3. Transformaciones, Blanquita

Francisco Castorena, A00827756

2023-08-21

```
df = read.csv('mc-donalds-menu-1.csv')
```

Primero seleccionaremos una variable a transformar para que tenga una distribución cercana a la normal, sin que esta sea muy similar, a continuación se ven los diferentes histogramas.

```
# load the package dplyr
library("dplyr")
numeric_df <- select_if(df, is.numeric)
```

```
#adjust plot margins
par(mar = c(1,1,1,1))
library(Hmisc)
```

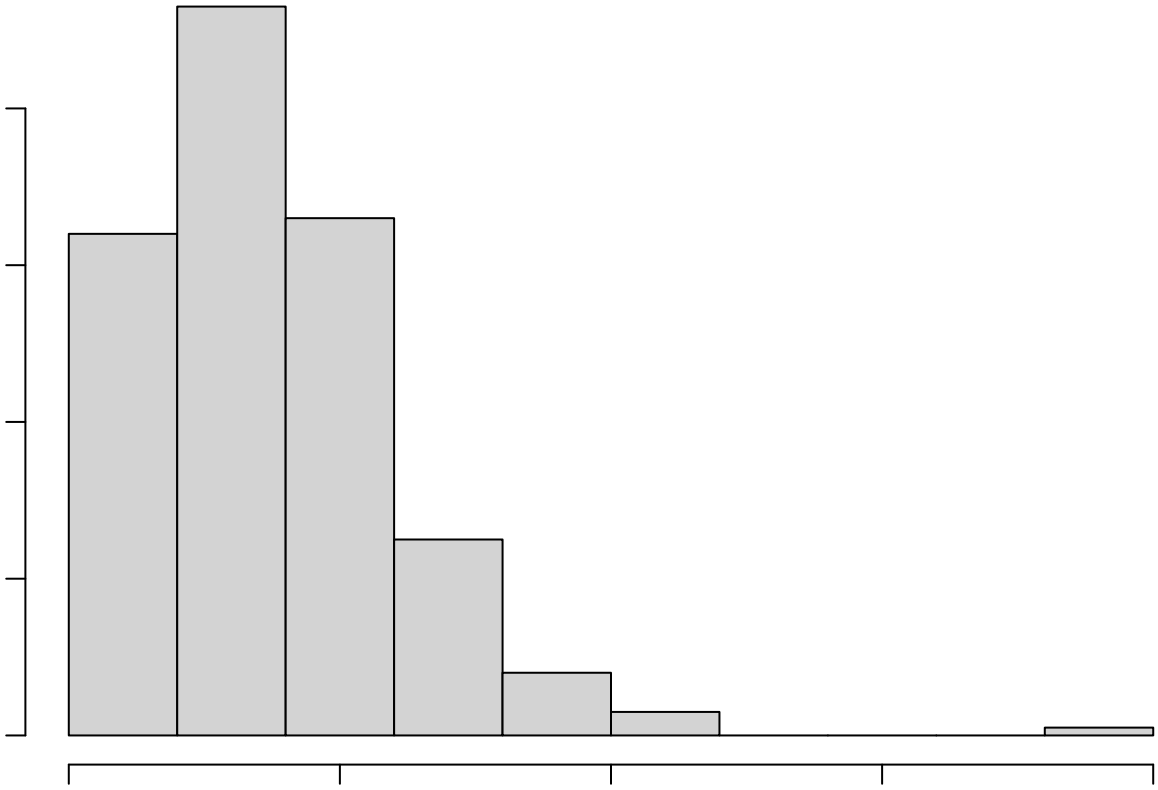
```
## Warning: package 'Hmisc' was built under R version 4.2.3
```

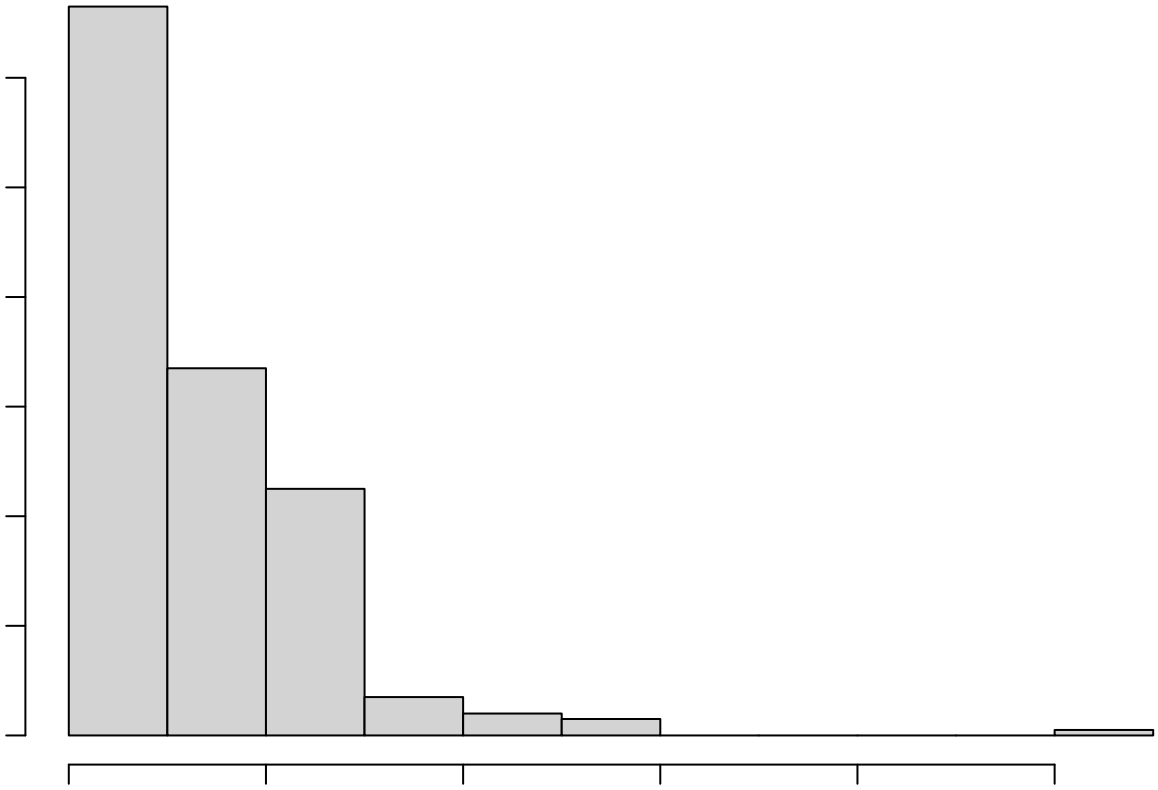
```
##
## Attaching package: 'Hmisc'
```

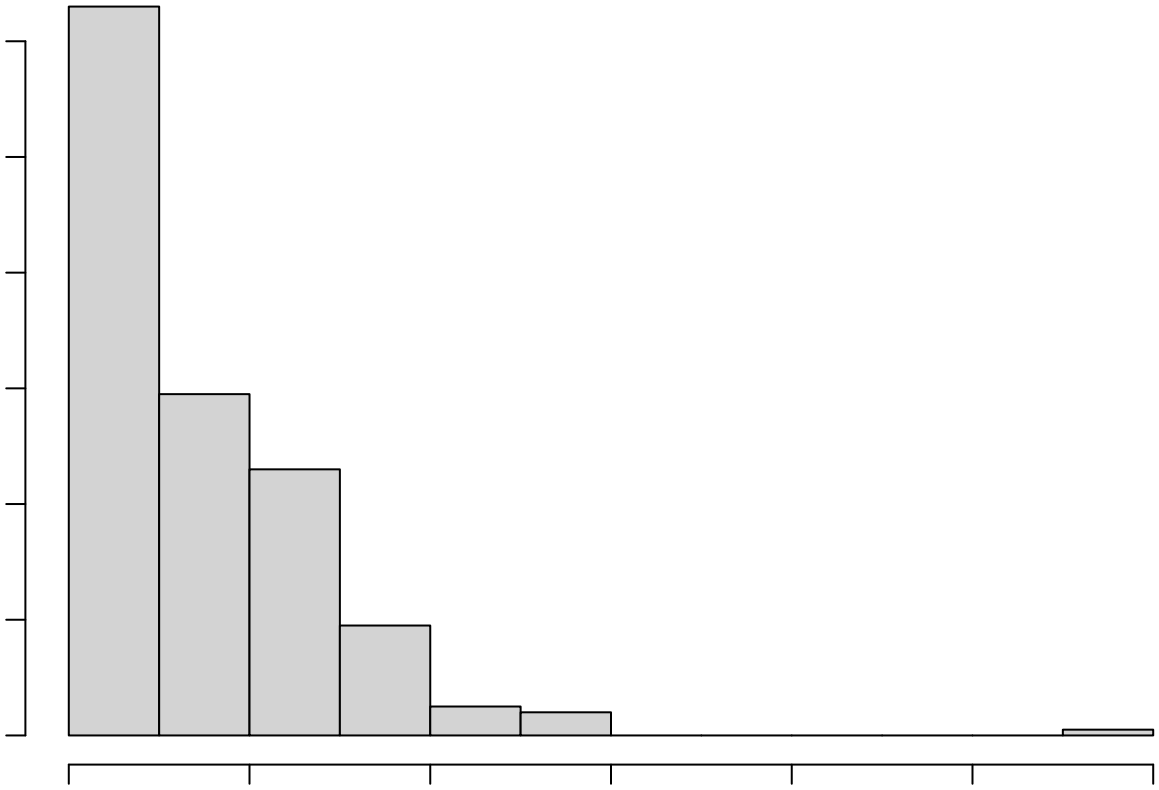
```
## The following objects are masked from 'package:dplyr':
##
##     src, summarize
```

