Actividad 5.

Adrian Pineda Sanchez

2024-08-14

```
M=read.csv("mc-donalds-menu.csv") #Leer la base de datos
```

Seleccion de las variables

```
# Instalar y cargar las librerías necesarias
if (!require(MASS)) install.packages("MASS", dependencies=TRUE)
```

```
## Loading required package: MASS
```

```
if (!require(car)) install.packages("car", dependencies=TRUE)
```

```
## Loading required package: car
```

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

```
if (!require(nortest)) install.packages("nortest", dependencies=TRUE)
```

```
## Loading required package: nortest
```

```
library(MASS)          # Para la transformación Box-Cox
library(car)           # Para la transformación Yeo-Johnson
library(e1071)         # Para el análisis de sesgo y curtosis
library(ggplot2)       # Para la visualización de datos
library(nortest)       # Para la prueba de Anderson-Darling
library(tseries)  # Para la prueba de Jarque-Bera
```

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

```
M=read.csv("mc-donalds-menu.csv") #Leer la base de datos
```

```
x <- M$Protein
```

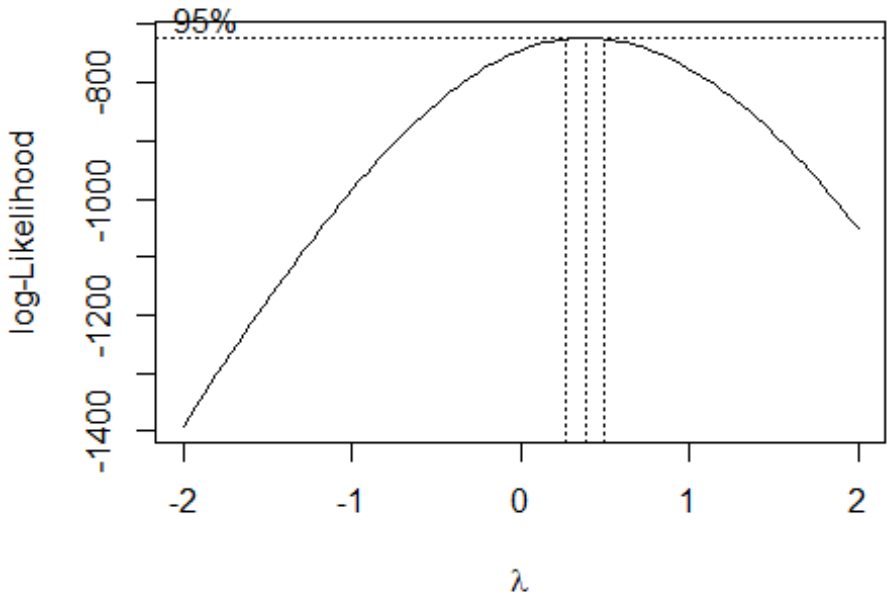
```
# Contar el número de datos
Conteo_datos_proteinas <- length(x)

print(Conteo_datos_proteinas)
```

```
## [1] 260
```

Debido a que contamos con mas de 260 datos podriamos omitir la de Shapiro debido a n>50 pero igualmente observaremos su implicacion

