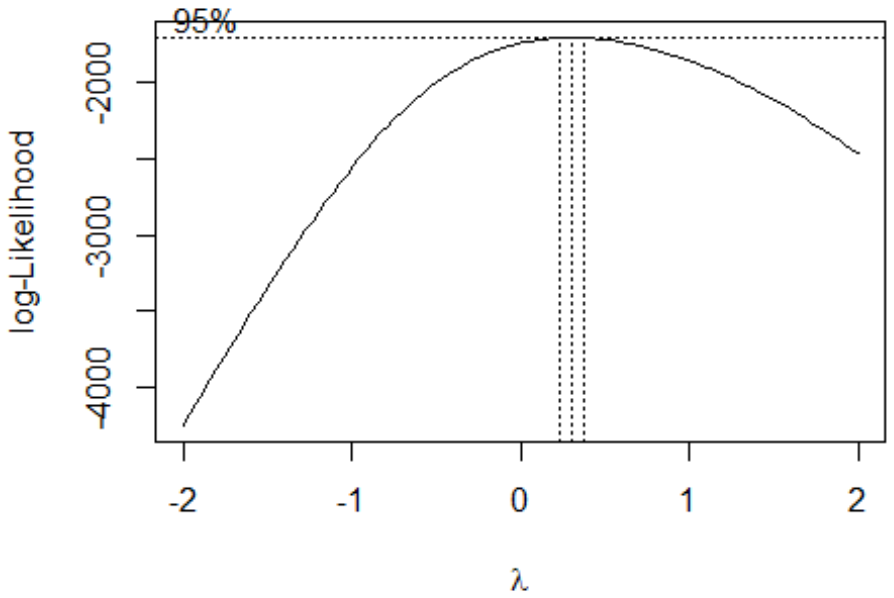
Medidas:

La media y la mediana no están cerca una de la otra, lo que sugiere una distribución asimétrica de los datos. El rango medio es significativamente mayor que la media y la mediana, lo que indica la presencia de valores extremos que influyen en la media. Asimismo comoya vimos lo sucedido con la curtosis donde tiene colas pesadas y es más apuntada que una distribucion normal, asi como el sesgo donde Un valor tan alejado de 0 confirma que la distribución no es simétrica. Y por lo tanto descartamos normalidad por este punto tambien