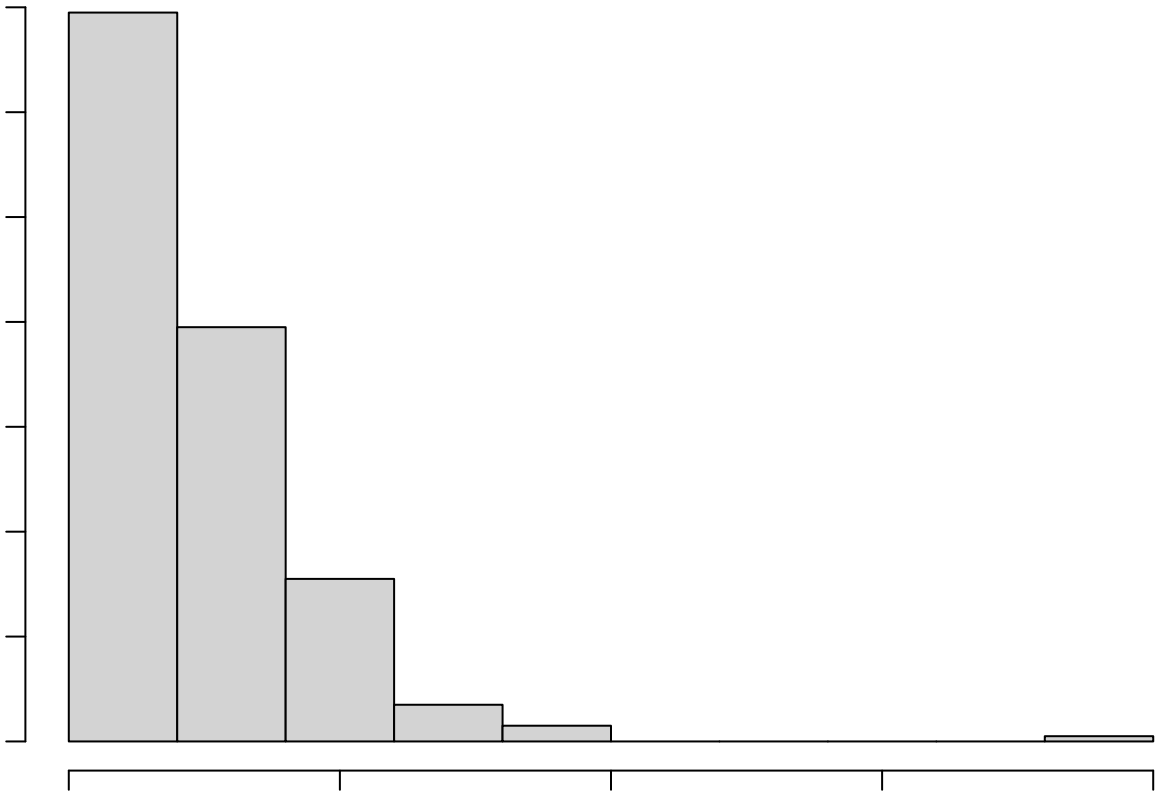
```
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

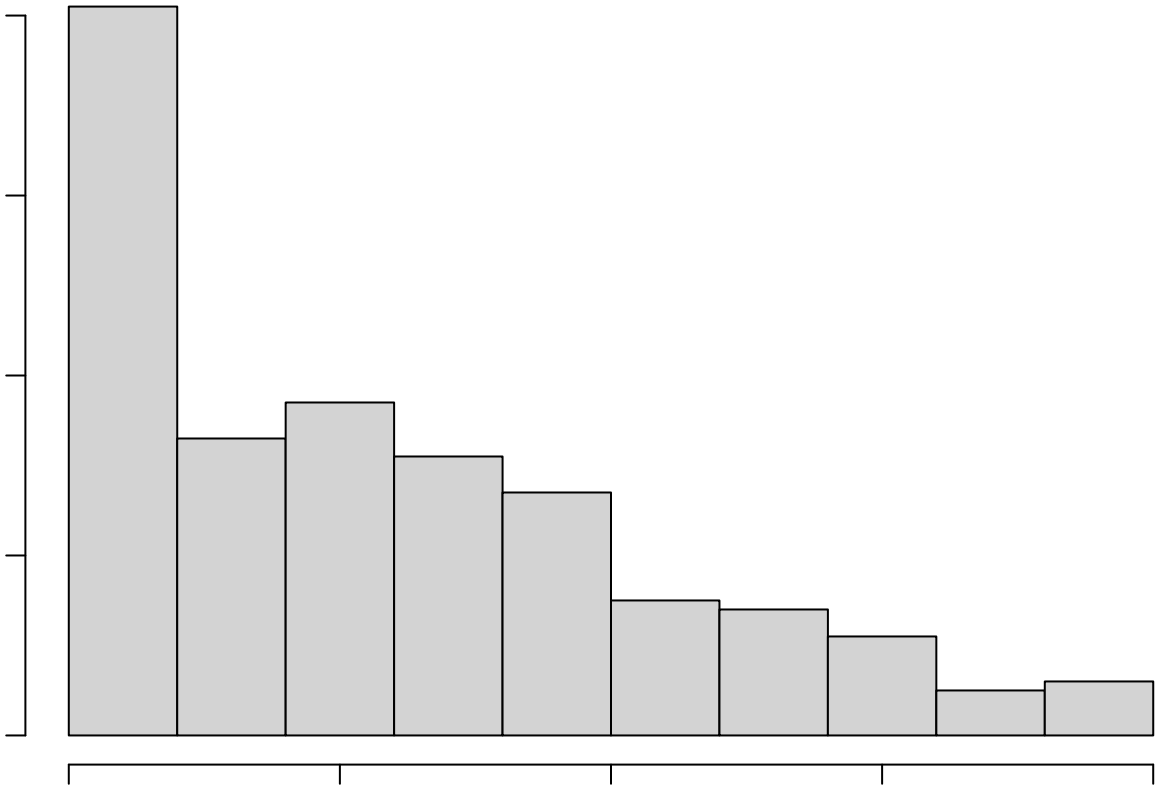
```
for (col_name in names(numeric_df)) {
  hist(numeric_df[[col_name]], main = col_name)
}
```

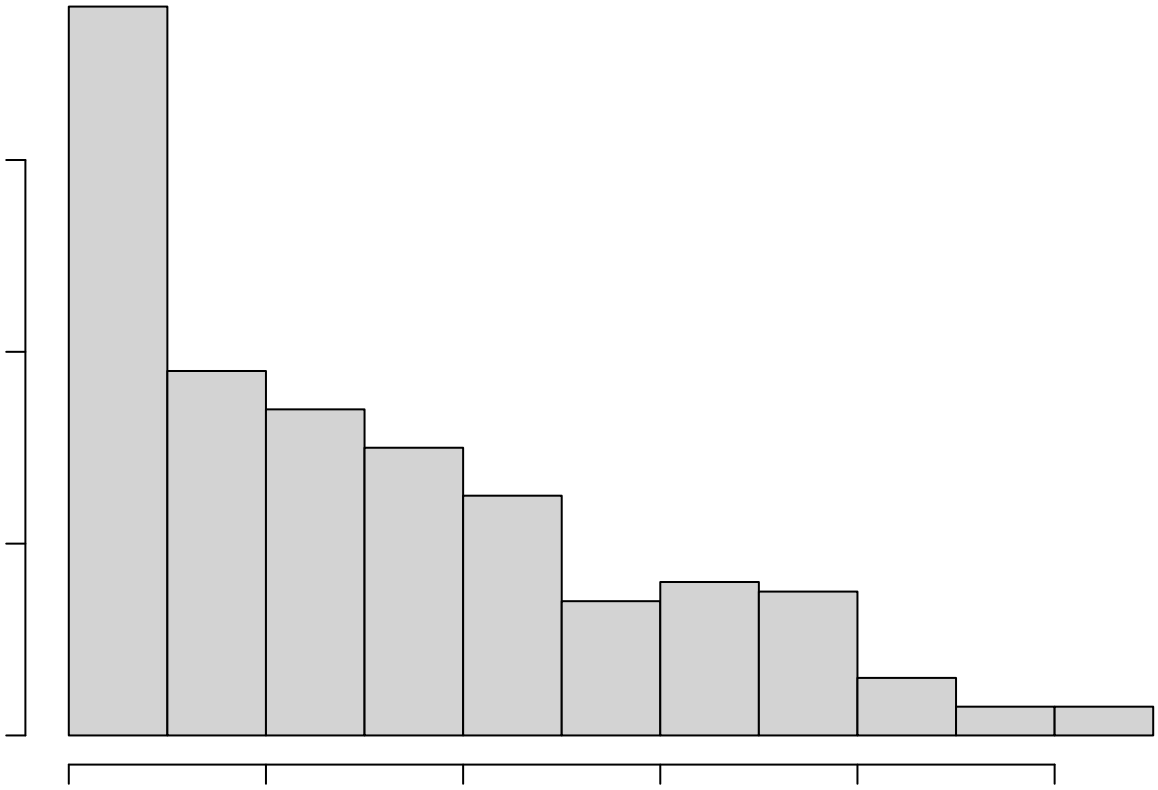

Calories

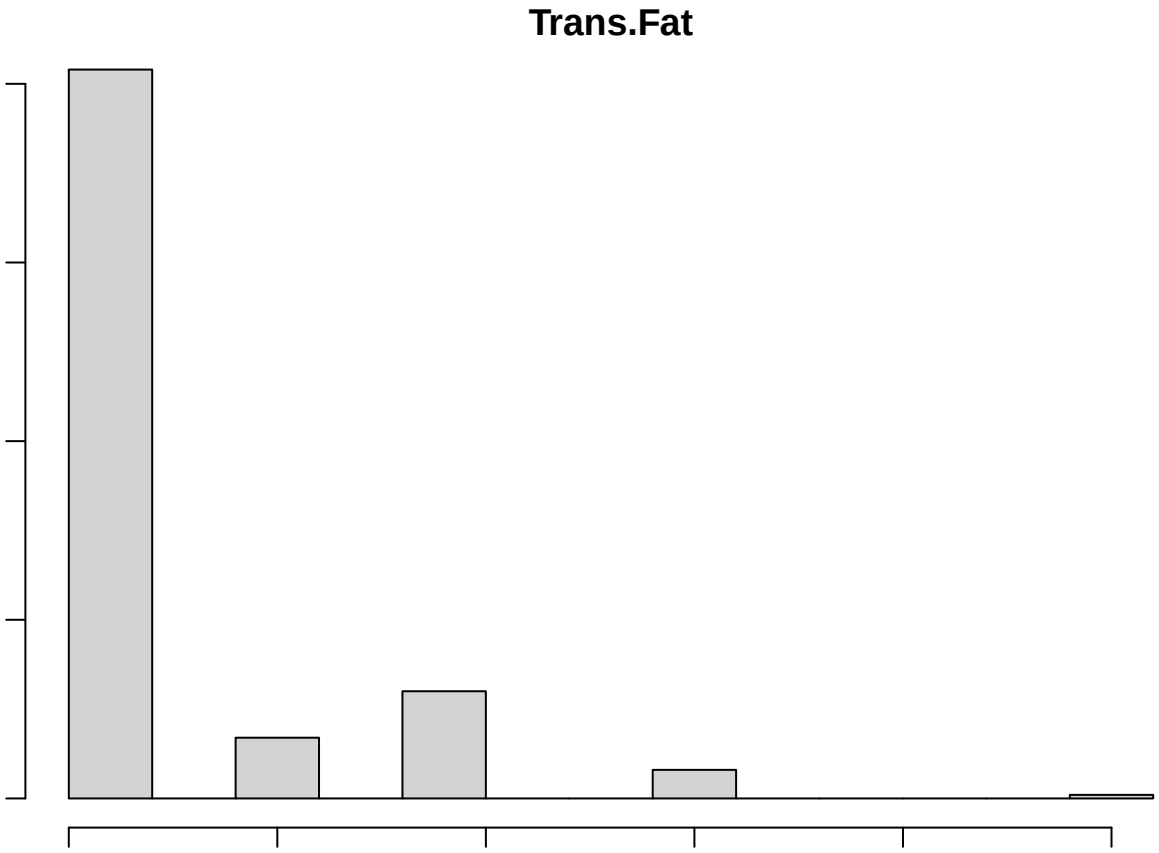

Calories.from.Fat