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

```
# Transforma los datos usando Box-Cox
# Añadir 1 a los datos para evitar problemas con ceros
bc <- boxcox((x + 1) ~ 1)
```




```
lambda_exact <- bc$x[which.max(bc$y)]

lambda_exact

## [1] 0.3838384

# Transformación aproximada
x_transf_aprox <- sqrt(x + 1)
x_transf_aprox

##   [1] 4.242641 4.358899 3.872983 4.690416 4.690416 5.196152 4.472136
4.472136
##   [9] 4.582576 4.582576 3.464102 3.464102 4.358899 4.358899 4.358899
4.358899
##  [17] 4.242641 4.242641 5.099020 4.472136 4.582576 3.464102 4.582576
4.690416
##  [25] 5.567764 5.567764 5.830952 5.385165 5.385165 5.196152 5.196152
6.082763
##  [33] 6.082763 6.000000 6.000000 3.000000 4.000000 3.605551 1.414214
```

```
2.645751
##  [41] 2.449490 2.449490 5.000000 5.567764 6.164414 6.164414 5.477226
7.000000
##  [49] 3.605551 4.000000 5.000000 6.324555 4.795832 5.291503 4.795832
4.795832
##  [57] 4.795832 5.000000 5.385165 6.082763 6.403124 5.744563 6.082763
6.082763
##  [65] 6.403124 4.690416 3.872983 4.795832 4.582576 3.872983 5.744563
6.082763
##  [73] 5.291503 5.567764 5.291503 5.656854 4.898979 5.291503 3.162278
3.741657
##  [81] 4.795832 6.708204 9.380832 4.000000 3.162278 5.099020 5.477226
2.645751
##  [89] 4.898979 5.291503 3.872983 4.123106 3.872983 4.123106 4.000000
4.123106
##  [97] 1.732051 2.236068 2.645751 1.414214 1.414214 1.000000 2.236068
1.732051
## [105] 1.732051 1.732051 1.414214 3.000000 2.828427 2.645751 1.000000
1.000000
## [113] 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000
## [121] 1.000000 1.000000 1.732051 2.000000 2.236068 1.414214 1.000000
1.000000
## [129] 1.000000 1.000000 3.000000 3.162278 1.000000 1.732051 2.000000
2.236068
## [137] 1.000000 1.000000 1.000000 1.000000 1.000000 1.414214 1.414214
1.414214
## [145] 1.000000 1.000000 1.000000 1.000000 3.162278 3.464102 4.000000
3.162278
## [153] 3.464102 4.000000 3.162278 3.464102 4.000000 3.162278 3.464102
4.000000
## [161] 3.162278 3.605551 4.000000 3.316625 3.605551 4.123106 3.316625
3.605551
## [169] 4.123106 3.316625 3.605551 4.123106 3.316625 3.605551 4.123106
3.316625
## [177] 3.605551 4.123106 3.316625 3.741657 4.123106 3.464102 3.741657
4.242641
## [185] 3.316625 3.605551 4.123106 3.316625 3.741657 4.242641 3.464102
3.872983
## [193] 4.242641 3.605551 3.872983 4.472136 1.414214 1.414214 1.732051
1.414214
## [201] 1.414214 1.732051 1.414214 1.414214 1.732051 1.414214 1.414214
1.732051
## [209] 1.414214 1.414214 1.732051 3.000000 3.162278 3.872983 3.000000
3.316625
## [217] 3.872983 3.000000 3.162278 3.741657 3.000000 3.316625 3.872983
2.828427
## [225] 3.162278 3.464102 2.828427 3.162278 3.464102 3.000000 3.162278
3.605551
## [233] 1.732051 2.000000 2.236068 2.000000 2.236068 2.449490 1.732051
```

```
2.000000
## [241] 2.236068 3.464102 3.872983 4.358899 3.605551 4.000000 4.358899
3.605551
## [249] 4.000000 4.472136 3.872983 4.358899 3.741657 4.582576 3.162278
3.605551
## [257] 4.000000 3.000000 4.690416 3.316625
```

```
# Transformación exacta
x_transf_exact <- ((x + 1)^lambda_exact - 1) / lambda_exact
x_transf_exact
```

```
##   [1]  5.2955787  5.4612587  4.7615655  5.9281928  5.9281928  6.6260599
##   [7]  5.6216488  5.6216488  5.7771703  5.7771703  4.1568554  4.1568554
##  [13]  5.4612587  5.4612587  5.4612587  5.4612587  5.2955787  5.2955787
##  [19]  6.4932972  5.6216488  5.7771703  4.1568554  5.7771703  5.9281928
##  [25]  7.1287843  7.1287843  7.4801104  6.8827676  6.8827676  6.6260599
##  [31]  6.6260599  7.8128150  7.8128150  7.7038246  7.7038246  3.4499091
##  [37]  4.9463384  4.3678356  0.7941071  2.8930904  2.5771963  2.5771963
##  [43]  6.3573496  7.1287843  7.9200053  7.9200053  7.0070394  8.9988974
##  [49]  4.3678356  4.9463384  6.3573496  8.1292837  6.0750423  6.7558264
##  [55]  6.0750423  6.0750423  6.0750423  6.3573496  6.8827676  7.8128150
##  [61]  8.2315091  7.3652046  7.8128150  7.8128150  8.2315091  5.9281928
##  [67]  4.7615655  6.0750423  5.7771703  4.7615655  7.3652046  7.8128150
##  [73]  6.7558264  7.1287843  6.7558264  7.2481328  6.2180085  6.7558264
##  [79]  3.6998078  4.5690368  6.0750423  8.6257262 11.9231668  4.9463384
##  [85]  3.6998078  6.4932972  7.0070394  2.8930904  6.2180085  6.7558264
##  [91]  4.7615655  5.1241251  4.7615655  5.1241251  4.9463384  5.1241251
##  [97]  1.3665522  2.2269170  2.8930904  0.7941071  0.7941071  0.0000000
## [103]  2.2269170  1.3665522  1.3665522  1.3665522  0.7941071  3.4499091
## [109]  3.1822528  2.8930904  0.0000000  0.0000000  0.0000000  0.0000000
## [115]  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
## [121]  0.0000000  0.0000000  1.3665522  1.8302649  2.2269170  0.7941071
## [127]  0.0000000  0.0000000  0.0000000  0.0000000  3.4499091  3.6998078
## [133]  0.0000000  1.3665522  1.8302649  2.2269170  0.0000000  0.0000000
## [139]  0.0000000  0.0000000  0.0000000  0.7941071  0.7941071  0.7941071
## [145]  0.0000000  0.0000000  0.0000000  0.0000000  3.6998078  4.1568554
## [151]  4.9463384  3.6998078  4.1568554  4.9463384  3.6998078  4.1568554
## [157]  4.9463384  3.6998078  4.1568554  4.9463384  3.6998078  4.3678356
## [163]  4.9463384  3.9347419  4.3678356  5.1241251  3.9347419  4.3678356
## [169]  5.1241251  3.9347419  4.3678356  5.1241251  3.9347419  4.3678356
## [175]  5.1241251  3.9347419  4.3678356  5.1241251  3.9347419  4.5690368
## [181]  5.1241251  4.1568554  4.5690368  5.2955787  3.9347419  4.3678356
## [187]  5.1241251  3.9347419  4.5690368  5.2955787  4.1568554  4.7615655
## [193]  5.2955787  4.3678356  4.7615655  5.6216488  0.7941071  0.7941071
## [199]  1.3665522  0.7941071  0.7941071  1.3665522  0.7941071  0.7941071
## [205]  1.3665522  0.7941071  0.7941071  1.3665522  0.7941071  0.7941071
## [211]  1.3665522  3.4499091  3.6998078  4.7615655  3.4499091  3.9347419
## [217]  4.7615655  3.4499091  3.6998078  4.5690368  3.4499091  3.9347419
## [223]  4.7615655  3.1822528  3.6998078  4.1568554  3.1822528  3.6998078
## [229]  4.1568554  3.4499091  3.6998078  4.3678356  1.3665522  1.8302649
```

```
## [235]  2.2269170  1.8302649  2.2269170  2.5771963  1.3665522  1.8302649
## [241]  2.2269170  4.1568554  4.7615655  5.4612587  4.3678356  4.9463384
## [247]  5.4612587  4.3678356  4.9463384  5.6216488  4.7615655  5.4612587
## [253]  4.5690368  5.7771703  3.6998078  4.3678356  4.9463384  3.4499091
## [259]  5.9281928  3.9347419
```