Punto 2. Transformación a normalidad

2.1 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.

```
library(MASS)          # Para la transformación Box-Cox
library(car)           # Para la transformación Yeo-Johnson

## Loading required package: carData

library(e1071)         # Para el análisis de sesgo y curtosis

##
## Attaching package: 'e1071'

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

library(ggplot2)       # Para la visualización de datos
library(nortest)       # Para la prueba de Anderson-Darling
library(tseries)  # Para la prueba de Jarque-Bera

# Aplicar la transformación Box-Cox
boxcox_model <- boxcox(lm(Calorias ~ 1), lambda = seq(-2, 2, by = 0.1))
```




```
lambda_optimo <- boxcox_model$x[which.max(boxcox_model$y)]
calorias_boxcox <- (Calorias^lambda_optimo - 1) / lambda_optimo  # Modelo
exacto

lambda_optimo

## [1] 0.3030303

# Transformación aproximada
x_transf_aprox <- sqrt(Calorias)
x_transf_aprox

##   [1]  7.141428 14.662878  7.000000  5.477226  5.477226  4.358899
10.770330
##   [8] 10.630146  8.426150  4.358899  4.582576 21.563859  9.899495
11.313708
##  [15] 10.000000  8.660254 10.295630 32.526912 11.661904 10.148892
10.535654
##  [22] 20.049938  9.486833 17.606817 17.720045 19.131126 17.748239
8.660254
##  [29]  9.486833 22.226111  9.899495 22.045408 19.104973 10.000000
22.912878
##  [36] 17.776389  8.944272 12.609520 21.517435 22.181073  8.000000
5.916080
##  [43]  7.483315 10.000000  9.380832 10.049876  9.380832 26.608269
9.273618
##  [50]  7.000000  6.782330  6.928203  9.695360  6.082763  5.196152
```

```
8.062258
##  [57] 17.578396 21.236761 14.933185 15.198684 17.972201 19.672316
20.880613
##  [64] 21.977261 23.000000 10.488088 27.910571 17.262677 15.000000
16.881943
##  [71] 18.708287 19.339080 13.747727 12.449900 16.124515 14.966630
10.000000
##  [78] 19.824228 12.688578 12.569805 12.369317  8.185353 19.646883
37.947332
##  [85] 20.856654 19.183326 18.547237  6.164414 12.649111 10.344080
21.725561
##  [92]  9.273618  9.695360 13.601471 15.165751 19.104973 18.110770
20.880613
##  [99] 14.899664  6.782330 21.447611 24.331050 14.456832 16.462078
14.933185
## [106] 20.297783 17.435596 25.139610 20.248457 14.560220 14.594520
24.657656
## [113] 21.023796 10.148892 10.099505 16.881943  9.165151 21.047565
18.439089
## [120] 21.400935 17.320508 12.041595 18.330303 13.416408 21.633308
15.099669
## [127] 16.822604 26.438608 18.193405 20.615528 13.341664 18.627936
13.747727
## [134] 13.114877 18.761663  5.916080  8.485281 17.748239 14.933185
15.716234
## [141] 16.552945 20.445048 21.771541 17.606817 14.832397 12.489996
12.845233
## [148]  5.477226  7.549834 13.892444  9.643651 22.627417 24.454039
39.724048
## [155] 19.467922 15.716234 22.472205 22.649503 15.198684 19.209373
21.071308
## [162] 12.922848 15.132746 18.384776  7.745967 21.307276 13.490738
13.266499
## [169] 15.968719 16.522712 18.220867 15.066519 12.884099 17.606817
17.776389
## [176] 14.525839 17.776389 13.266499  6.082763 12.884099 14.071247
15.066519
## [183] 11.789826  8.062258 14.035669 16.370706  3.162278 12.727922
16.941074
## [190] 12.961481  4.123106 11.224972 10.954451 15.297059 13.747727
14.106736
## [197] 17.888544  8.831761 20.760539 15.329710 12.165525  8.485281
8.426150
## [204]  5.567764 13.638182  8.366600 13.638182 11.269428 10.000000
12.206556
## [211] 17.549929 13.190906 13.711309 12.688578 11.618950 14.177447
9.539392
## [218] 14.247807 13.638182 10.295630  6.324555 19.519221  4.123106
11.789826
## [225]  9.899495  6.244998 16.822604 12.922848 12.409674  7.348469
```

```
11.269428
## [232] 13.453624 13.711309 15.491933 11.313708 16.763055 10.862780
19.261360
## [239] 12.041595  3.316625 13.190906 11.789826 13.453624 18.303005
3.464102
## [246]  2.828427 13.416408  7.810250  9.273618 15.748016  9.949874
10.630146
## [253] 10.630146  9.746794 12.165525 12.806248 11.180340 19.570386
20.518285
## [260]  8.366600 12.489996 13.601471  9.848858 19.442222 19.570386
14.387495
## [267] 12.569805 17.204651 12.165525 15.811388  8.544004 18.920888
21.908902
## [274] 10.198039 10.246951 14.628739 20.976177 18.411953  5.196152
19.467922
## [281]  8.888194 24.515301 12.767145 13.964240  8.366600 19.467922
9.899495
## [288] 16.941074 19.026298 17.606817 11.357817 10.630146 19.364917
27.856777
## [295]  6.928203 12.288206 16.155494  6.000000 14.142136 16.093477
13.638182
## [302] 24.207437 21.424285 19.493589 17.691806 25.357445 30.692019
19.131126
## [309] 23.937418 22.203603 20.420578 22.891046 11.135529 22.649503
13.152946
## [316] 17.088007 20.297783 32.572995 23.727621 28.861739 19.442222
7.549834
## [323] 19.235384 20.248457 16.062378 22.891046 19.078784 21.748563
15.968719
## [330] 23.916521 28.106939 24.166092 23.853721 30.692019 25.199206
15.842980
## [337] 17.549929 18.027756 17.691806 16.401219 16.124515 16.881943
16.673332
## [344] 18.220867 12.083046 18.654758 11.618950 10.198039 18.867962
5.196152
## [351] 16.278821 12.165525 16.583124 17.117243 14.764823  7.615773
6.082763
## [358] 17.058722 13.747727 10.535654  9.165151 20.174241 27.073973
21.771541
## [365] 19.052559  8.062258 10.862780 14.000000 10.908712 19.339080
4.582576
## [372] 11.789826 18.788294 14.594520  5.477226  8.774964 13.638182
14.764823
## [379] 23.769729 11.958261  7.141428  6.708204 13.747727 15.748016
10.677078
## [386]  8.888194 12.165525  7.416198  6.557439 20.371549 15.198684
10.535654
## [393] 15.362291 13.856406  6.480741 17.146428 16.613248  8.246211
17.029386
## [400] 13.638182 15.297059  9.539392 16.703293 26.627054 13.114877
```

```
19.261360
## [407] 13.190906 27.477263 19.570386 15.968719  4.472136 11.180340
18.681542
## [414] 22.203603 16.278821 27.568098 18.708287 28.705400 19.697716
5.291503
## [421] 26.664583 11.832160  1.732051  6.855655 11.180340 27.549955
19.261360
## [428] 13.564660 14.525839 10.535654 16.673332  5.916080  4.690416
10.816654
## [435]  6.082763 10.198039 13.152946  5.000000 13.152946  7.348469
10.295630
## [442]  6.403124  7.348469 16.643317  5.477226  4.690416 14.106736
4.690416
## [449]  9.486833 10.440307  6.480741 19.364917 13.747727  8.831761
15.937377
## [456]  5.196152 23.366643  6.082763 12.961481  8.185353 27.440845
7.745967
## [463] 23.366643  6.855655  6.480741 18.841444 14.352700  9.110434
20.099751
## [470]  5.567764 15.165751 19.697716 10.295630  9.055385 12.165525
8.888194
## [477] 15.362291  1.732051 21.000000 13.638182 15.427249 16.031220
9.327379
## [484] 15.427249  6.928203 12.961481 18.138357 13.564660 12.727922
15.716234
## [491] 23.706539 13.490738 19.026298 17.606817  2.828427 22.803509
12.845233
## [498] 23.727621 10.099505 23.021729 18.110770 10.488088 22.803509
18.193405
## [505] 17.888544 11.313708 12.000000  8.000000  2.645751  4.358899
11.401754
## [512]  3.316625 11.832160  6.403124  9.433981  5.000000  8.831761
6.403124
## [519]  2.828427 18.547237  4.795832 15.264338 11.357817  9.899495
11.832160
## [526] 11.832160  7.211103  2.000000  8.426150  5.385165 12.845233
5.099020
## [533]  3.162278 10.770330  5.291503  4.242641  6.244998 15.297059
15.066519
## [540] 11.575837  5.385165 13.964240 11.789826  2.828427  7.810250
7.000000
## [547] 12.609520  7.071068  5.744563 12.124356 19.313208

# Transformación exacta
x_transf_exact <- ((Calorias )^lambda_optimo - 1) / lambda_optimo
x_transf_exact

##   [1]  7.563242 13.500162  7.432344  5.949674  5.949674  4.754063
10.635010
##   [8] 10.524802  8.708847  4.754063  5.002071 17.924282  9.940847
```

```
11.056958
##  [15] 10.022157  8.909962 10.259479 23.928565 11.323148 10.142022
10.450193
##  [22] 17.008290  9.603533 15.470335 15.543401 16.439056 15.561566
8.909962
##  [29]  9.603533 18.316971  9.940847 18.210284 16.422698 10.022157
18.719357
##  [36] 15.579691  9.151104 12.032182 17.896578 18.290413  8.337005
6.391999
##  [43]  7.875524 10.022157  9.515959 10.062387  9.515959 20.807956
9.426987
##  [50]  7.432344  7.228826  7.365494  9.774692  6.556588  5.659021
8.391807
##  [57] 15.451966 17.728576 13.687187 13.869593 15.705458 16.775611
17.514133
##  [64] 18.169961 18.770061 10.412536 21.516322 15.247118 13.733211
14.998114
##  [71] 16.173484 16.568817 12.856633 11.914259 14.496068 13.710235
10.022157
##  [78] 16.869424 12.090370 12.002897 11.854501  8.499672 16.759877
26.594651
##  [85] 17.499655 16.471680 16.071712  6.636564 12.061339 10.298115
18.020599
##  [92]  9.426987  9.774692 12.752241 13.847035 16.422698 15.794133
17.514133
##  [99] 13.664068  7.228826 17.854865 19.535547 13.356685 14.720937
13.687187
## [106] 17.160066 15.359495 19.992499 17.129918 13.428777 13.452650
19.720836
## [113] 17.600518 10.142022 10.102341 14.998114  9.336560 17.614836
16.003176
## [120] 17.826950 15.284751 11.609873 15.934075 12.619516 17.965684
13.801714
## [127] 14.959107 20.714675 15.846887 17.353584 12.565706 16.122751
12.856633
## [134] 12.401705 16.207137  6.391999  8.759852 15.561566 13.687187
14.221594
## [141] 14.781158 17.249903 18.047934 15.470335 13.617609 11.943937
12.205250
## [148]  5.949674  7.935625 12.959495  9.732386 18.552687 19.605435
27.435287
## [155] 16.648937 14.221594 18.461716 18.565611 13.869593 16.487946
17.629131
## [162] 12.261963 13.824409 15.968696  8.111629 17.770866 12.672910
12.511473
## [169] 14.391658 14.761135 15.864398 13.778950 12.233666 15.470335
15.579691
## [176] 13.404826 15.579691 12.511473  6.556588 12.233666 13.086005
13.778950
## [183] 11.420154  8.391807 13.060883 14.660249  3.330469 12.119275
```

```
15.036930
## [190] 12.290143  4.487127 10.988606 10.778904 13.936859 12.856633
13.111039
## [197] 15.651793  9.055944 17.441511 13.959148 11.702685  8.759852
8.708847
## [204]  6.042040 12.778485  8.657339 12.778485 11.022876 10.022157
11.733331
## [211] 15.433556 12.456809 12.830680 12.090370 11.290482 13.160846
9.646812
## [218] 13.210308 12.778485 10.259479  6.792219 16.680779  4.487127
11.420154
## [225]  9.940847  6.715087 14.959107 12.261963 11.884448  7.753040
11.022876
## [232] 12.646264 12.830680 14.069610 11.056958 14.919907 10.707382
16.520385
## [239] 11.609873  3.524762 12.456809 11.420154 12.646264 15.916710
3.707104
## [246]  2.896944 12.619516  8.168932  9.426987 14.243060  9.981645
10.524802
## [253] 10.524802  9.816686 11.702685 12.176713 10.954146 16.712505
17.294485
## [260]  8.657339 11.943937 12.752241  9.899758 16.632972 16.712505
13.308222
## [267] 12.002897 15.209309 11.702685 14.285812  8.810365 16.307304
18.129463
## [274] 10.181436 10.220587 13.476445 17.571814 15.985954  5.659021
16.648937
## [281]  9.103734 19.640195 12.148055 13.010370  8.657339 16.648937
9.940847
## [288] 15.036930 16.373434 15.470335 11.090854 10.524802 16.584900
21.487322
## [295]  7.365494 11.794196 14.516781  6.475090 13.135986 14.475298
12.778485
## [302] 19.465165 17.840918 16.664873 15.525196 20.114612 22.987028
16.439056
## [309] 19.310927 18.303701 17.234993 18.706639 10.919494 18.565611
12.429312
## [316] 15.133154 17.160066 23.951938 19.190615 22.025491 16.632972
7.935625
## [323] 16.504181 17.129918 14.454473 18.706639 16.406308 18.034277
14.391658
## [330] 19.298962 21.621992 19.441592 19.262979 22.987028 20.025948
14.307099
## [337] 15.433556 15.741042 15.525196 14.680531 14.496068 14.998114
14.860741
## [344] 15.864398 11.640958 16.139696 11.290482 10.181436 16.274046
5.659021
## [351] 14.599087 11.702685 14.801129 15.152261 13.570856  7.994995
6.556588
## [358] 15.114002 12.856633 10.450193  9.336560 17.084503 21.062804
```

```
18.047934
## [365] 16.389887  8.391807 10.707382 13.035672 10.743248 16.568817
5.002071
## [372] 11.420154 16.223914 13.452650  5.949674  9.007725 12.778485
13.570856
## [379] 19.214796 11.547252  7.563242  7.158934 12.856633 14.243060
10.561762
## [386]  9.103734 11.702685  7.814670  7.015835 17.205098 13.869593
10.450193
## [393] 13.981371 12.933920  6.942540 15.171322 14.821050  8.552765
15.094803
## [400] 12.778485 13.936859  9.646812 14.880513 20.818269 12.401705
16.520385
## [407] 12.456809 21.282105 16.712505 14.391658  4.880229 10.954146
16.156607
## [414] 18.303701 14.599087 21.331324 16.173484 21.942261 16.791317
5.758300
## [421] 20.838865 11.452165  1.303585  7.297667 10.954146 21.321498
16.520385
## [428] 12.725898 13.404826 10.450193 14.860741  6.391999  5.119934
10.671304
## [435]  6.556588 10.181436 12.429312  5.452500 12.429312  7.753040
10.259479
## [442]  6.868018  7.753040 14.840921  5.949674  5.119934 13.111039
5.119934
## [449]  9.603533 10.374640  6.942540 16.584900 12.856633  9.055944
14.370605
## [456]  5.659021 18.982620  6.556588 12.290143  8.499672 21.262354
8.111629
## [463] 18.982620  7.297667  6.942540 16.257368 13.283868  9.290783
17.038854
## [470]  6.042040 13.847035 16.791317 10.259479  9.244621 11.702685
9.103734
## [477] 13.981371  1.303585 17.586178 12.778485 14.025620 14.433592
9.471651
## [484] 14.025620  7.365494 12.290143 15.811755 12.725898 12.119275
14.221594
## [491] 19.178502 12.672910 16.373434 15.470335  2.896944 18.655597
12.205250
## [498] 19.190615 10.102341 18.782695 15.794133 10.412536 18.655597
15.846887
## [505] 15.651793 11.056958 11.578638  8.337005  2.651196  4.754063
11.124569
## [512]  3.524762 11.452165  6.868018  9.559918  5.452500  9.055944
6.868018
## [519]  2.896944 16.071712  5.234120 13.914504 11.090854  9.940847
11.452165
## [526] 11.452165  7.627353  1.722921  8.708847  5.855137 12.205250
5.557145
## [533]  3.330469 10.635010  5.758300  4.623180  6.715087 13.936859
```

```
13.778950
## [540] 11.257646  5.855137 13.010370 11.420154  2.896944  8.168932
7.432344
## [547] 12.032182  7.498249  6.220718 11.671895 16.552703
```