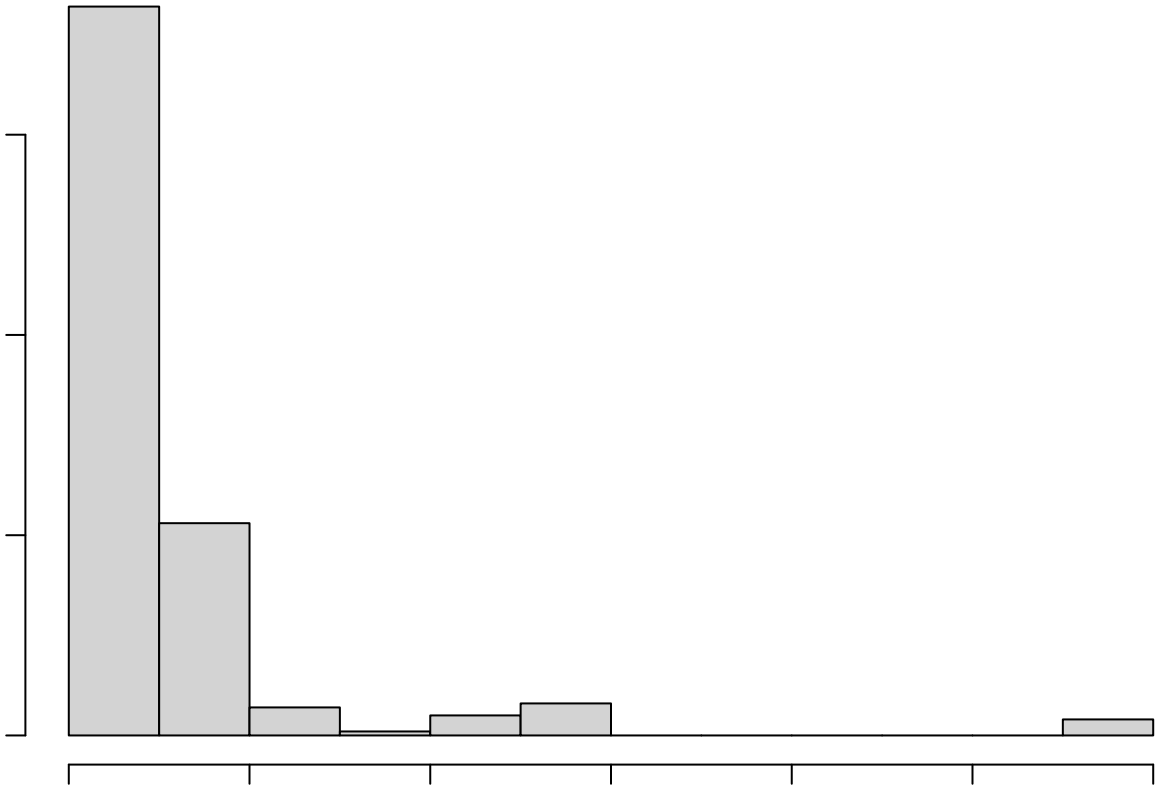

Total.Fat

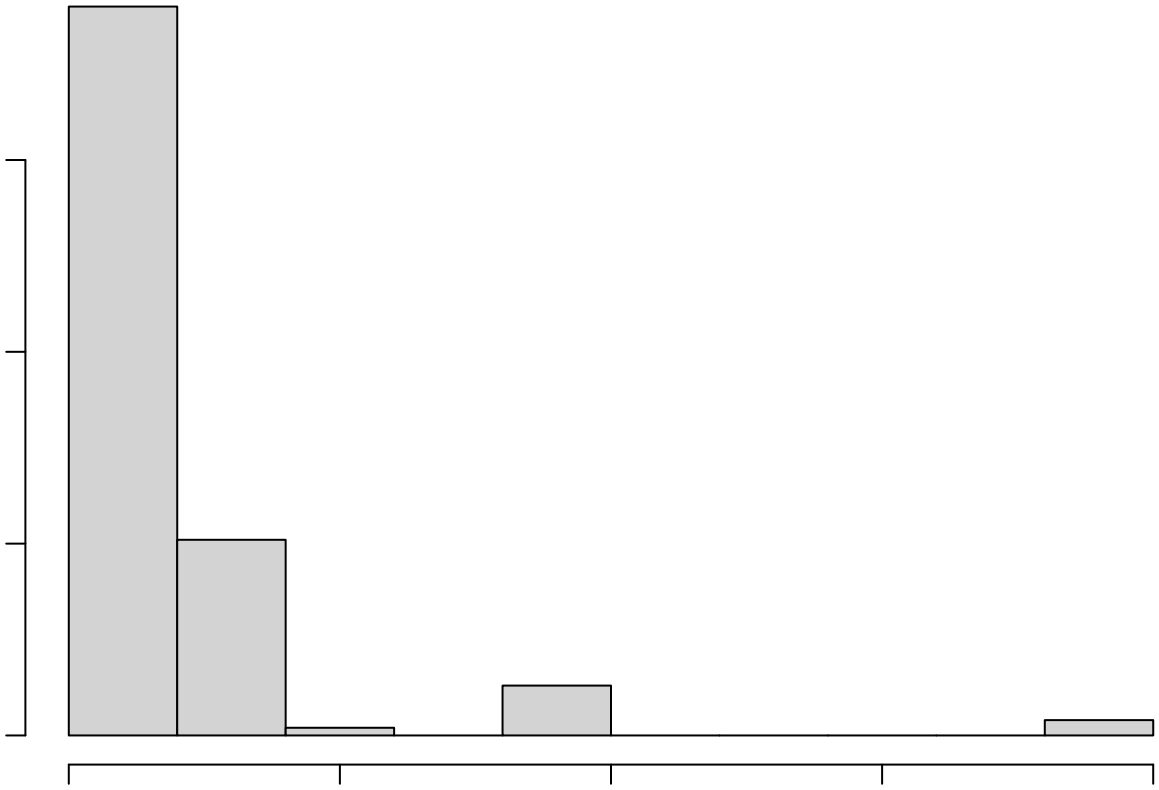

Total.Fat....Daily.Value.

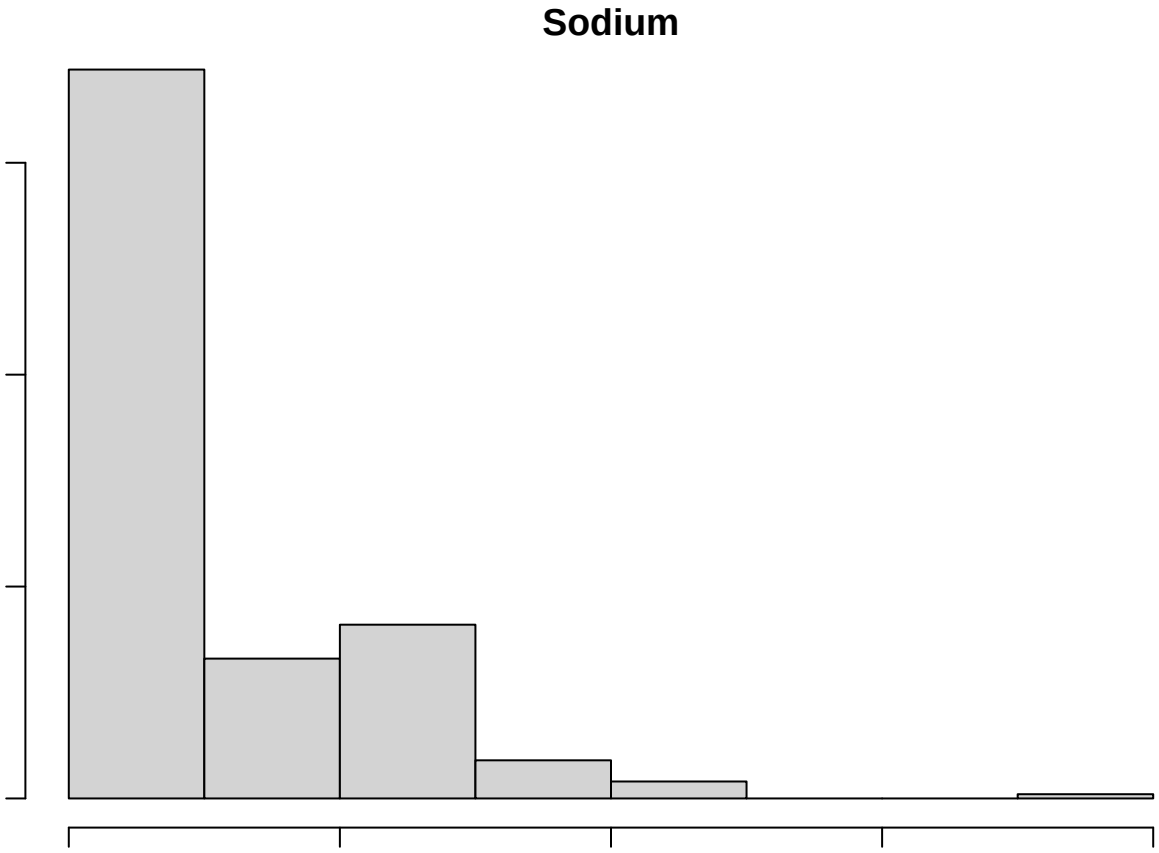

Saturated.Fat

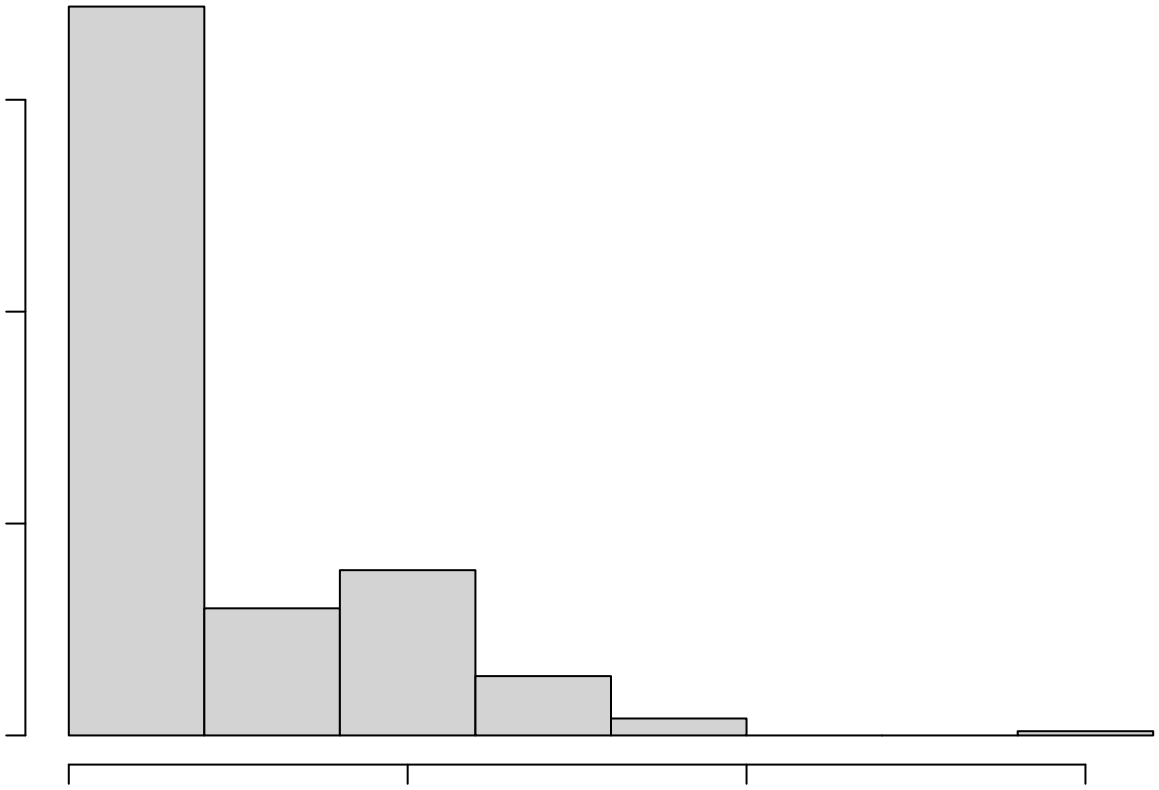

Saturated.Fat....Daily.Value.

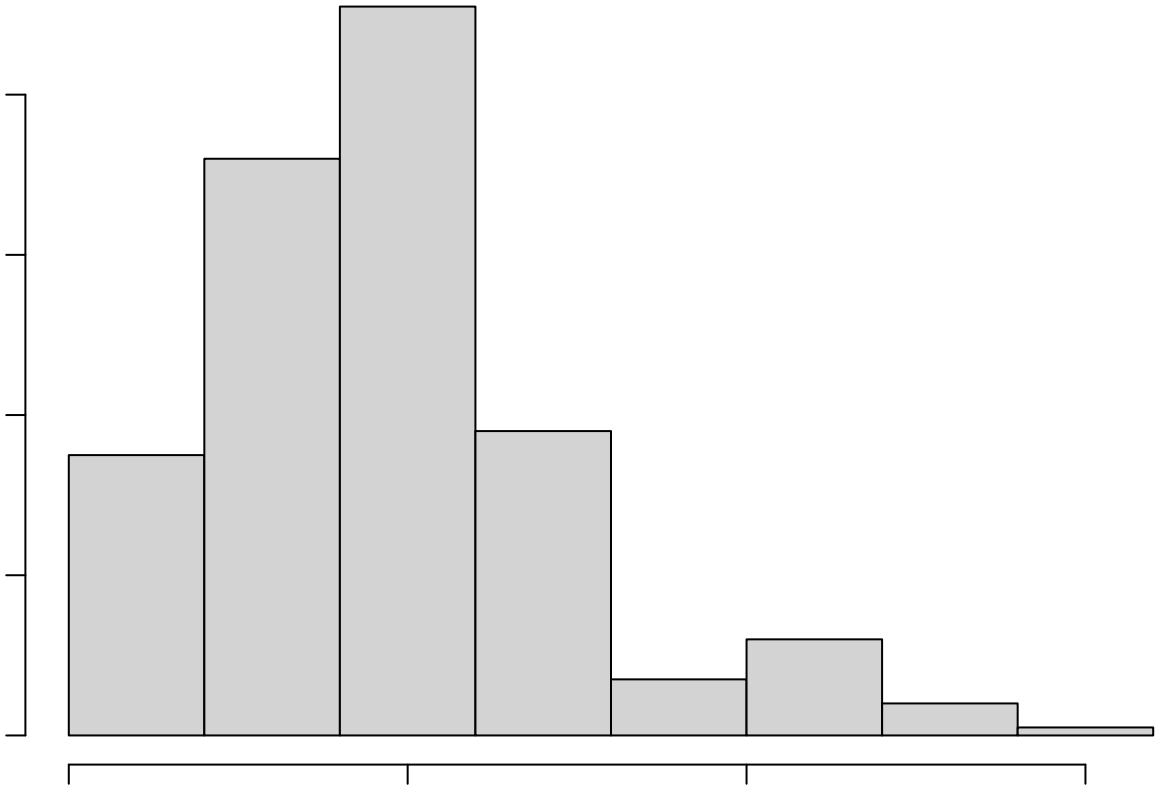

Trans.Fat

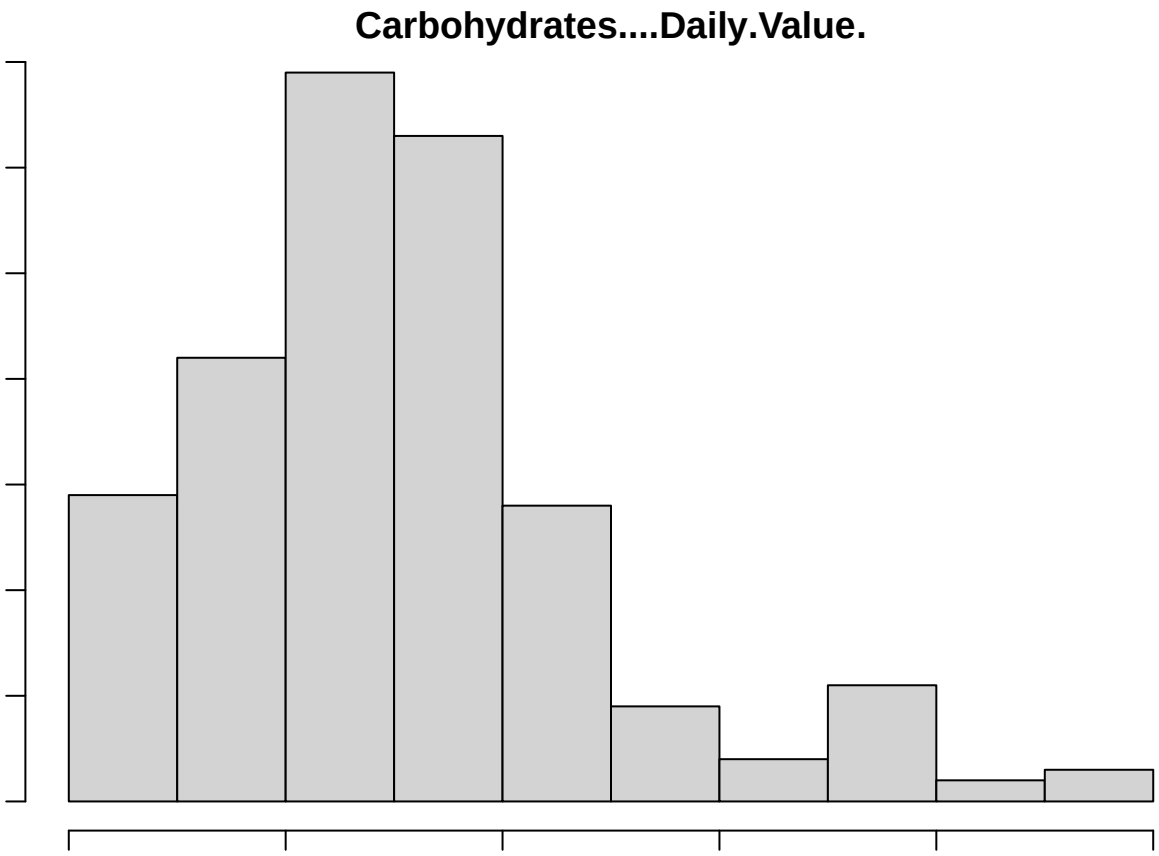

Cholesterol