2. Escribe las ecuaciones de los modelos encontrados.

```
# Escribir las ecuaciones de los modelos en R

cat("Ecuación del modelo exacto Box-Cox: y(lambda) = ((x + 1)^",
lambda_exact, " - 1) / ", lambda_exact, "\n")

## Ecuación del modelo exacto Box-Cox: y(lambda) = ((x + 1)^ 0.3838384  - 1)
/  0.3838384

cat("\n")

cat("Ecuación del modelo aproximado Box-Cox: y(lambda) = sqrt(x + 1)\n")

## Ecuación del modelo aproximado Box-Cox: y(lambda) = sqrt(x + 1)
```

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

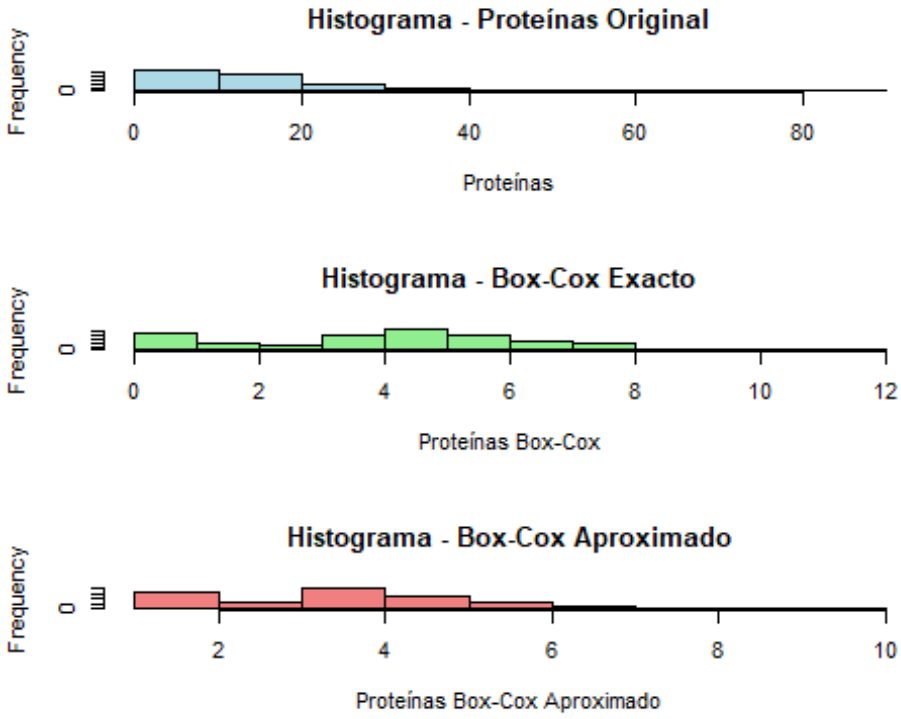
1. Compara las medidas: Mínimo, máximo, media, mediana, cuartil 1 y cuartil 3, sesgo y curtosis.

```
# Cálculo de medidas estadísticas
medidas_original <- c(min(x), max(x), mean(x), median(x), quantile(x, c(0.25,
0.75)), skewness(x), kurtosis(x))
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
)
print(medidas)
```

```
##   Estadísticas  Original BoxCox_Exacto BoxCox_Aproximado
## 1       Mínimo  0.000000     0.0000000         1.0000000
## 2       Máximo 87.000000    11.9231668         9.3808315
## 3        Media 13.338462     4.0205358         3.4609831
## 4      Mediana 12.000000     4.3678356         3.6055513
## 5    Cuartil 1  4.000000     2.2269170         2.2360680
## 6    Cuartil 3 19.000000     5.6216488         4.4721360
## 7        Sesgo  1.561741    -0.1145447         0.1346233
## 8     Curtosis  5.795500    -0.5104494        -0.1941125
```

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

```
# Graficar histogramas
par(mfrow = c(3, 1))
hist(x, main = "Histograma - Proteínas Original", xlab = "Proteínas", col =
"lightblue")
hist(x_transf_exact, main = "Histograma - Box-Cox Exacto", xlab = "Proteínas
Box-Cox", col = "lightgreen")
hist(x_transf_aprox, main = "Histograma - Box-Cox Aproximado", xlab =
"Proteínas Box-Cox Aproximado", col = "lightcoral")
```