2.2 Escribe las ecuaciones de los modelos de transformación encontrados.

$$\text{Modelo aproximado} = \sqrt{\text{Calorías}}$$

$$\text{Modelo exacto} = \frac{\text{Calorías}^{0.3030303} - 1}{0.3030303}$$

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

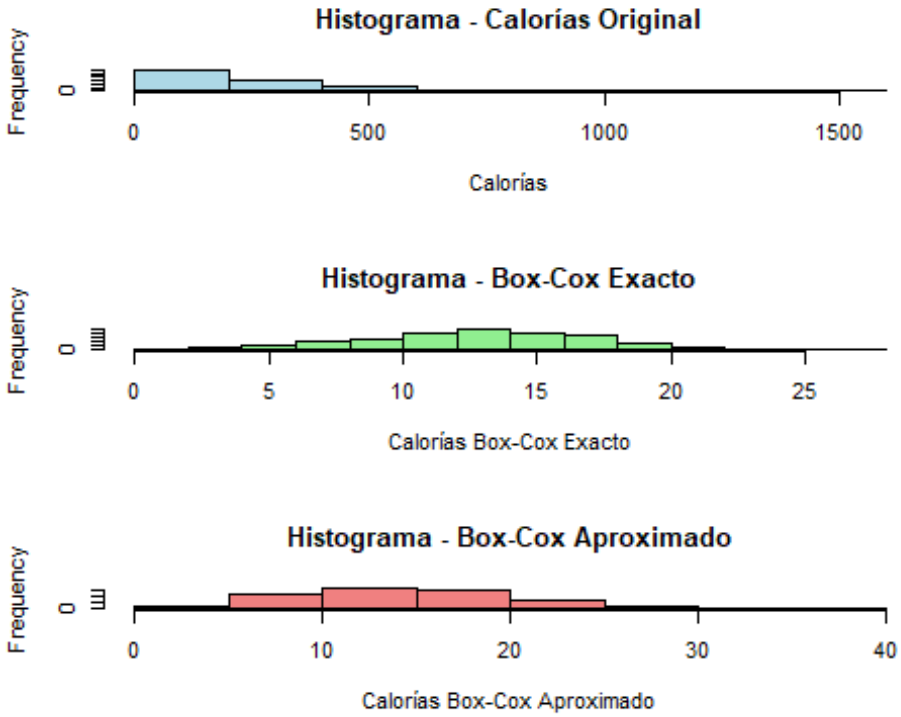
2.3.1 Compara las medidas: Mínimo, máximo, media, mediana, cuartil 1 y cuartil 3, sesgo y curtosis.

```
# Cálculo de medidas estadísticas
medidas_original <- c(min(Calorias), max(Calorias), mean(Calorias),
median(Calorias), quantile(Calorias, c(0.25, 0.75)), skewness(Calorias),
kurtosis(Calorias))
medidas_exacta <- c(min(x_transf_exact), max(x_transf_exact),
mean(x_transf_exact), median(x_transf_exact), quantile(x_transf_exact,
c(0.25, 0.75)), skewness(x_transf_exact), kurtosis(x_transf_exact))
medidas_aproximada <- c(min(x_transf_aprox), max(x_transf_aprox),
mean(x_transf_aprox), median(x_transf_aprox), quantile(x_transf_aprox,
c(0.25, 0.75)), skewness(x_transf_aprox), kurtosis(x_transf_aprox))

# Crear un DataFrame para comparar las medidas
medidas <- data.frame(
  Estadísticas = c("Mínimo", "Máximo", "Media", "Mediana", "Cuartil 1",
"Cuartil 3", "Sesgo", "Curtosis"),
  Original = medidas_original,
  BoxCox_Exacto = medidas_exacta,
  BoxCox_Aproximado = medidas_aproximada
)

# Imprimir el DataFrame
print(medidas)
```

```
##   Estadísticas    Original BoxCox_Exacto BoxCox_Aproximado
## 1       Mínimo    3.000000    1.30358470          1.732051
## 2       Máximo 1578.000000   27.43528700         39.724048
## 3        Media  237.359347   12.73576346         14.132788
## 4      Mediana  186.000000   12.77848538         13.638182
## 5    Cuartil 1   94.500000    9.79568908          9.721077
## 6    Cuartil 3  337.000000   15.95138551         18.357540
## 7        Sesgo    1.917503   -0.02223906          0.476366
## 8     Curtosis    6.725447   -0.18683613          0.341690
```