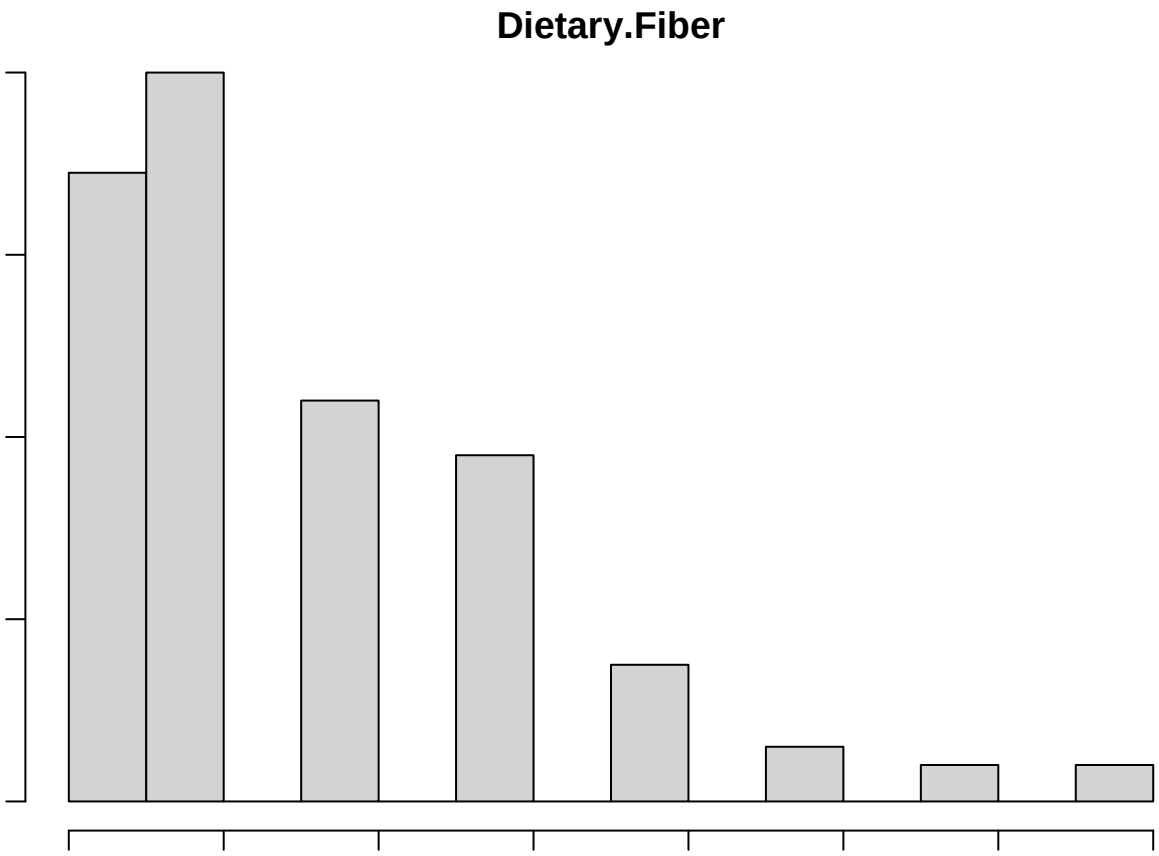

Cholesterol....Daily.Value.

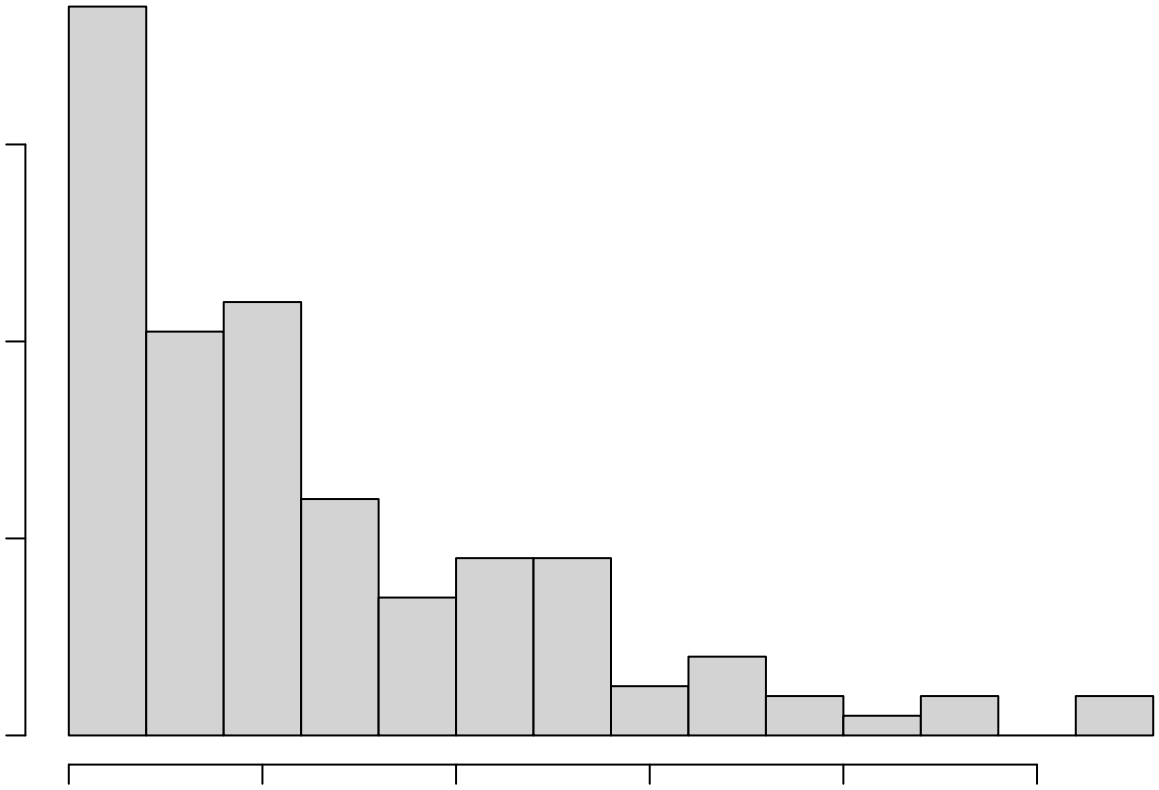

Sodium

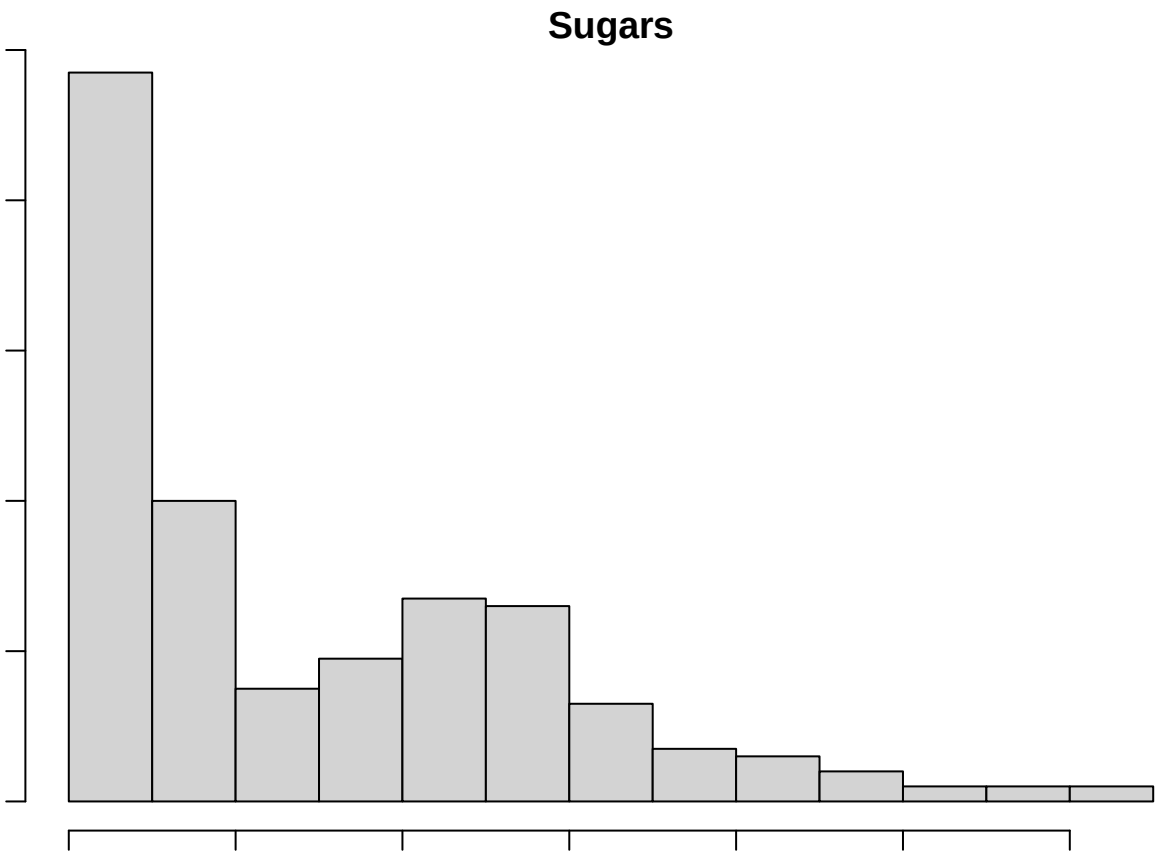

Sodium....Daily.Value.

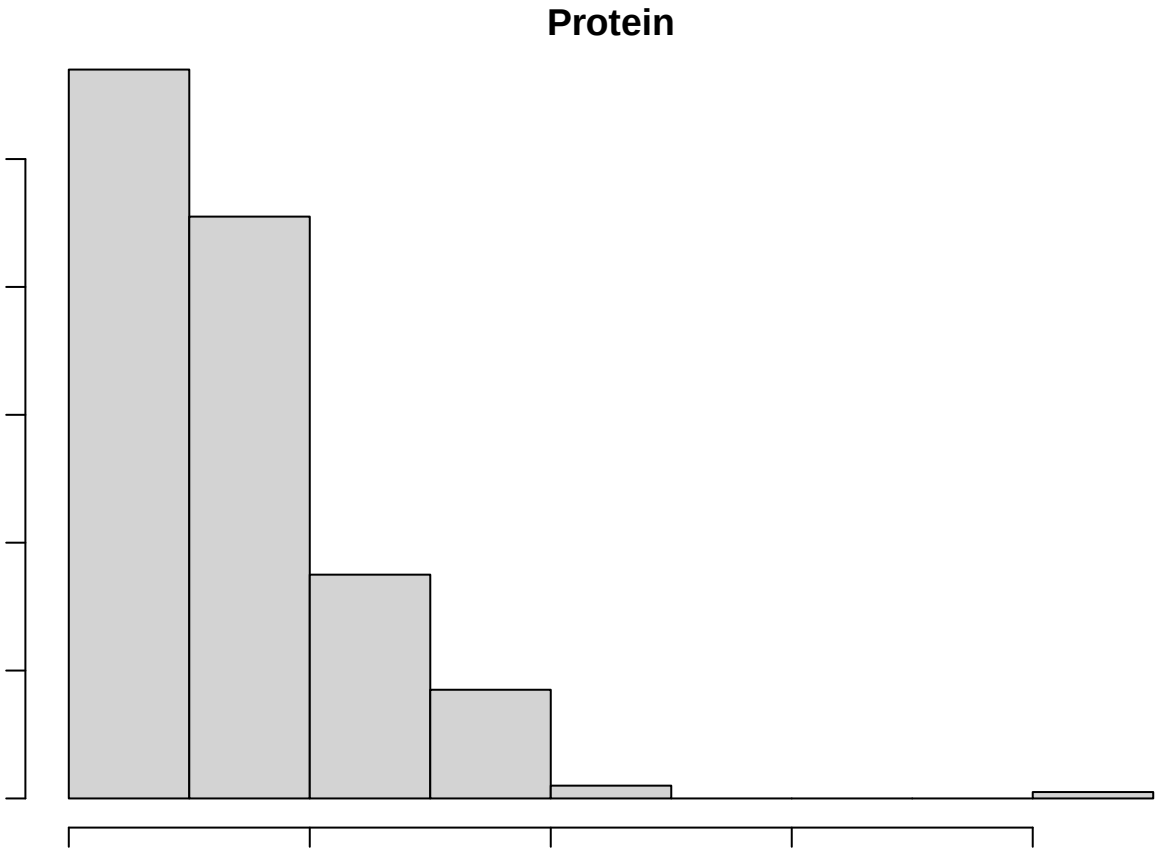

Carbohydrates

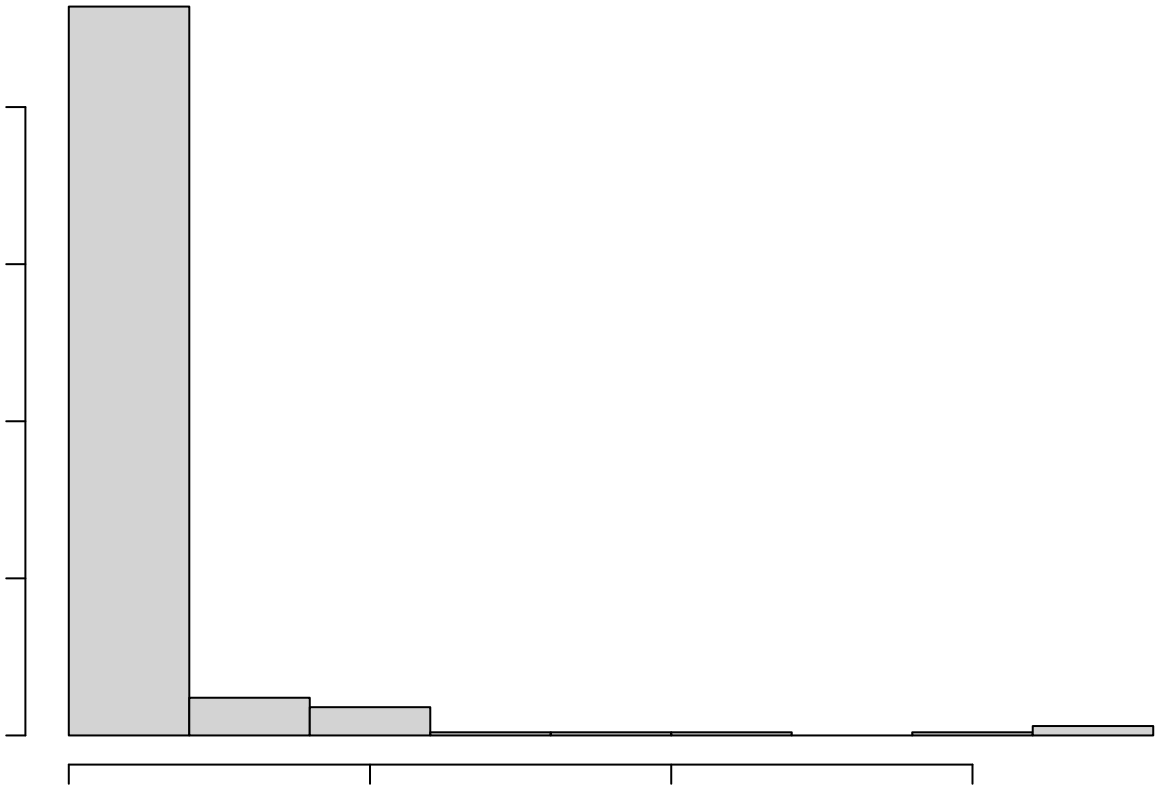

Carbohydrates....Daily.Value.

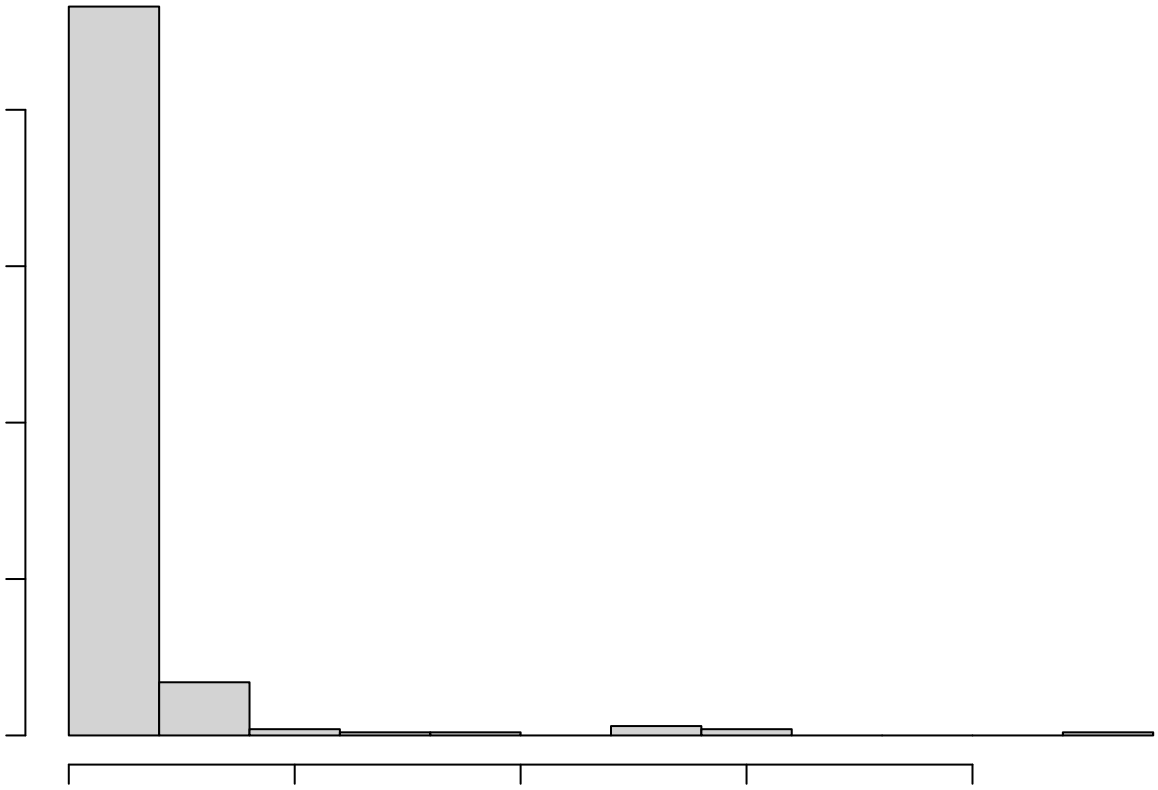