Podriamos quitar los alimentos que tengan no solo 0 proteinas porque puede tratarse de un carb, sino tambien 0 calorias, seria algo dificil evaluar las bebidas con calorias dentro de si y distinguirlo de los alimentos

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

```
# Pruebas de normalidad
anderson_test_original <- ad.test(x)
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 = 8.515383e-
12
```

```
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 = 1.831193e-08
```

```
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 = 3.21759e-06
```

```
# Prueba de Jarque-Bera
jarque_bera_original <- jarque.bera.test(x)
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 = 0.2030226
```

```
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 = 0.5722153
```

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

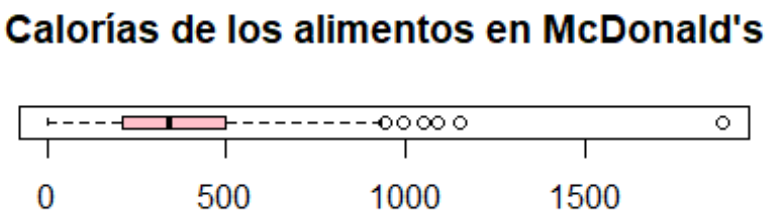
```r
# Detectar outliers utilizando IQR
IQR_x <- IQR(x)
limite_inferior <- quantile(x, 0.25) - 1.5 * IQR_x
limite_superior <- quantile(x, 0.75) + 1.5 * IQR_x

# Identificar outliers
outliers <- x[x < limite_inferior | x > limite_superior]
cat("Outliers detectados:\n")

## Outliers detectados:

print(outliers)

## [1] 48 44 87
```

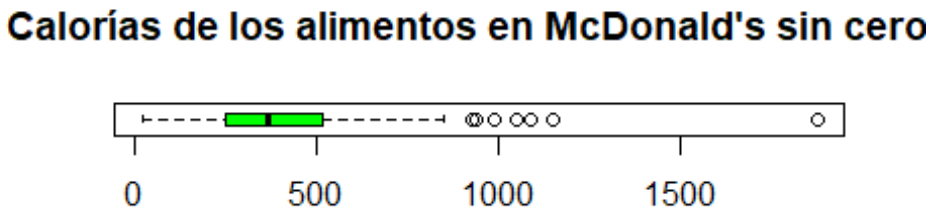
La realidad es que quitar estos valores, no seria conveniente ya que ocmo explicamos en la clase poasada, si guardan sentido, sin embargi aunque no aparezcan quitar los alimentos no caloricos aunque hablemos de proteinas, si es relevante ya que un alimento podria no tener proteinas y si calorias y es valido, pero si no tiene calorias, seria una bebida con edulcorante no propia de la marca directamente

Por lo tanto quitamos esos

Comparativa con Calorias

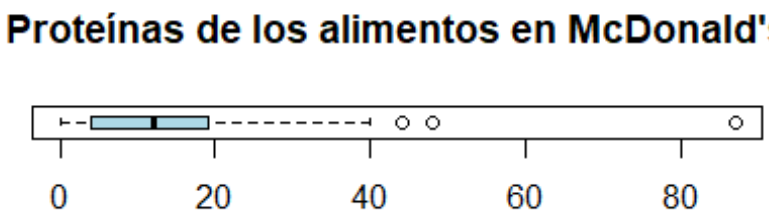
```r
# Crear la variable 'cal' que hace referencia a la columna 'Calories' de M
cal <- M$Calories

# Quitar los alimentos que tienen 0 tanto en calorías como en proteínas por si acaso
M2 <- subset(M, !(cal == 0 | x == 0))

# Graficar boxplots para calorías y proteínas antes y después de quitar los 0 en ambas columnas
par(mfrow = c(2, 1))  # Para mostrar cuatro gráficos (2x2)

# Boxplot de calorías original
boxplot(cal, horizontal = TRUE, col = "pink", main = "Calorías de los alimentos en McDonald's")

# Boxplot de calorías después de eliminar los 0 en ambas columnas
boxplot(M2$Calories, horizontal = TRUE, col = "green", main = "Calorías de los alimentos en McDonald's sin ceros")
```

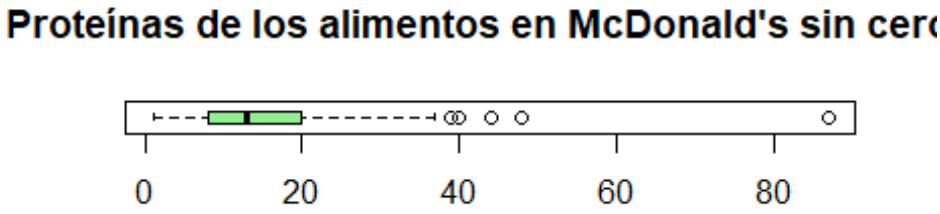




Comparativa con proteinas

```r
par(mfrow = c(2, 1))

# Boxplot de proteínas original
boxplot(x, horizontal = TRUE, col = "lightblue", main = "Proteínas de los alimentos en McDonald's")

# Boxplot de proteínas después de eliminar los 0 en ambas columnas
boxplot(M2$Protein, horizontal = TRUE, col = "lightgreen", main = "Proteínas de los alimentos en McDonald's sin ceros")
```





5. 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).

```
x2 = M2$Protein
```