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

```
# Graficar histogramas para los datos originales y transformados
par(mfrow = c(3, 1))  # Dividir la ventana gráfica en 3 filas

# Histograma para los datos originales
hist(Calorias, main = "Histograma - Calorías Original", xlab = "Calorías",
col = "lightblue")

# Histograma para los datos transformados con Box-Cox Exacto
hist(x_transf_exact, main = "Histograma - Box-Cox Exacto", xlab = "Calorías
Box-Cox Exacto", col = "lightgreen")

# Histograma para los datos transformados con Box-Cox Aproximado
hist(x_transf_aprox, main = "Histograma - Box-Cox Aproximado", xlab =
"Calorías Box-Cox Aproximado", col = "lightcoral")
```




```
# Restaurar la ventana gráfica a su configuración original
par(mfrow = c(1, 1))
```

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

```
# Pruebas de normalidad
anderson_test_original <- ad.test(Calorias)
anderson_test_boxcox <- ad.test(x_transf_exact)
anderson_test_boxcox_aprox <- ad.test(x_transf_aprox)
```

```r
# Mostrar los resultados
cat("Prueba de Anderson-Darling para los datos originales: p-value =",
anderson_test_original$p.value, "\n")

## Prueba de Anderson-Darling para los datos originales: p-value = 3.7e-24

cat("Prueba de Anderson-Darling para Box-Cox exacto: p-value =",
anderson_test_boxcox$p.value, "\n")

## Prueba de Anderson-Darling para Box-Cox exacto: p-value = 0.1328423

cat("Prueba de Anderson-Darling para Box-Cox aproximado: p-value =",
anderson_test_boxcox_aprox$p.value, "\n")

## Prueba de Anderson-Darling para Box-Cox aproximado: p-value = 0.002964427

# Prueba de Jarque-Bera
jarque_bera_original <- jarque.bera.test(Calorias)
jarque_bera_boxcox <- jarque.bera.test(x_transf_exact)
jarque_bera_boxcox_aprox <- jarque.bera.test(x_transf_aprox)


# Mostrar los resultados de Jarque-Bera
cat("Prueba de Jarque-Bera para los datos originales: p-value =",
jarque_bera_original$p.value, "\n")

## Prueba de Jarque-Bera para los datos originales: p-value = 0

cat("Prueba de Jarque-Bera para Box-Cox exacto: p-value =",
jarque_bera_boxcox$p.value, "\n")

## Prueba de Jarque-Bera para Box-Cox exacto: p-value = 0.6832947

cat("Prueba de Jarque-Bera para Box-Cox aproximado: p-value =",
jarque_bera_boxcox_aprox$p.value, "\n")

## Prueba de Jarque-Bera para Box-Cox aproximado: p-value = 6.696789e-06
```

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

```r
# Filtrar y contar los valores de Calorías iguales a 0
calorias_cero <- Calorias[Calorias == 0]
length(calorias_cero)

## [1] 0

# Detectar outliers utilizando IQR para la variable Calorias
IQR_calorias <- IQR(Calorias)
limite_inferior <- quantile(Calorias, 0.25) - 3 * IQR_calorias
limite_superior <- quantile(Calorias, 0.75) + 3 * IQR_calorias
```

```r
# Identificar outliers
outliers_calorias <- Calorias[Calorias < limite_inferior | Calorias > 
limite_superior]

# Mostrar los outliers detectados
cat("Outliers detectados en Calorias:\n")
```

```
## Outliers detectados en Calorias:
```

```r
print(outliers_calorias)
```

```
## [1] 1440 1578
```

Podriamos si acaso borrar estos que escapan mucho aunque realmente no lo aconsejaria, debido a que puede que expliquen parte de la naturaleza de los datos, sin embargo como el objetivo es normalidad si los quitamos

```r
# Filtrar los datos para eliminar los outliers
Calorias_sin_outliers <- Calorias[Calorias >= limite_inferior & Calorias <= 
limite_superior]

# Mostrar los datos sin outliers
cat("Datos de Calorias sin outliers:\n")
```

```
## Datos de Calorias sin outliers:
```

```r
print(Calorias_sin_outliers)
```

