3-transformaciones

August 19, 2023

1 Transformaciones

Francisco Mestizo Hernández A01731549

En esta actividad se mostrará un uso de las tranformadas en un set de datos real. El set a utilizar será el de McDonalds.

Primero, importamos las librerias que utilizaremos para algunos métodos de transformaciones.

- **moments**: La usamos para calcular la curtosis y sesgo de los datos.
- **MASS**: Tiene implementada la función de Box-cox.
- **nortest**: Tiene implementada la prueba de normalidad Anderson-Darling.
- **VGAM**: Tiene implementada la transformación de Yeo Johnson.

[1]:
```
#Instalamos las librerias que vamos a usar (para curtosis y coeficiente de␣
 ↪sesgo)
install.packages("moments")
install.packages("MASS")
install.packages("nortest")
install.packages("VGAM")

#Leemos los datos del csv (No los imprimo porque ocupan mucho espacio en la␣
 ↪pantalla)
M = read.csv('/content/sample_data/mc-donalds-menu-1.csv')
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

1.1 Transformación con Box-cox

Para iniciar con la transformación Box-cox, seleccionamos el dato que vamos a analizar (las proteínas de los alimentos de McDonalds) y posteriormente verificamos si tiene valores en 0. Esto lo hacemos porque es un método que no puede trabajar con negativos ni cero en los datos.

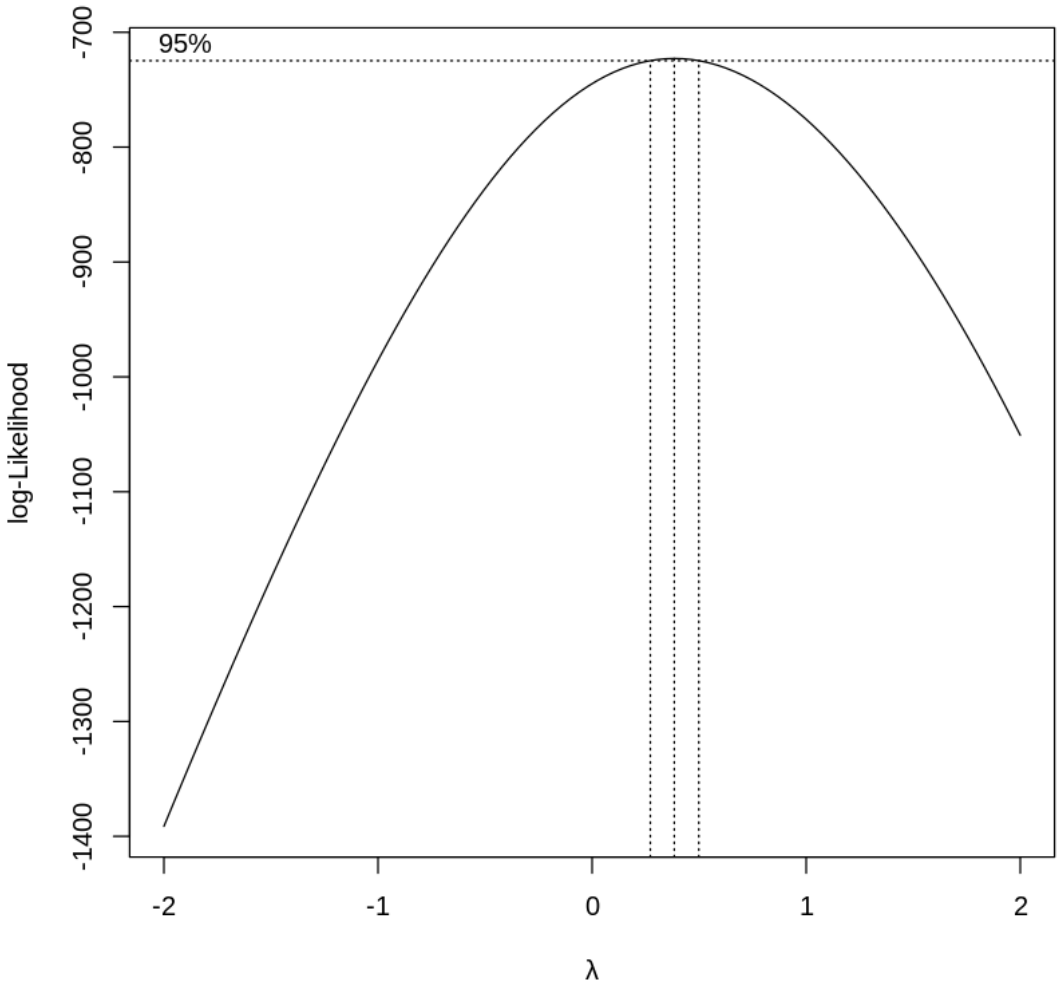
[42]:
```
#Seleccionamos la variable
x = M$Protein