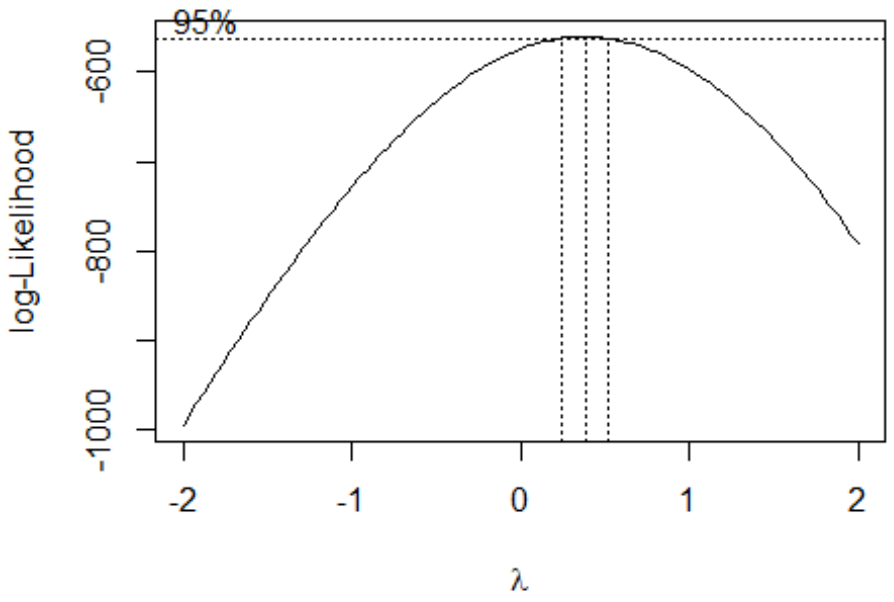
Normalmente en Calorias ya no habria habria valores 0, sin embargo aca si

```
M2$Calories
```

```
##   [1]  300  250  370  450  400  430  460  520  410  470  430  480  510
570  460
##  [16]  520  410  470  540  460  400  420  550  500  620  570  670  740
800  640
##  [31]  690 1090 1150  990 1050  350  520  300  150  460  290  260  530
520  600
##  [46]  610  540  750  240  290  430  720  380  440  430  430  500  510
350  670
##  [61]  510  610  450  750  590  430  360  480  430  360  630  480  610
450  670
##  [76]  520  540  380  190  280  470  940 1880  390  140  380  220  140
450  290
##  [91]  340  260  330  250  360  280  230  340  510  110   20  150  250
160  150
## [106]   45  330  340  280  100  130  150  190  280  150  180  220  170
```

```
210  280
## [121]  270  340  430  270  330  430  260  330  420  210  260  330  100
130  170
## [136]  200  250  310  200  250  310  190  240  300  140  170  220  340
410  500
## [151]  270  330  390  320  390  480  250  310  370  360  440  540  280
340  400
## [166]  140  190  270  130  180  260  130  180  250  120  170  240   80
120  160
## [181]  290  350  480  240  290  390  280  340  460  230  270  370  450
550  670
## [196]  450  550  670  530  630  760  220  260  340  210  250  330  210
260  340
## [211]  530  660  820  550  690  850  560  700  850  660  820  650  930
430  510
## [226]  690  340  810  410
```

```
bc <- boxcox((x2+1) ~ 1)
```




```
lambda_exact2 <- bc$x[which.max(bc$y)]

lambda_exact2

## [1] 0.3838384

library(VGAM)

## Loading required package: stats4
```

```
## Loading required package: splines

##
## Attaching package: 'VGAM'

## The following object is masked from 'package:car':
##
##     logit
```

```r
# Aplicar la transformación de Yeo-Johnson
x_yeo <- yeo.johnson(x2, lambda = lambda_exact2)

# Aplicar una transformación aproximada para comparación
x_aprox <- sqrt(x2 + 1)

x_yeo
```

```
##   [1]  5.2955787  5.4612587  4.7615655  5.9281928  5.9281928  6.6260599
##   [7]  5.6216488  5.6216488  5.7771703  5.7771703  4.1568554  4.1568554
##  [13]  5.4612587  5.4612587  5.4612587  5.4612587  5.2955787  5.2955787
##  [19]  6.4932972  5.6216488  5.7771703  4.1568554  5.7771703  5.9281928
##  [25]  7.1287843  7.1287843  7.4801104  6.8827676  6.8827676  6.6260599
##  [31]  6.6260599  7.8128150  7.8128150  7.7038246  7.7038246  3.4499091
##  [37]  4.9463384  4.3678356  0.7941071  2.8930904  2.5771963  2.5771963
##  [43]  6.3573496  7.1287843  7.9200053  7.9200053  7.0070394  8.9988974
##  [49]  4.3678356  4.9463384  6.3573496  8.1292837  6.0750423  6.7558264
##  [55]  6.0750423  6.0750423  6.0750423  6.3573496  6.8827676  7.8128150
##  [61]  8.2315091  7.3652046  7.8128150  7.8128150  8.2315091  5.9281928
##  [67]  4.7615655  6.0750423  5.7771703  4.7615655  7.3652046  7.8128150
##  [73]  6.7558264  7.1287843  6.7558264  7.2481328  6.2180085  6.7558264
##  [79]  3.6998078  4.5690368  6.0750423  8.6257262 11.9231668  4.9463384
##  [85]  3.6998078  6.4932972  7.0070394  2.8930904  6.2180085  6.7558264
##  [91]  4.7615655  5.1241251  4.7615655  5.1241251  4.9463384  5.1241251
##  [97]  1.3665522  2.2269170  2.8930904  0.7941071  0.7941071  2.2269170
## [103]  1.3665522  1.3665522  1.3665522  0.7941071  3.4499091  3.1822528
## [109]  2.8930904  3.4499091  3.6998078  1.3665522  1.8302649  2.2269170
## [115]  0.7941071  0.7941071  0.7941071  3.6998078  4.1568554  4.9463384
## [121]  3.6998078  4.1568554  4.9463384  3.6998078  4.1568554  4.9463384
## [127]  3.6998078  4.1568554  4.9463384  3.6998078  4.3678356  