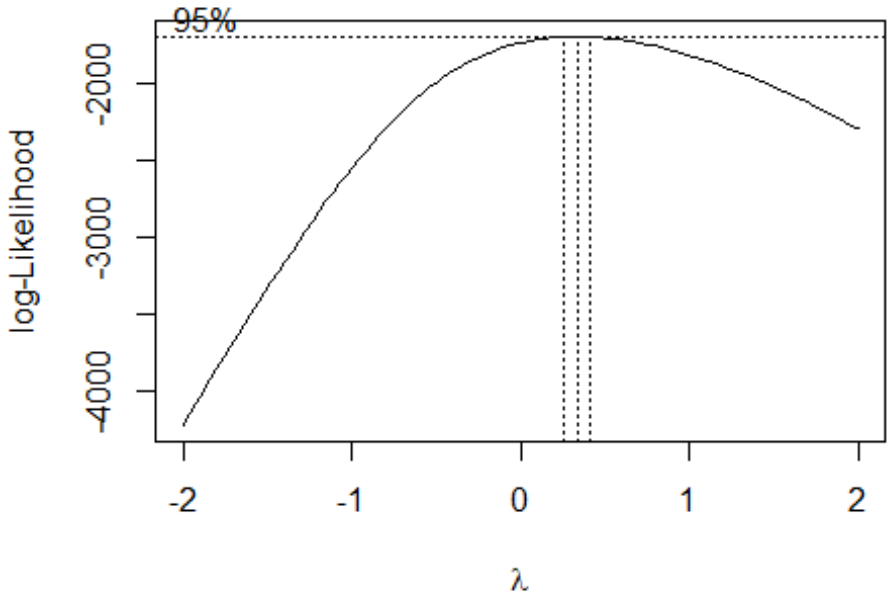
```
##   [1]   51  215   49   30   30   19  116  113   71   19   21  465   98
128  100
##  [16]   75  106 1058  136  103  111  402   90  310  314  366  315   75
90  494
##  [31]   98  486  365  100  525  316   80  159  463  492   64   35   56
100   88
##  [46]  101   88  708   86   49   46   48   94   37   27   65  309  451
223  231
##  [61]  323  387  436  483  529  110  779  298  225  285  350  374  189
155  260
##  [76]  224  100  393  161  158  153   67  386  435  368  344   38  160
107  472
##  [91]   86   94  185  230  365  328  436  222   46  460  592  209  271
223  412
## [106]  304  632  410  212  213  608  442  103  102  285   84  443  340
458  300
## [121]  145  336  180  468  228  283  699  331  425  178  347  189  172
352   35
## [136]   72  315  223  247  274  418  474  310  220  156  165   30   57
193   93
## [151]  512  598  379  247  505  513  231  369  444  167  229  338   60
454  182
## [166]  176  255  273  332  227  166  310  316  211  316  176   37  166
```

```
198  227
## [181]  139   65  197  268   10  162  287  168   17  126  120  234  189
199  320
## [196]   78  431  235  148   72   71   31  186   70  186  127  100  149
308  174
## [211]  188  161  135  201   91  203  186  106   40  381   17  139   98
39  283
## [226]  167  154   54  127  181  188  240  128  281  118  371  145   11
174  139
## [241]  181  335   12    8  180   61   86  248   99  113  113   95  148
164  125
## [256]  383  421   70  156  185   97  378  383  207  158  296  148  250
73  358
## [271]  480  104  105  214  440  339   27  379   79  601  163  195   70
379   98
## [286]  287  362  310  129  113  375  776   48  151  261   36  200  259
186  586
## [301]  459  380  313  643  942  366  573  493  417  524  124  513  173
292  412
## [316] 1061  563  833  378   57  370  410  258  524  364  473  255  572
790  584
## [331]  569  942  635  251  308  325  313  269  260  285  278  332  146
348  135
## [346]  104  356   27  265  148  275  293  218   58   37  291  189  111
84  407
## [361]  733  474  363   65  118  196  119  374   21  139  353  213   30
77  186
## [376]  218  565  143   51   45  189  248  114   79  148   55   43  415
231  111
## [391]  236  192   42  294  276   68  290  186  234   91  279  709  172
371  174
## [406]  755  383  255   20  125  349  493  265  760  350  824  388   28
711  140
## [421]    3   47  125  759  371  184  211  111  278   35   22  117   37
104  173
## [436]   25  173   54  106   41   54  277   30   22  199   22   90  109
42  375
## [451]  189   78  254   27  546   37  168   67  753   60  546   47   42
355  206
## [466]   83  404   31  230  388  106   82  148   79  236    3  441  186
238  257
## [481]   87  238   48  168  329  184  162  247  562  182  362  310    8
520  165
## [496]  563  102  530  328  110  520  331  320  128  144   64    7   19
130   11
## [511]  140   41   89   25   78   41    8  344   23  233  129   98  140
140   52
## [526]    4   71   29  165   26   10  116   28   18   39  234  227  134
29  195
## [541]  139    8   61   49  159   50   33  147  373
```

```r
# Verificar la cantidad de datos eliminados
cat("Cantidad de datos originales: ", length(Calorias), "\n")

## Cantidad de datos originales:  551

cat("Cantidad de datos sin outliers: ", length(Calorias_sin_outliers), "\n")

## Cantidad de datos sin outliers:  549

Calorias <- Calorias_sin_outliers
```