Dietary.Fiber

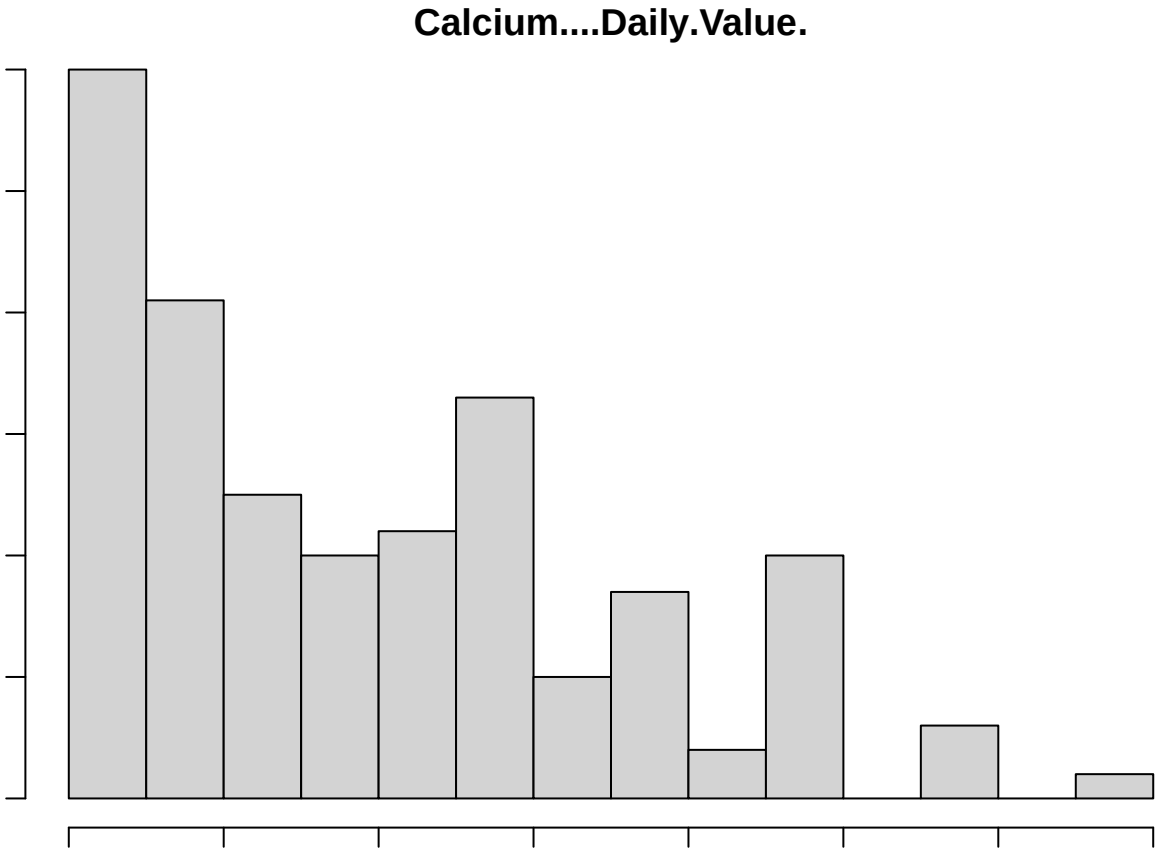

Dietary.Fiber....Daily.Value.

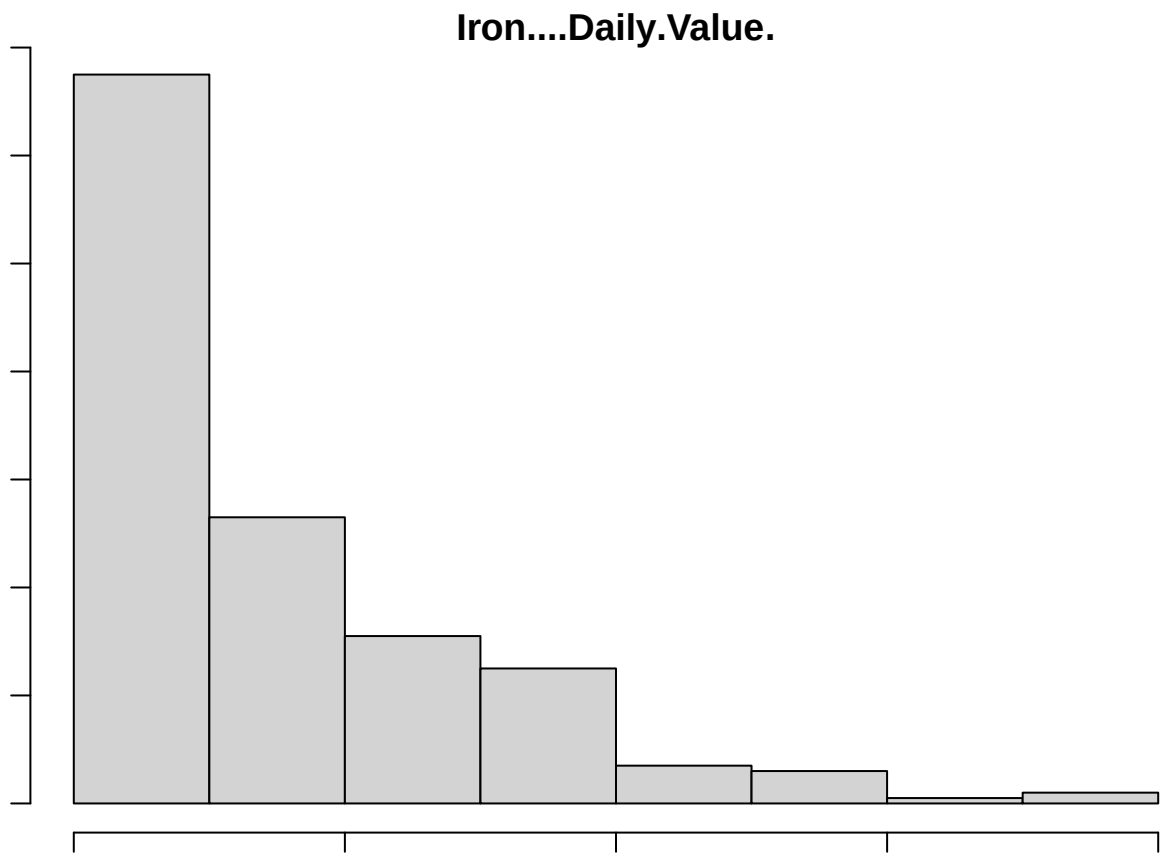

Sugars


Protein


Vitamin.A....Daily.Value.


Vitamin.C....Daily.Value.


Calcium....Daily.Value.



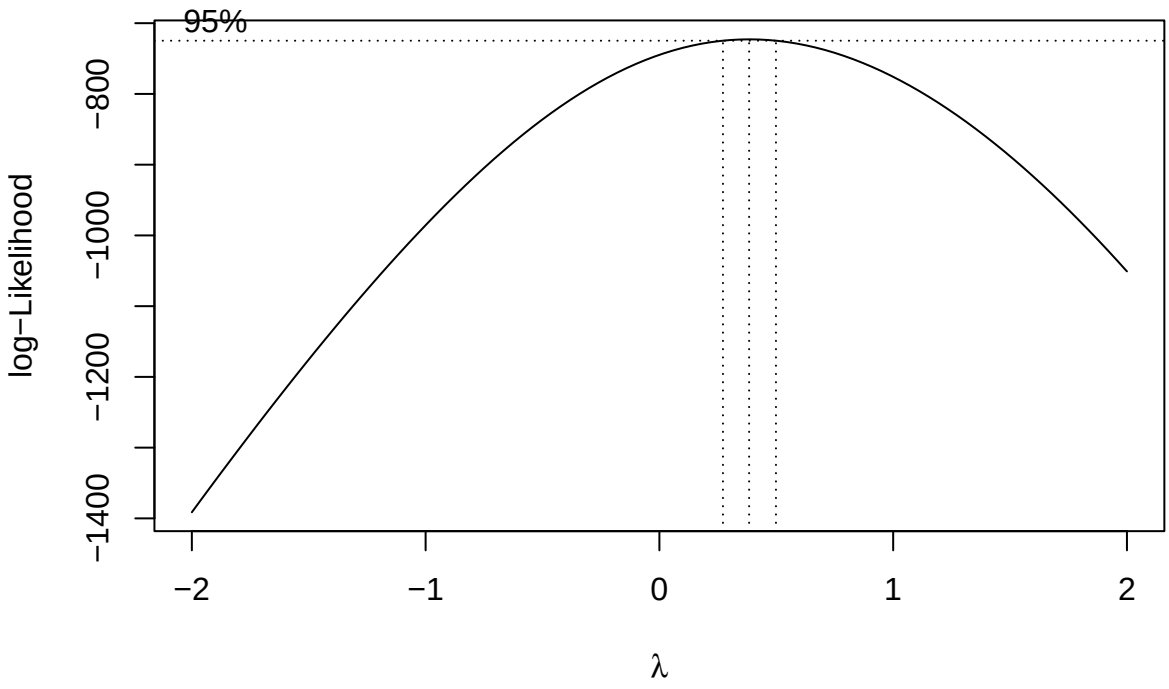
Escogeremos a la variable Protein para tratar de que se comparte como una variable normal.