4.9463384
## [133]  3.9347419  4.3678356  5.1241251  3.9347419  4.3678356  5.1241251
## [139]  3.9347419  4.3678356  5.1241251  3.9347419  4.3678356  5.1241251
## [145]  3.9347419  4.3678356  5.1241251  3.9347419  4.5690368  5.1241251
## [151]  4.1568554  4.5690368  5.2955787  3.9347419  4.3678356  5.1241251
## [157]  3.9347419  4.5690368  5.2955787  4.1568554  4.7615655  5.2955787
## [163]  4.3678356  4.7615655  5.6216488  0.7941071  0.7941071  1.3665522
## [169]  0.7941071  0.7941071  1.3665522  0.7941071  0.7941071  1.3665522
## [175]  0.7941071  0.7941071  1.3665522  0.7941071  0.7941071  1.3665522
## [181]  3.4499091  3.6998078  4.7615655  3.4499091  3.9347419  4.7615655
## [187]  3.4499091  3.6998078  4.5690368  3.4499091  3.9347419  4.7615655
## [193]  3.1822528  3.6998078  4.1568554  3.1822528  3.6998078  4.1568554
## [199]  3.4499091  3.6998078  4.3678356  1.3665522  1.8302649  2.2269170
```

```
## [205]  1.8302649  2.2269170  2.5771963  1.3665522  1.8302649  2.2269170
## [211]  4.1568554  4.7615655  5.4612587  4.3678356  4.9463384  5.4612587
## [217]  4.3678356  4.9463384  5.6216488  4.7615655  5.4612587  4.5690368
## [223]  5.7771703  3.6998078  4.3678356  4.9463384  3.4499091  5.9281928
## [229]  3.9347419

x_aprox

##   [1] 4.242641 4.358899 3.872983 4.690416 4.690416 5.196152 4.472136
4.472136
##   [9] 4.582576 4.582576 3.464102 3.464102 4.358899 4.358899 4.358899
4.358899
##  [17] 4.242641 4.242641 5.099020 4.472136 4.582576 3.464102 4.582576
4.690416
##  [25] 5.567764 5.567764 5.830952 5.385165 5.385165 5.196152 5.196152
6.082763
##  [33] 6.082763 6.000000 6.000000 3.000000 4.000000 3.605551 1.414214
2.645751
##  [41] 2.449490 2.449490 5.000000 5.567764 6.164414 6.164414 5.477226
7.000000
##  [49] 3.605551 4.000000 5.000000 6.324555 4.795832 5.291503 4.795832
4.795832
##  [57] 4.795832 5.000000 5.385165 6.082763 6.403124 5.744563 6.082763
6.082763
##  [65] 6.403124 4.690416 3.872983 4.795832 4.582576 3.872983 5.744563
6.082763
##  [73] 5.291503 5.567764 5.291503 5.656854 4.898979 5.291503 3.162278
3.741657
##  [81] 4.795832 6.708204 9.380832 4.000000 3.162278 5.099020 5.477226
2.645751
##  [89] 4.898979 5.291503 3.872983 4.123106 3.872983 4.123106 4.000000
4.123106
##  [97] 1.732051 2.236068 2.645751 1.414214 1.414214 2.236068 1.732051
1.732051
## [105] 1.732051 1.414214 3.000000 2.828427 2.645751 3.000000 3.162278
1.732051
## [113] 2.000000 2.236068 1.414214 1.414214 1.414214 3.162278 3.464102
4.000000
## [121] 3.162278 3.464102 4.000000 3.162278 3.464102 4.000000 3.162278
3.464102
## [129] 4.000000 3.162278 3.605551 4.000000 3.316625 3.605551 4.123106
3.316625
## [137] 3.605551 4.123106 3.316625 3.605551 4.123106 3.316625 3.605551
4.123106
## [145] 3.316625 3.605551 4.123106 3.316625 3.741657 4.123106 3.464102
3.741657
## [153] 4.242641 3.316625 3.605551 4.123106 3.316625 3.741657 4.242641
3.464102
## [161] 3.872983 4.242641 3.605551 3.872983 4.472136 1.414214 1.414214
1.732051
```

```
## [169] 1.414214 1.414214 1.732051 1.414214 1.414214 1.732051 1.414214
1.414214
## [177] 1.732051 1.414214 1.414214 1.732051 3.000000 3.162278 3.872983
3.000000
## [185] 3.316625 3.872983 3.000000 3.162278 3.741657 3.000000 3.316625
3.872983
## [193] 2.828427 3.162278 3.464102 2.828427 3.162278 3.464102 3.000000
3.162278
## [201] 3.605551 1.732051 2.000000 2.236068 2.000000 2.236068 2.449490
1.732051
## [209] 2.000000 2.236068 3.464102 3.872983 4.358899 3.605551 4.000000
4.358899
## [217] 3.605551 4.000000 4.472136 3.872983 4.358899 3.741657 4.582576
3.162278
## [225] 3.605551 4.000000 3.000000 4.690416 3.316625
```