2.5 Comenta la normalidad de las transformaciones obtenidas. Utiliza como argumento de normalidad:

```r
# Aplicar la transformación Box-Cox
boxcox_model <- boxcox(lm(Calorias ~ 1), lambda = seq(-2, 2, by = 0.1))
```




```r
lambda_optimo <- boxcox_model$x[which.max(boxcox_model$y)]
calorias_boxcox <- (Calorias^lambda_optimo - 1) / lambda_optimo  # Modelo exacto

lambda_optimo

## [1] 0.3434343

# Transformación aproximada
x_transf_aprox <- sqrt(Calorias)
x_transf_aprox
```

```
##   [1]  7.141428 14.662878  7.000000  5.477226  5.477226  4.358899
10.770330
##   [8] 10.630146  8.426150  4.358899  4.582576 21.563859  9.899495
11.313708
##  [15] 10.000000  8.660254 10.295630 32.526912 11.661904 10.148892
10.535654
##  [22] 20.049938  9.486833 17.606817 17.720045 19.131126 17.748239
8.660254
##  [29]  9.486833 22.226111  9.899495 22.045408 19.104973 10.000000
22.912878
##  [36] 17.776389  8.944272 12.609520 21.517435 22.181073  8.000000
5.916080
##  [43]  7.483315 10.000000  9.380832 10.049876  9.380832 26.608269
9.273618
##  [50]  7.000000  6.782330  6.928203  9.695360  6.082763  5.196152
8.062258
##  [57] 17.578396 21.236761 14.933185 15.198684 17.972201 19.672316
20.880613
##  [64] 21.977261 23.000000 10.488088 27.910571 17.262677 15.000000
16.881943
##  [71] 18.708287 19.339080 13.747727 12.449900 16.124515 14.966630
10.000000
##  [78] 19.824228 12.688578 12.569805 12.369317  8.185353 19.646883
20.856654
##  [85] 19.183326 18.547237  6.164414 12.649111 10.344080 21.725561
9.273618
##  [92]  9.695360 13.601471 15.165751 19.104973 18.110770 20.880613
14.899664
##  [99]  6.782330 21.447611 24.331050 14.456832 16.462078 14.933185
20.297783
## [106] 17.435596 25.139610 20.248457 14.560220 14.594520 24.657656
21.023796
## [113] 10.148892 10.099505 16.881943  9.165151 21.047565 18.439089
21.400935
## [120] 17.320508 12.041595 18.330303 13.416408 21.633308 15.099669
16.822604
## [127] 26.438608 18.193405 20.615528 13.341664 18.627936 13.747727
13.114877
## [134] 18.761663  5.916080  8.485281 17.748239 14.933185 15.716234
16.552945
## [141] 20.445048 21.771541 17.606817 14.832397 12.489996 12.845233
5.477226
## [148]  7.549834 13.892444  9.643651 22.627417 24.454039 19.467922
15.716234
## [155] 22.472205 22.649503 15.198684 19.209373 21.071308 12.922848
15.132746
## [162] 18.384776  7.745967 21.307276 13.490738 13.266499 15.968719
16.522712
## [169] 18.220867 15.066519 12.884099 17.606817 17.776389 14.525839
17.776389
```

```
## [176] 13.266499  6.082763 12.884099 14.071247 15.066519 11.789826
8.062258
## [183] 14.035669 16.370706  3.162278 12.727922 16.941074 12.961481
4.123106
## [190] 11.224972 10.954451 15.297059 13.747727 14.106736 17.888544
8.831761
## [197] 20.760539 15.329710 12.165525  8.485281  8.426150  5.567764
13.638182
## [204]  8.366600 13.638182 11.269428 10.000000 12.206556 17.549929
13.190906
## [211] 13.711309 12.688578 11.618950 14.177447  9.539392 14.247807
13.638182
## [218] 10.295630  6.324555 19.519221  4.123106 11.789826  9.899495
6.244998
## [225] 16.822604 12.922848 12.409674  7.348469 11.269428 13.453624
13.711309
## [232] 15.491933 11.313708 16.763055 10.862780 19.261360 12.041595
3.316625
## [239] 13.190906 11.789826 13.453624 18.303005  3.464102  2.828427
13.416408
## [246]  7.810250  9.273618 15.748016  9.949874 10.630146 10.630146
9.746794
## [253] 12.165525 12.806248 11.180340 19.570386 20.518285  8.366600
12.489996
## [260] 13.601471  9.848858 19.442222 19.570386 14.387495 12.569805
17.204651
## [267] 12.165525 15.811388  8.544004 18.920888 21.908902 10.198039
10.246951
## [274] 14.628739 20.976177 18.411953  5.196152 19.467922  8.888194
24.515301
## [281] 12.767145 13.964240  8.366600 19.467922  9.899495 16.941074
19.026298
## [288] 17.606817 11.357817 10.630146 19.364917 27.856777  6.928203
12.288206
## [295] 16.155494  6.000000 14.142136 16.093477 13.638182 24.207437
21.424285
## [302] 19.493589 17.691806 25.357445 30.692019 19.131126 23.937418
22.203603
## [309] 20.420578 22.891046 11.135529 22.649503 13.152946 17.088007
20.297783
## [316] 32.572995 23.727621 28.861739 19.442222  7.549834 19.235384
20.248457
## [323] 16.062378 22.891046 19.078784 21.748563 15.968719 23.916521
28.106939
## [330] 24.166092 23.853721 30.692019 25.199206 15.842980 17.549929
18.027756
## [337] 17.691806 16.401219 16.124515 16.881943 16.673332 18.220867
12.083046
## [344] 18.654758 11.618950 10.198039 18.867962  5.196152 16.278821
12.165525
```

```
## [351] 16.583124 17.117243 14.764823  7.615773  6.082763 17.058722
13.747727
## [358] 10.535654  9.165151 20.174241 27.073973 21.771541 19.052559
8.062258
## [365] 10.862780 14.000000 10.908712 19.339080  4.582576 11.789826
18.788294
## [372] 14.594520  5.477226  8.774964 13.638182 14.764823 23.769729
11.958261
## [379]  7.141428  6.708204 13.747727 15.748016 10.677078  8.888194
12.165525
## [386]  7.416198  6.557439 20.371549 15.198684 10.535654 15.362291
13.856406
## [393]  6.480741 17.146428 16.613248  8.246211 17.029386 13.638182
15.297059
## [400]  9.539392 16.703293 26.627054 13.114877 19.261360 13.190906
27.477263
## [407] 19.570386 15.968719  4.472136 11.180340 18.681542 22.203603
16.278821
## [414] 27.568098 18.708287 28.705400 19.697716  5.291503 26.664583
11.832160
## [421]  1.732051  6.855655 11.180340 27.549955 19.261360 13.564660
14.525839
## [428] 10.535654 16.673332  5.916080  4.690416 10.816654  6.082763
10.198039
## [435] 13.152946  5.000000 13.152946  7.348469 10.295630  6.403124
7.348469
## [442] 16.643317  5.477226  4.690416 14.106736  4.690416  9.486833
10.440307
## [449]  6.480741 19.364917 13.747727  8.831761 15.937377  5.196152
23.366643
## [456]  6.082763 12.961481  8.185353 27.440845  7.745967 23.366643
6.855655
## [463]  6.480741 18.841444 14.352700  9.110434 20.099751  5.567764
15.165751
## [470] 19.697716 10.295630  9.055385 12.165525  8.888194 15.362291
1.732051
## [477] 21.000000 13.638182 15.427249 16.031220  9.327379 15.427249
6.928203
## [484] 12.961481 18.138357 13.564660 12.727922 15.716234 23.706539
13.490738
## [491] 19.026298 17.606817  2.828427 22.803509 12.845233 23.727621
10.099505
## [498] 23.021729 18.110770 10.488088 22.803509 18.193405 17.888544
11.313708
## [505] 12.000000  8.000000  2.645751  4.358899 11.401754  3.316625
11.832160
## [512]  6.403124  9.433981  5.000000  8.831761  6.403124  2.828427
18.547237
## [519]  4.795832 15.264338 11.357817  9.899495 11.832160 11.832160
7.211103
```

```
## [526]  2.000000  8.426150  5.385165 12.845233  5.099020  3.162278
10.770330
## [533]  5.291503  4.242641  6.244998 15.297059 15.066519 11.575837
5.385165
## [540] 13.964240 11.789826  2.828427  7.810250  7.000000 12.609520
7.071068
## [547]  5.744563 12.124356 19.313208

# Transformación exacta
x_transf_exact <- ((Calorias )^lambda_optimo - 1) / lambda_optimo
x_transf_exact

##   [1]  8.323790 15.504254  8.170478  6.451999  6.451999  5.092550
11.987404
##   [8] 11.853931  9.675807  5.092550  5.372459 21.090445 11.149089
12.499722
##  [15] 11.246987  9.914988 11.533179 28.920573 12.823962 11.391452
11.763652
##  [22] 19.919867 10.743819 17.970397 18.062545 19.195943 18.085461
9.914988
##  [29] 10.743819 21.594359 11.149089 21.457333 19.175180 11.246987
22.111986
##  [36] 18.108330 10.202464 13.691430 21.054941 21.560240  9.234997
6.961079
##  [43]  8.690535 11.246987 10.638832 11.295454 10.638832 24.818600
10.532266
##  [50]  8.170478  7.932608  8.092278 10.949286  7.151308  6.119233
9.299847
##  [57] 17.947238 20.839769 15.736774 15.963881 18.267096 19.623626
20.565451
##  [64] 21.405566 22.177301 11.718111 25.743838 17.689179 15.794046
17.376005
##  [71] 18.859147 19.360724 14.706837 13.546779 16.746314 15.765452
11.246987
##  [78] 19.743012 13.762860 13.655493 13.473533  9.427608 19.603611
20.546944
##  [85] 19.237358 18.730244  7.243897 13.727218 11.579836 21.213928
10.532266
##  [92] 10.949286 14.577877 15.935778 19.175180 18.379123 20.565451
15.708012
##  [99]  7.932608 21.001494 23.165784 15.326109 17.028067 15.736774
20.113349
## [106] 17.830699 23.757971 20.074901 15.415595 15.445239 23.405720
20.675910
## [113] 11.391452 11.343607 17.376005 10.424060 20.694224 18.643486
20.965736
## [120] 17.736558 13.174095 18.556055 14.414076 21.143515 15.879330
17.326997
## [127] 24.697029 18.445801 20.360321 14.347719 18.794878 14.706837
14.145662
```

```
## [134] 18.901792  6.961079  9.736415 18.085461 15.736774 16.403052
17.103602
## [141] 20.227962 21.248988 17.970397 15.650231 13.583169 13.903992
6.451999
## [148]  8.761276 14.834018 10.898466 21.897428 23.256253 19.462540
16.403052
## [155] 21.780412 21.914058 15.963881 19.258011 20.712511 13.973717
15.907594
## [162] 18.599856  8.968729 20.893911 14.479950 14.280870 16.615653
17.078484
## [169] 18.467939 15.850984 13.938923 17.970397 18.108330 15.385859
18.108330
## [176] 14.280870  7.151308 13.938923 14.990583 15.850984 12.942319
9.299847
## [183] 14.959479 16.951981  3.509057 13.798357 17.424788 14.008374
4.792562
## [190] 12.416594 12.161889 16.047713 14.706837 15.021583 18.199333
10.088930
## [197] 20.472636 16.075500 13.287626  9.736415  9.675807  6.558042
14.610287
## [204]  9.614636 14.610287 12.458266 11.246987 13.325134 17.924029
14.213521
## [211] 14.674766 13.762860 12.784129 15.083279 10.795739 15.144573
14.610287
## [218] 11.533179  7.424383 19.503019  4.792562 12.942319 11.149089
7.334900
## [225] 17.326997 13.973717 13.510234  8.546525 12.458266 14.447073
14.674766
## [232] 16.213285 12.499722 17.277761 12.075132 19.299205 13.174095
3.722706
## [239] 14.213521 12.942319 14.447073 18.534091  3.923954  3.035378
14.414076
## [246]  9.036363 10.532266 16.429872 11.198200 11.853931 11.853931
10.999752
## [253] 13.287626 13.868922 12.374705 19.543359 20.284864  9.614636
13.583169
## [260] 14.577877 11.099648 19.442247 19.543359 15.265981 13.655493
17.641590
## [267] 13.287626 16.483300  9.796473 19.028781 21.353588 11.438992
11.486233
## [274] 15.474792 20.639201 18.621692  6.119233 19.462540 10.145932
23.301264
## [281] 13.833710 14.896960  9.614636 19.462540 11.149089 17.424788
19.112665
## [288] 17.970397 12.540967 11.853931 19.381158 25.705890  8.092278
13.399656
## [295] 16.772247  7.057061 15.052482 16.720314 14.610287 23.074711
20.983628
## [302] 19.482797 18.039580 23.916487 27.675983 19.195943 22.875264
21.577311
```

```
## [309] 20.208936 22.095606 12.332595 21.914058 14.179656 17.545776
20.113349
## [316] 28.951543 22.719812 26.411078 19.442247  8.761276 19.278626
20.074901
## [323] 16.694249 22.095606 19.154379 21.231470 16.615653 22.859799
25.882165
## [330] 23.044217 22.813298 27.675983 23.801381 16.509909 17.924029
18.312042
## [337] 18.039580 16.977405 16.746314 17.376005 17.203474 18.467939
13.212109
## [344] 18.816342 12.784129 11.438992 18.986607  6.119233 16.875335
13.287626
## [351] 17.128659 17.569810 15.592104  8.831207  7.151308 17.521688
14.706837
## [358] 11.763652 10.424060 20.016998 25.151060 21.248988 19.133541
9.299847
## [365] 12.075132 14.928272 12.118630 19.360724  5.372459 12.942319
18.923055
## [372] 15.445239  6.451999 10.031445 14.610287 15.592104 22.751047
13.097548
## [379]  8.323790  7.851059 14.706837 16.429872 11.898678 10.145932
13.287626
## [386]  8.618960  7.684321 20.170792 15.963881 11.763652 16.103210
14.802386
## [393]  7.599037 17.593790 17.153657  9.490551 17.497545 14.610287
16.047713
## [400] 10.795739 17.228295 24.832045 14.145662 19.299205 14.213521
25.437520
## [407] 19.543359 16.615653  5.234803 12.374705 18.837764 21.577311
16.875335
## [414] 25.501858 18.859147 26.301886 19.643608  6.232737 24.858898
12.981399
## [421]  1.334588  8.013000 12.374705 25.489012 19.299205 14.545351
15.385859
## [428] 11.763652 17.203474  6.961079  5.505875 12.031391  7.151308
11.438992
## [435] 14.179656  5.883661 14.179656  8.546525 11.533179  7.512409
8.546525
## [442] 17.178595  6.451999  5.505875 15.021583  5.505875 10.743819
11.672298
## [449]  7.599037 19.381158 14.706837 10.088930 16.589320  6.119233
22.451330
## [456]  7.151308 14.008374  9.427608 25.411706  8.968729 22.451330
8.013000
## [463]  7.599037 18.965463 15.235775 10.369322 19.958814  6.558042
15.935778
## [470] 19.643608 11.533179 10.314149 13.287626 10.145932 16.103210
1.334588
## [477] 20.657569 14.610287 16.158399 16.668118 10.585750 16.158399
8.092278
```

```
## [484] 14.008374 18.401393 14.545351 13.798357 16.403052 22.704167
14.479950
## [491] 19.112665 17.970397  3.035378 22.029881 13.903992 22.719812
11.343607
## [498] 22.193579 18.379123 11.718111 22.029881 18.445801 18.199333
12.499722
## [505] 13.135909  9.234997  2.768806  5.092550 12.582002  3.722706
12.981399
## [512]  7.512409 10.691519  5.883661 10.088930  7.512409  3.035378
18.730244
## [519]  5.635367 16.019848 12.540967 11.149089 12.981399 12.981399
8.398969
## [526]  1.775553  9.675807  6.343609 13.903992  6.002935  3.509057
11.987404
## [533]  6.232737  4.945293  7.334900 16.047713 15.850984 12.744102
6.343609
## [540] 14.896960 12.942319  3.035378  9.036363  8.170478 13.691430
8.247637
## [547]  6.763573 13.249952 19.340254
```