```
prote <- df$Protein
# Adding a constant to make values positive
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
bc = boxcox((prote+1)~1)
```




```
# Valor de lambda con maximiza verosimilitud
opt_lambda <- bc$x[which.max(bc$y)]
```

```
# Ecuaciones de modelos
prote_aprox <- sqrt(prote+1)
prote_exact <- ((prote+1)^opt_lambda-1)/opt_lambda
```

Ecuaciones de los modelos encontrados

Análisis de normalidad

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

```
summary(prote_exact)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   2.227   4.368   4.021   5.622  11.923
```

```
summary(prote_aprox)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.236   3.606   3.461   4.472   9.381
```

```r
library(moments)
skewness(prote_exact)
```

```
## [1] -0.1152087
```

```r
skewness(prote_aprox)
```

```
## [1] 0.1354037
```

Podemos ver que el sesgo para los datos exactos es hacia la izquierda y que para los aproximados es hacia la derecha casi en un mismo valor.

```r
kurtosis(prote_exact)
```

```
## [1] 2.508812
```

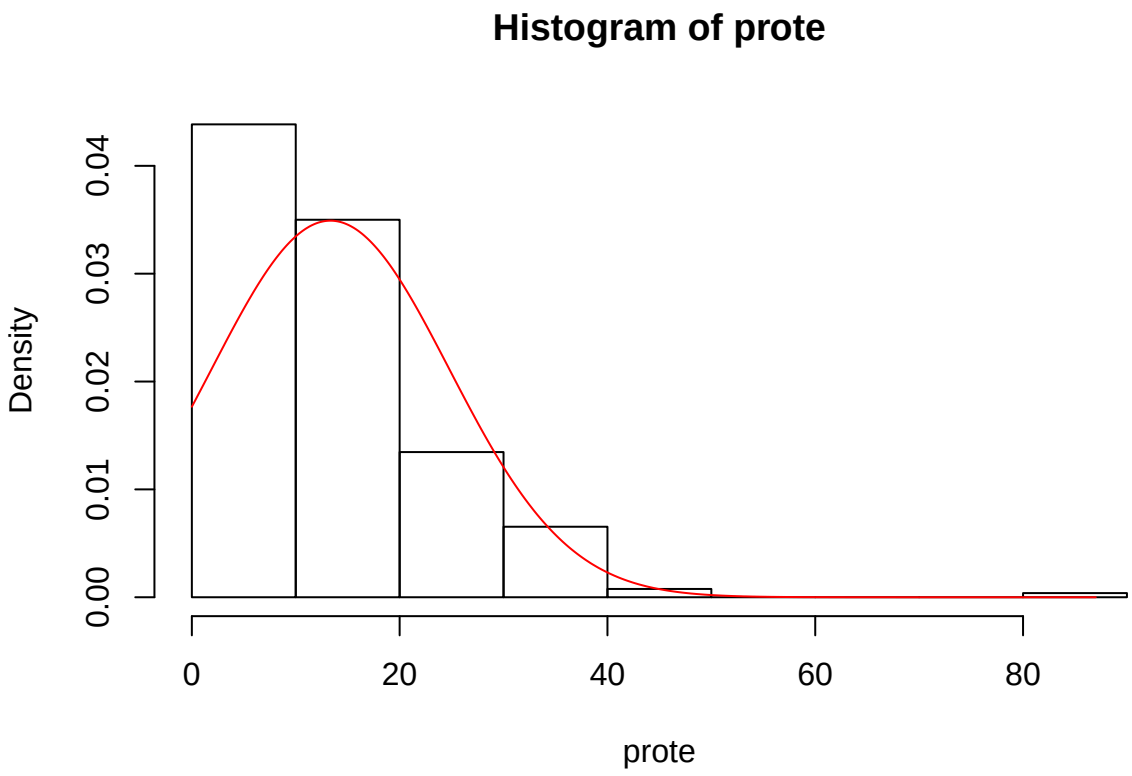
```r
kurtosis(prote_aprox)
```