6. Escribe la ecuación del modelo encontrado.

```
# Escribir las ecuaciones de los modelos en R

cat("Ecuación del modelo exacto Box-Cox: y(lambda) = ((x + 1)^",lambda_exact2
, " - 1) / ", lambda_exact2 , "\n")

## Ecuación del modelo exacto Box-Cox: y(lambda) = ((x + 1)^ 0.3838384  - 1)
/  0.3838384

cat("\n")

cat("Ecuación del modelo aproximado Box-Cox: y(lambda) = sqrt(x + 1)\n")

## Ecuación del modelo aproximado Box-Cox: y(lambda) = sqrt(x + 1)
```

7. 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.

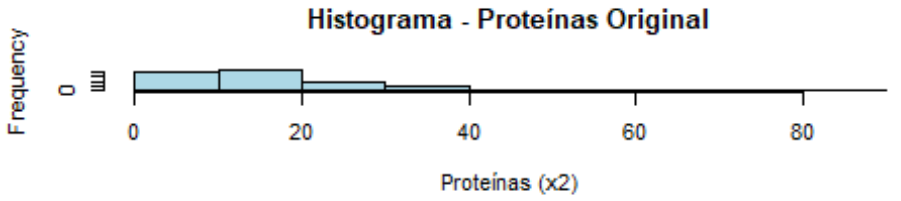
```
# Cálculo de medidas estadísticas
medidas_original <- c(min(x2), max(x2), mean(x2), median(x2), quantile(x2,
c(0.25, 0.75)), skewness(x2), kurtosis(x2))
medidas_exacta <- c(min(x_yeo), max(x_yeo), mean(x_yeo), median(x_yeo),
quantile(x_yeo, c(0.25, 0.75)), skewness(x_yeo), kurtosis(x_yeo))
medidas_aproximada <- c(min(x_aprox), max(x_aprox), mean(x_aprox),
median(x_aprox), quantile(x_aprox, c(0.25, 0.75)), skewness(x_aprox),
kurtosis(x_aprox))

# Crear un DataFrame para comparar las medidas
medidas <- data.frame(
```

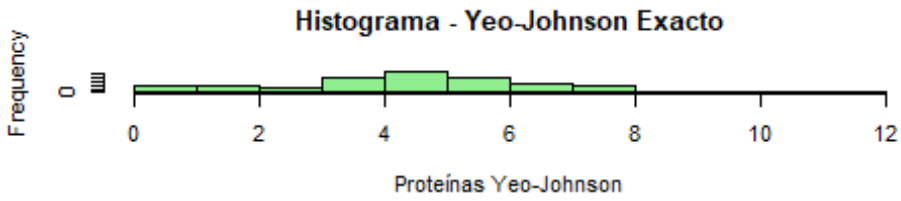
```
  Estadísticas = c("Mínimo", "Máximo", "Media", "Mediana", "Cuartil 1",
"Cuartil 3", "Sesgo", "Curtosis"),
  Original = medidas_original,
  YeoJohnson_Exacta = medidas_exacta,
  YeoJohnson_Aproximada = medidas_aproximada
)

# Mostrar el DataFrame
print(medidas)
```

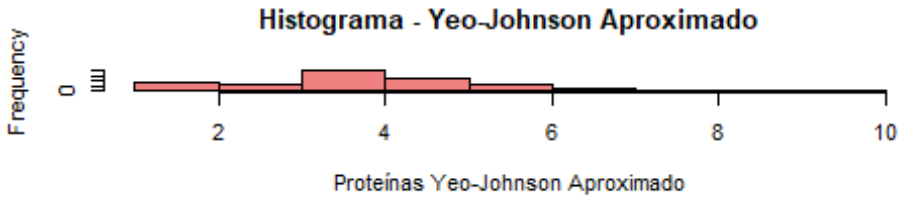
```
##   Estadísticas  Original YeoJohnson_Exacta YeoJohnson_Aproximada
## 1       Mínimo  1.000000        0.79410706             1.4142136
## 2       Máximo 87.000000       11.92316681             9.3808315
## 3        Media 15.100437        4.53764830             3.7793593
## 4      Mediana 13.000000        4.56903680             3.7416574
## 5    Cuartil 1  8.000000        3.44990907             3.0000000
## 6    Cuartil 3 20.000000        5.77717031             4.5825757
## 7        Sesgo  1.706122       -0.01171615             0.2550548
## 8     Curtosis  6.920046        0.05695929             0.4519455
```

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

```
# Graficar histogramas
par(mfrow = c(3, 1))  # Configurar la disposición de los gráficos en 3 filas
y 1 columna

# Histograma de la variable original
hist(x2, main = "Histograma - Proteínas Original", xlab = "Proteínas (x2)",
col = "lightblue")

# Histograma de la transformación exacta de Yeo-Johnson
hist(x_yeo, main = "Histograma - Yeo-Johnson Exacto", xlab = "Proteínas Yeo-
Johnson", col = "lightgreen")

# Histograma de la transformación aproximada de Yeo-Johnson
hist(x_aprox, main = "Histograma - Yeo-Johnson Aproximado", xlab = "Proteínas
Yeo-Johnson Aproximado", col = "lightcoral")
```