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

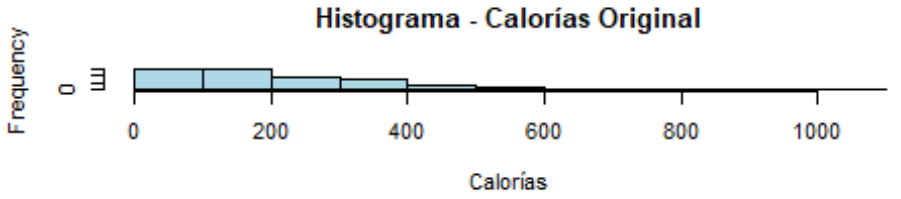
```
# Cálculo de medidas estadísticas
medidas_original <- c(min(Calorias), max(Calorias), mean(Calorias),
median(Calorias), quantile(Calorias, c(0.25, 0.75)), skewness(Calorias),
kurtosis(Calorias))
medidas_exacta <- c(min(x_transf_exact), max(x_transf_exact),
mean(x_transf_exact), median(x_transf_exact), quantile(x_transf_exact,
c(0.25, 0.75)), skewness(x_transf_exact), kurtosis(x_transf_exact))
medidas_aproximada <- c(min(x_transf_aprox), max(x_transf_aprox),
mean(x_transf_aprox), median(x_transf_aprox), quantile(x_transf_aprox,
c(0.25, 0.75)), skewness(x_transf_aprox), kurtosis(x_transf_aprox))

# Crear un DataFrame para comparar las medidas
medidas <- data.frame(
  Estadísticas = c("Mínimo", "Máximo", "Media", "Mediana", "Cuartil 1",
"Cuartil 3", "Sesgo", "Curtosis"),
  Original = medidas_original,
  BoxCox_Exacto = medidas_exacta,
  BoxCox_Aproximado = medidas_aproximada
)

# Imprimir el DataFrame
print(medidas)
```