```
## [1] 2.827596
```

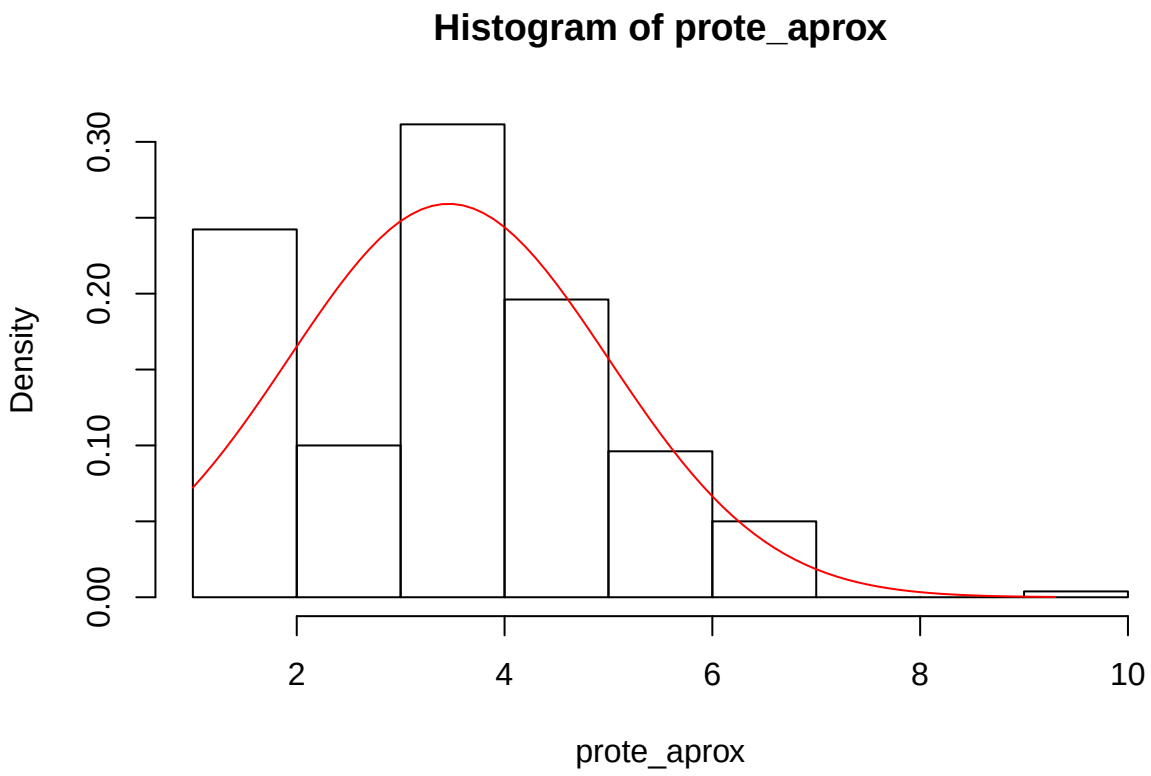
Vemos que la curtosis de ambos modelos no cambian mucho, ambos son menores y cercanos a tres por lo que diremos que son mesocúrticos.

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

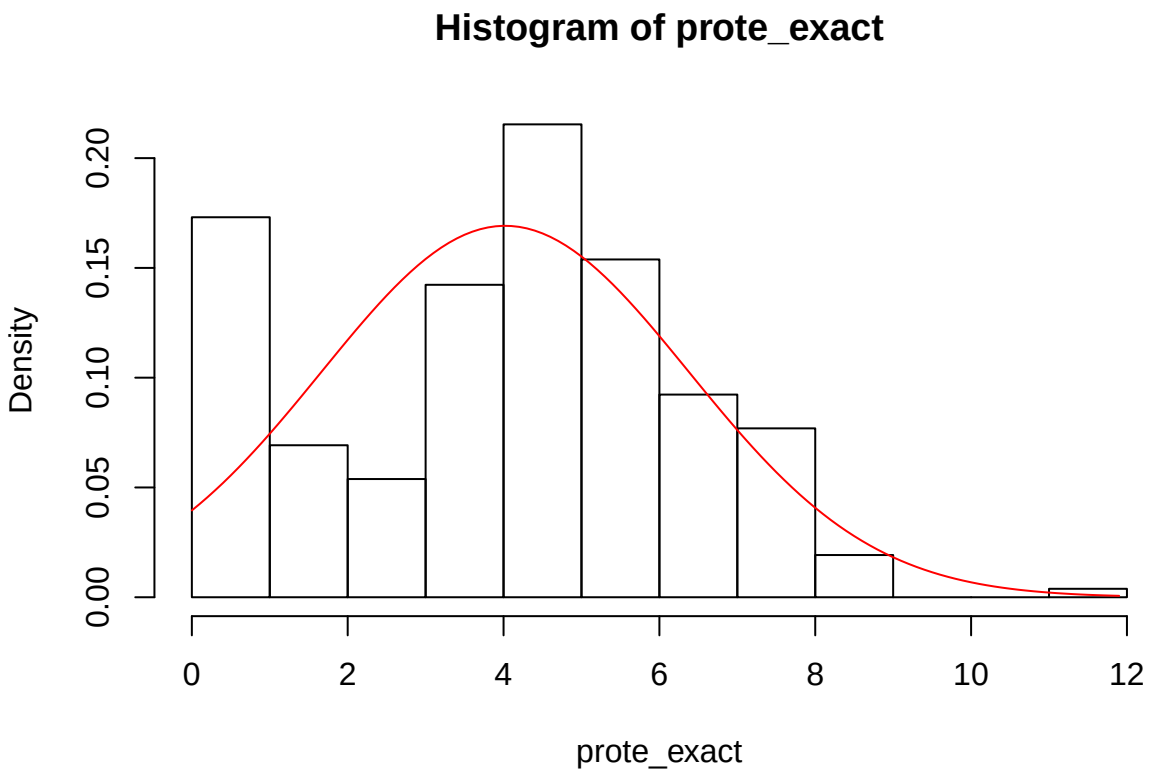
```r
# Histograma de datos originales
hist(prote,prob=TRUE,col=0)
x=seq(min(prote),max(prote),0.1)
y=dnorm(x,mean(prote),sd(prote))
lines(x,y,col="red")
```




```
# Histograma de modelo aproximado
hist(prote_aprox,prob=TRUE,col=0)
x=seq(min(prote_aprox),max(prote_aprox),0.1)
y=dnorm(x,mean(prote_aprox),sd(prote_aprox))
lines(x,y,col="red")
```




```
# Histograma de modelo exacto
hist(prote_exact,prob=TRUE,col=0)
x=seq(min(prote_exact),max(prote_exact),0.1)
y=dnorm(x,mean(prote_exact),sd(prote_exact))
lines(x,y,col="red")
```



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

```
library(nortest)

# Para datos originales
ad.test(prote)
```

```
##
##  Anderson-Darling normality test
##
## data:  prote
## A = 4.7515, p-value = 8.515e-12
```

```
# Para datos exactos
ad.test(prote_exact)
```

```
##
##  Anderson-Darling normality test
##
## data:  prote_exact
## A = 3.3747, p-value = 1.831e-08
```

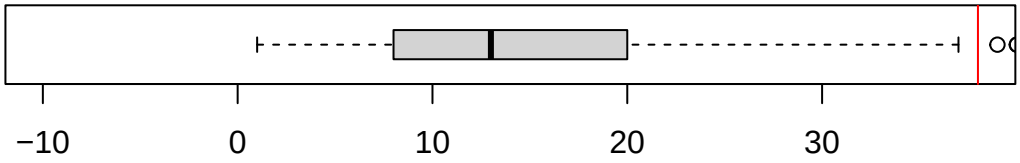
```r
# Para datos aproximados
ad.test(prote_aprox)
```

```
##
##  Anderson-Darling normality test
##
## data:  prote_aprox
## A = 2.4544, p-value = 3.218e-06
```

El p-value es muy bajo para los datos originales y transformados, se rechaza la hipótesis nula de distribución normal.

```r
new_prote=prote[prote != 0]
X = new_prote
q1=quantile(X,0.25)  #Cuantil 1 de la variable X
q3=quantile(X,0.75)
ri= q3-q1   #Rango intercuartílico de X
par(mfrow=c(2,1)) #Matriz de gráficos de 2x1
y1 = q1-1.5*ri
y2 = q3+1.5*ri
boxplot(X,horizontal=TRUE,ylim=c(y1,y2))
abline(v=q3+1.5*ri,col="red")  #linea vertical en el límite de los datos atípicos o extremos
X1= new_prote[new_prote<q3+1.5*ri]  #En la matriz M, quitar datos más allá de 3 rangos intercuartílicos
```

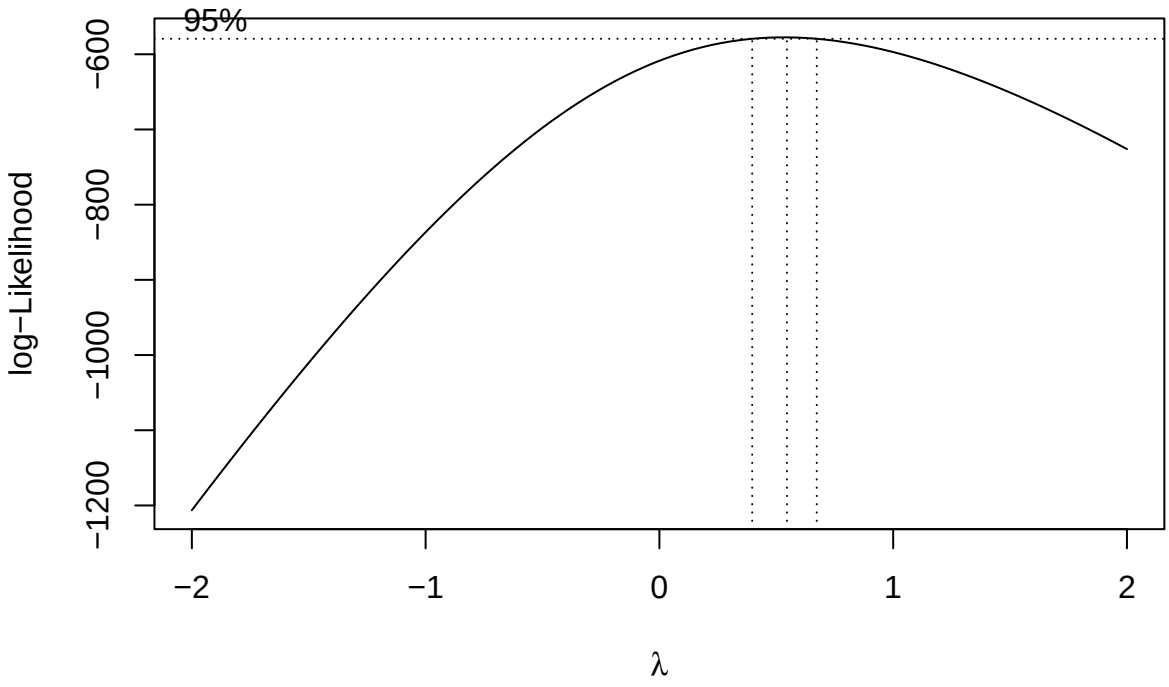
Detecta anomalías y corrige tu base de datos (datos atípicos, ceros anámalos, etc).



En el boxplot anterior se puede observar la distribución de los datos de la variable Protein del dataset, aquellos puntos que excedan la linea roja serán eliminados debido a que estan más de 3 cuartiles arriba del tercer cuartil, por lo que son considerados datos extremos.

```
library(MASS)
bc = boxcox((X1)~1)
```

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).



5. Transformación de Yeo Johnson

```
# Valor de lambda que maximiza la verosimilitud
opt_l = bc$x[which.max(bc$y)]
print(opt_l)
```

```
## [1] 0.5454545
```

Transformación Yeo Johnson

```
library(VGAM)
```

```
## Warning: package 'VGAM' was built under R version 4.2.3
```

```
## Loading required package: stats4
```

```
## Loading required package: splines
```

```
proteYeoTrans <- yeo.johnson(X1, lambda = opt_l)
```

```
Yeo_exact=((X1+1)^opt_l-1)/opt_l
Yeo_aprox=sqrt(X1)
```

6. Ecuación del modelo encontrado

7. Análisis de normalidad con la transformación de Yeo Johnson

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

```
summary(Yeo_exact)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8424  4.2443  5.9006  5.7614  7.5616 11.5001
```

```
summary(Yeo_aprox)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.828   3.606   3.490   4.359   6.083
```

Podemos observar que los valores del modelo exacto tanto de la media o el máximo, son mayores que comparados a los que se tienen en el modelo aproximado.

```
skewness(Yeo_exact)
```

```
## [1] -0.03158043
```

```
skewness(Yeo_aprox)
```

```
## [1] -0.2014249
```

Podemos ver un sesgo pequeño para el modelo exacto hecho con la transformación de Yeo Johnson, el modelo aproximado muestra mucho más sesgo hacia la izquierda.