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

```
# Cargar las librerías necesarias
library(nortest)   # Para la prueba de Anderson-Darling
library(tseries)   # Para la prueba de Jarque-Bera

# Pruebas de normalidad con Anderson-Darling
anderson_test_original <- ad.test(x2)
anderson_test_yeo <- ad.test(x_yeo)
anderson_test_yeo_aprox <- ad.test(x_aprox)

# Mostrar los resultados de Anderson-Darling
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 = 2.644597e-
10
```

```
cat("Prueba de Anderson-Darling para Yeo-Johnson exacto: p-value =",
anderson_test_yeo$p.value, "\n")
```

```
## Prueba de Anderson-Darling para Yeo-Johnson exacto: p-value = 0.0003279796
```

```
cat("Prueba de Anderson-Darling para Yeo-Johnson aproximado: p-value =",
anderson_test_yeo_aprox$p.value, "\n")
```

```
## Prueba de Anderson-Darling para Yeo-Johnson aproximado: p-value =
0.002843782
```

Mejoramos un poco en el incremento de p value aunque sigue siendo sumamente bajo para obtener normalidad, sin embargo en esta trasnformacion Yeo Johnson obtuvo mejores resultados

```
# Pruebas de normalidad con Jarque-Bera
jarque_bera_original <- jarque.bera.test(x2)
jarque_bera_yeo <- jarque.bera.test(x_yeo)
jarque_bera_yeo_aprox <- jarque.bera.test(x_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 Yeo-Johnson exacto: p-value =",
jarque_bera_yeo$p.value, "\n")

## Prueba de Jarque-Bera para Yeo-Johnson exacto: p-value = 0.964463

cat("Prueba de Jarque-Bera para Yeo-Johnson aproximado: p-value =",
jarque_bera_yeo_aprox$p.value, "\n")

## Prueba de Jarque-Bera para Yeo-Johnson aproximado: p-value = 0.0937094
```

8. 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 valores p de la prueba de Anderson-Darling para los datos originales son muy bajos, lo que indica una fuerte evidencia contra la normalidad.

Después de la transformación exacta de Yeo-Johnson, el valor p mejora, pero sigue siendo bajo (0.0003279796), lo que indica que aunque hay una mejora, los datos aún no son perfectamente normales.

La transformación aproximada de Yeo-Johnson también mejora la normalidad, pero el valor p sigue siendo bajo en comparación con la transformación exacta.

9. Concluye sobre las ventajas y desventajes de los modelos de Box Cox y de Yeo Johnson.

Ventajas de Yeo-Johnson sobre Box-Cox:

Flexibilidad: Yeo-Johnson puede manejar datos con valores negativos y ceros, lo que lo hace aplicable en más situaciones.

Mejor Normalización en Algunos Casos: Como se observó en los resultados, la transformación exacta de Yeo-Johnson logró un mejor ajuste a la normalidad que las transformaciones de Box-Cox en este caso particular.

Desventajas de Yeo-Johnson en Comparación con Box-Cox:

Complejidad: La transformación de Yeo-Johnson es más compleja computacionalmente y puede requerir más ajustes.

Menor Eficiencia en Datos Positivos: Box-Cox sigue siendo más eficiente cuando los datos son todos positivos y no hay ceros.

10. Analiza las diferencias entre la transformación y el escalamiento de los datos:

1. Escribe al menos 3 diferencias entre lo que es la transformación y el escalamiento de los datos

La Transformación m odifica la forma de la distribución de los datos. Se utiliza para hacer que los datos se ajusten mejor a una distribución específica, como la normal, o para corregir la asimetría de los datos. Mientras que el escalamiento Cambia la escala de los datos, pero no altera la forma de la distribución. Se utiliza para poner los datos en un rango común, como [0, 1] o para estandarizarlos con media 0 y desviación estándar 1.

La Transformación Puede cambiar la estructura de los datos, ajustando sesgo, curtosis y otros aspectos de la distribución, sin embargo, el Escalamiento Mantiene la estructura original de los datos pero ajusta su magnitud, es decir, la relación entre los valores sigue siendo la misma.

La Transformación es común en análisis estadísticos donde la normalidad de los datos es un supuesto importante, como en ANOVA, regresiones lineales o modelos paramétricos.Por otro lado el escalamiento es crucial en algoritmos de machine learning, como SVM, k-means o redes neuronales, donde las diferencias en la escala de las características pueden influir en los resultados.

2. Indica cuándo es necesario utilizar cada uno

Transformación: Es necesaria cuando los datos no siguen la distribución requerida por un análisis estadístico, como la normalidad en regresiones, ANOVA, o pruebas paramétricas. Tambien para corregir asimetrías en la distribución o para estabilizar la varianza.

Escalamiento: Es necesario cuando se aplican algoritmos de machine learning que son sensibles a la escala de los datos, como SVM, k-means o redes neuronales. Tambien para igualar las escalas de diferentes características, especialmente cuando las variables tienen diferentes unidades de medida.