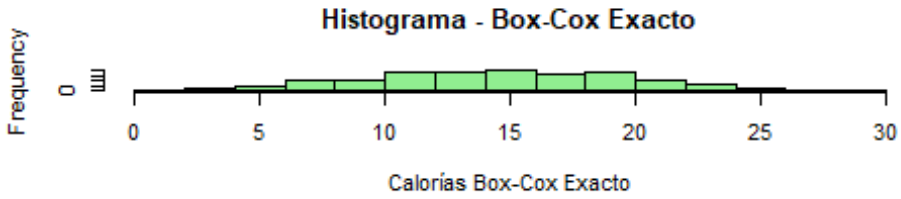
```
##   Estadísticas    Original BoxCox_Exacto BoxCox_Aproximado
## 1       Mínimo    3.000000    1.33458837         1.7320508
## 2       Máximo 1061.000000   28.95154273        32.5729949
## 3        Media  232.726776   14.59343798        14.0427952
## 4      Mediana  186.000000   14.61028732        13.6381817
```

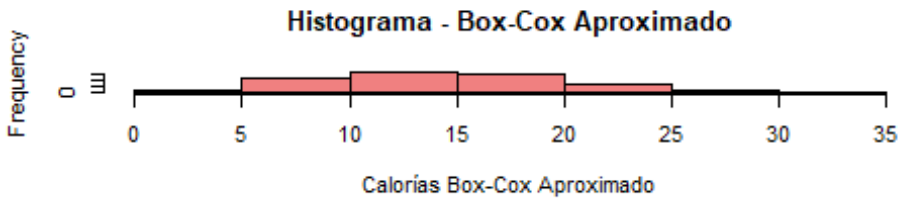
```
## 5    Cuartil 1   94.000000   10.94928582         9.6953597
## 6    Cuartil 3  335.000000   18.53409117        18.3030052
## 7        Sesgo    1.303081   -0.02749767         0.3065846
## 8     Curtosis    2.074965   -0.43311575        -0.3006685
```

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

```
# Graficar histogramas para los datos originales y transformados
par(mfrow = c(3, 1))  # Dividir la ventana gráfica en 3 filas

# Histograma para los datos originales
hist(Calorias, main = "Histograma - Calorías Original", xlab = "Calorías",
col = "lightblue")

# Histograma para los datos transformados con Box-Cox Exacto
hist(x_transf_exact, main = "Histograma - Box-Cox Exacto", xlab = "Calorías
Box-Cox Exacto", col = "lightgreen")

# Histograma para los datos transformados con Box-Cox Aproximado
hist(x_transf_aprox, main = "Histograma - Box-Cox Aproximado", xlab =
"Calorías Box-Cox Aproximado", col = "lightcoral")
```








```
# Restaurar la ventana gráfica a su configuración original
par(mfrow = c(1, 1))
```

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

```
# Pruebas de normalidad
anderson_test_original <- ad.test(Calorias)
anderson_test_boxcox <- ad.test(x_transf_exact)
anderson_test_boxcox_aprox <- ad.test(x_transf_aprox)

# Mostrar los resultados
cat("Prueba de Anderson-Darling para los datos originales: p-value =",
anderson_test_original$p.value, "\n")

## Prueba de Anderson-Darling para los datos originales: p-value = 3.7e-24

cat("Prueba de Anderson-Darling para Box-Cox exacto: p-value =",
anderson_test_boxcox$p.value, "\n")

## Prueba de Anderson-Darling para Box-Cox exacto: p-value = 0.1215676

cat("Prueba de Anderson-Darling para Box-Cox aproximado: p-value =",
anderson_test_boxcox_aprox$p.value, "\n")

## Prueba de Anderson-Darling para Box-Cox aproximado: p-value = 0.005091513

# Prueba de Jarque-Bera
jarque_bera_original <- jarque.bera.test(Calorias)
jarque_bera_boxcox <- jarque.bera.test(x_transf_exact)
jarque_bera_boxcox_aprox <- jarque.bera.test(x_transf_aprox)


# Mostrar los resultados de Jarque-Bera
cat("Prueba de Jarque-Bera para los datos originales: p-value =",
jarque_bera_original$p.value, "\n")

## Prueba de Jarque-Bera para los datos originales: p-value = 0

cat("Prueba de Jarque-Bera para Box-Cox exacto: p-value =",
jarque_bera_boxcox$p.value, "\n")

## Prueba de Jarque-Bera para Box-Cox exacto: p-value = 0.1238803

cat("Prueba de Jarque-Bera para Box-Cox aproximado: p-value =",
jarque_bera_boxcox_aprox$p.value, "\n")

## Prueba de Jarque-Bera para Box-Cox aproximado: p-value = 0.005036342
```

2.5.4 Indica posibilidades de motivos de alejamiento de normalidad (sesgo, curtosis, datos atípicos, etc)

Podemos observar como el Sesgo y la curtosis estan significativamente alejadas de 0 lo cual podria ser un motivo de alejamiento d enormalidad, sinceramente, la cantidad de datos atipicos los cuales podemos observar ya una normalidad sin quitar nada despues de la trasnformacion de box-cox con modelo exacto nos hacen ver por ejemplo a traves del

histograma como una distribucion normal, asi como los valores obtenidos en las pruebas de normalidad de anderson y jarque con p value > 0.05 lo que no nos da suficiente evidencia para rechazar normalidad, en cambio los otros, con el modelo aproximado aunque mejora un poco con respecto del original no es suficiente para no rechazar la hipotesis nula y termina descartandose normalidad por ese medio tambien.

2.6 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.

Los resultados de las pruebas de normalidad indican que, aunque eliminar los outliers mejora ligeramente la normalidad de los datos, esta mejora no es significativa. Sin embargo, la transformación utilizando el modelo exacto de Box-Cox con λ= 0.3030303 asi como el de 0.34 proporciona una mejora considerable en la normalidad de los datos en comparación con los datos originales y con el modelo aproximado. Esto se refleja en los valores p de las pruebas de Anderson-Darling y Jarque-Bera, donde el modelo exacto se aproxima mucho más a una distribución normal.

La transformación exacta de Box-Cox proporciona la mayor mejora en la normalización de los datos, creando una distribución más simétrica y centrada en comparación con los datos originales y la transformación aproximada. y Realmente si hacemos la comparactiva con los histogramas no hace una diferencia significativa.