```
kurtosis(Yeo_exact)
```

```
## [1] 2.384764
```

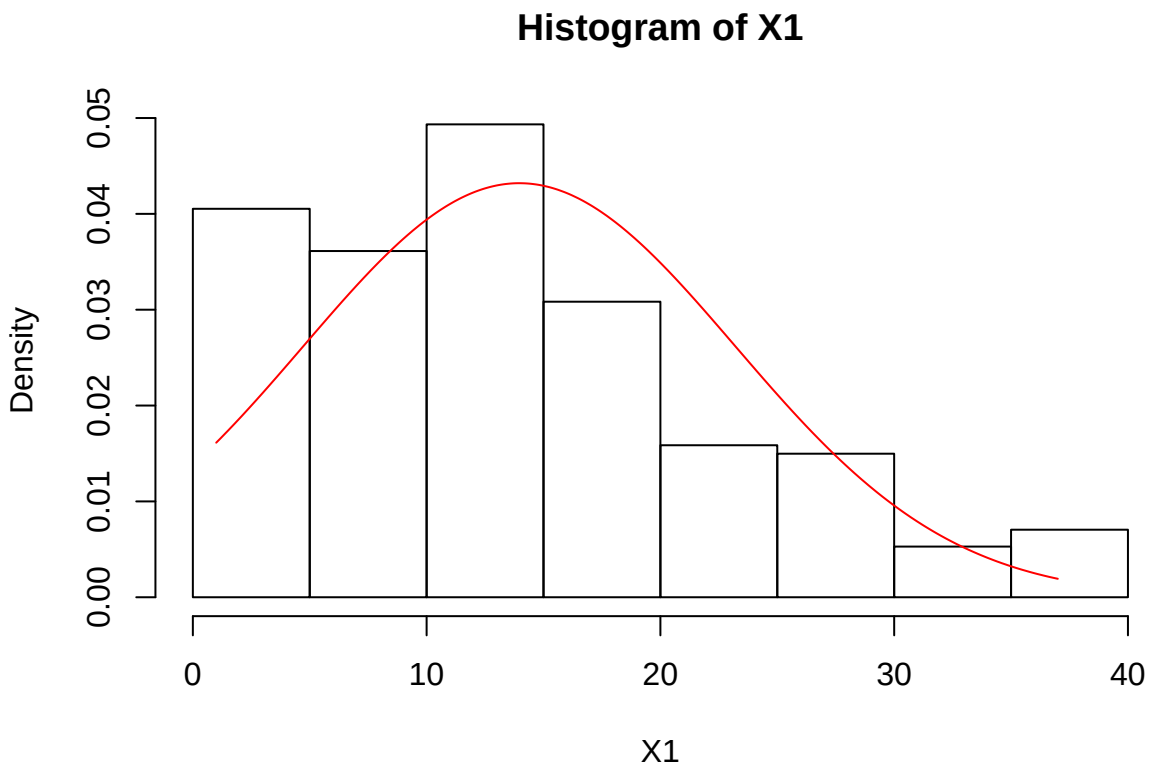
```
kurtosis(Yeo_aprox)
```

```
## [1] 2.428149
```

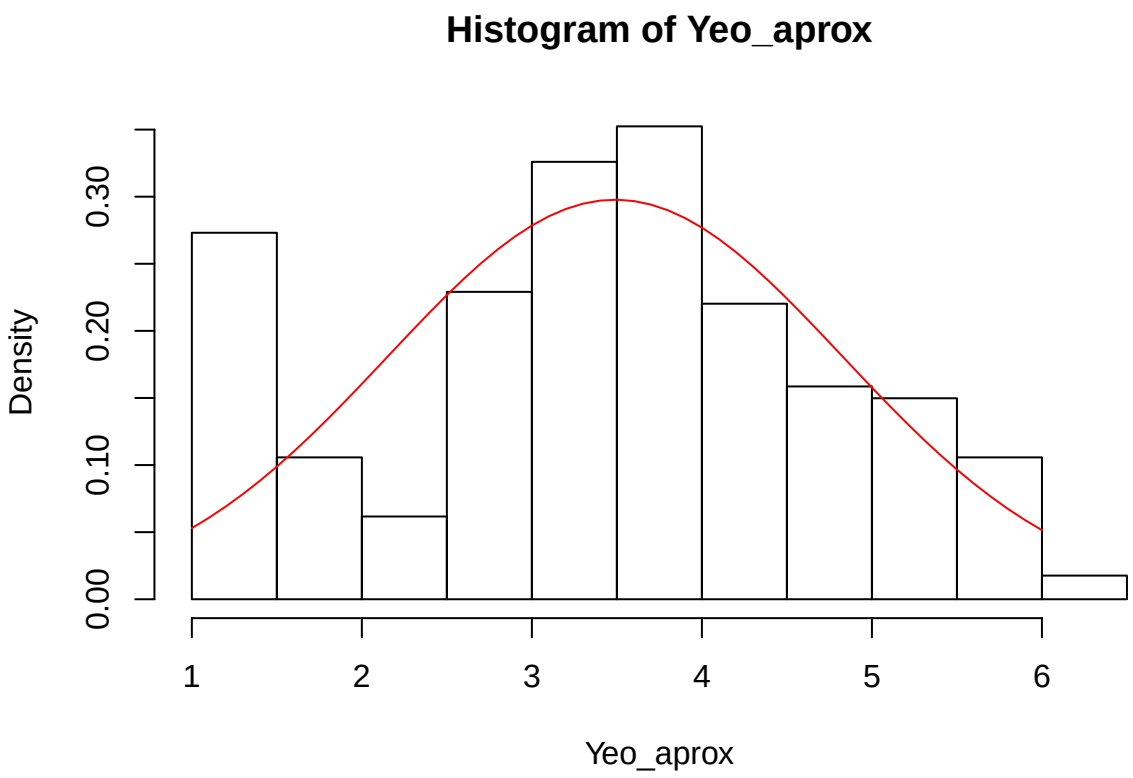
Vemos que la curtosis de ambos modelos no cambian mucho, ambos son menores y cercanos a tres por lo que diremos que son mesocúrticos.

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

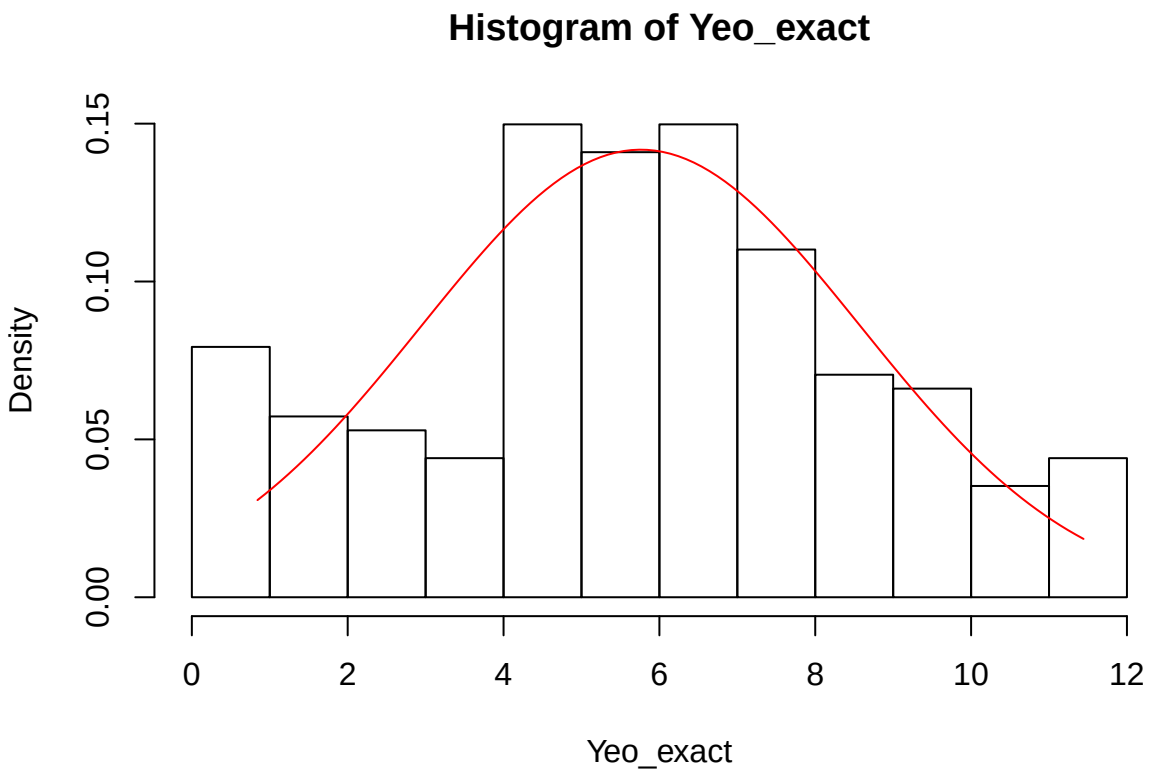
```
# Histograma de datos originales
hist(X1,prob=TRUE,col=0)
x=seq(min(X1),max(X1),0.1)
y=dnorm(x,mean(X1),sd(X1))
lines(x,y,col="red")
```




```
# Histograma de modelo aproximado
hist(Yeo_aprox,prob=TRUE,col=0)
x=seq(min(Yeo_aprox),max(Yeo_aprox),0.1)
y=dnorm(x,mean(Yeo_aprox),sd(Yeo_aprox))
lines(x,y,col="red")
```




```
# Histograma de modelo exacto
hist(Yeo_exact,prob=TRUE,col=0)
x=seq(min(Yeo_exact),max(Yeo_exact),0.1)
y=dnorm(x,mean(Yeo_exact),sd(Yeo_exact))
lines(x,y,col="red")
```



Se pueden observar que para ambos modelos transformados con Yeo Johnson,la distribución de los histogramas siguen en mayor medida a una normal que comparada a los datos originales.

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

```
library(nortest)

# Para datos originales
ad.test(X1)
```

```
##
##  Anderson-Darling normality test
##
## data:  X1
## A = 2.6706, p-value = 9.468e-07
```

```
# Para datos exactos
ad.test(Yeo_exact)
```

```
##
##  Anderson-Darling normality test
##
## data:  Yeo_exact
## A = 1.4337, p-value = 0.001028
```

```r
# Para datos aproximados
ad.test(Yeo_aprox)
```

```
##
##  Anderson-Darling normality test
##
## data:  Yeo_aprox
## A = 1.8542, p-value = 9.488e-05
```

Se observa que el p-value de todos los modelos es muy bajo, sin embargo el modelo exacto obtenido con la transformación de Yeo Johnson es mucho mayor al de sus otros dos casos, aún así se sigue rechazando la hipótesis nula de normalidad.

8. Define la mejor transformación de los datos de acuerdo a las características de los modelos que encontraste. La mejor transformación que se obtuvo bajo el objetivo de querer tener una distribución normal sobre los datos originales trabajados es el modelo exacto obtenido por la transformación de Yeo Johnson, aun y que el p-value es bajo y se rechaze la hipótesis de normalidad, las otras pruebas presentaron p-values mucho más bajos al de este por lo que diría que este es el modelo que sigue una distribución más normal de los presentes.

9. Concluye sobre las ventajas y desventajes de los modelos de Box Cox y de Yeo Johnson. Tanto los modelos Box-Cox como Yeo-Johnson ofrecen ventajas al mejorar la normalidad de datos, con Box-Cox adecuado para valores estrictamente positivos y Yeo-Johnson para datos más diversos. Aunque se observaron mejoras en la distribución, ninguna transformación garantizó normalidad completa en nuestro estudio. La elección entre ambas técnicas debe considerar la naturaleza de los datos y objetivos del análisis, y en algunos casos, podrían requerirse enfoques alternativos.

10. Analiza las diferencias entre la transformación y escalamiento de datos La transformación modifica la distribución de los datos para cumplir con ciertas suposiciones estadísticas, como normalidad. Puede incluir logaritmos, raíces cuadradas, etc. El escalamiento ajusta la escala de los datos sin cambiar su distribución, para asegurar que las diferentes características tengan un rango comparable.

La transformación es más útil cuando los datos presentan asimetrías o no siguen una distribución deseada. El escalamiento se aplica cuando las características tienen diferentes escalas numéricas.

El escalamiento puede preservar toda la información de los datos originales, mientras que la transformación en algunos casos puede alterar la relación entre los datos originales, esto puede llegar a ser problemático dependiendo de la naturaleza del problema.