#verificamos si hay 0
min(x)
```

0

Podemos ver que los datos sí contienen valores en 0 para las proteínas. Debido a esto, ejecutamos el método sumando un 1 a todos los datos. Una vez ejecutado el modelo, desplegamos el grafico de las lambdas calculadas y cual es la que mejor transforma el modelo. Esta se encuentra en la punta superior de la curva.

[43]:
```
bc <- MASS::boxcox((x+1)~1)
l = bc$x[which.max(bc$y)]
l #lambda que optimiza
```

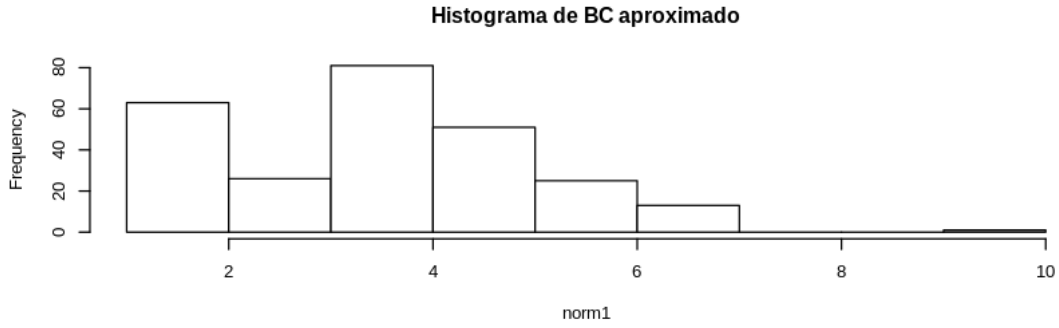
0.383838383838384

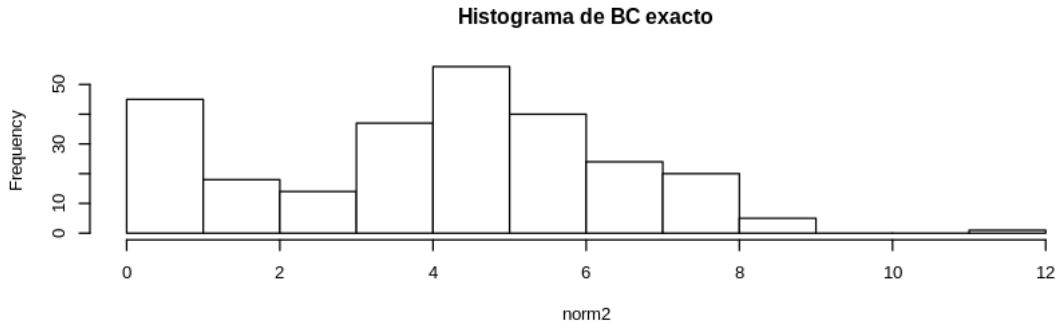


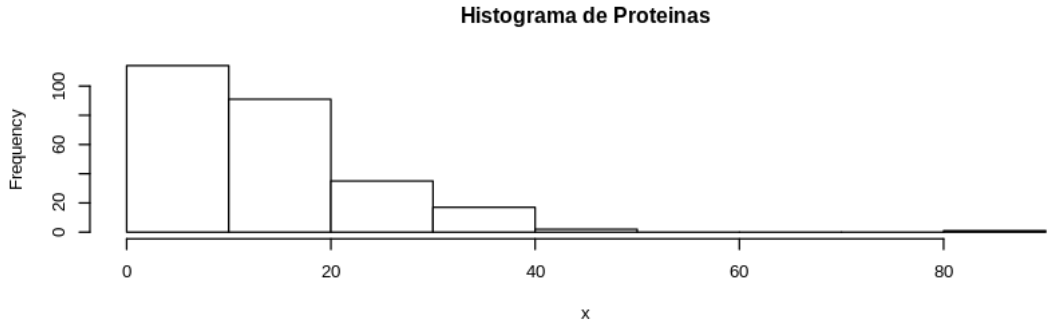
Obtuvimos un valor aproximado para lambda de 0.383. A continuacion hacemos la definición de la funcion exacta (norm2) y de la función aproximada (norm1). La funcion aproximada la obtenemos de la tabla, y como el valor más cercano a 0.383 es 0.5, entonces seleccionamos la función de raíz cuadrada de x.

Sumamos un 1 a la x de los dos modelos porque el set de datos lo estabamos trabajando con ese uno extra.

[44]:
```
norm1 = sqrt(x+1)
norm2 = ((x+1)^l - 1) / l
par(mfrow = c(3,1))
hist(norm1, col = 0, main = "Histograma de BC aproximado")
hist(norm2, col = 0, main = "Histograma de BC exacto")
hist(x, col = 0, main = "Histograma de Proteinas")
```







Podemos ver en los histogramas que, aunque realizamos la transformación, todavía existe una gran cantidad de datos cargados a 0. Antes de verificar esos datos, podemos hacer la prueba de normalidad a las tres transformaciones.

[45]:
```
D0 = nortest::ad.test(x)
D1 = nortest::ad.test(norm1)
D2 = nortest::ad.test(norm2)

m0 = round(c(as.numeric(summary(x)), moments::kurtosis(x), moments::
 ↪skewness(x), D0$p.value), 3)
m1 = round(c(as.numeric(summary(norm1)), moments::kurtosis(norm1), moments::
 ↪skewness(norm1), D1$p.value), 3)
```

```
m2 = round(c(as.numeric(summary(norm2)), moments::kurtosis(norm2), moments::
 ↪skewness(norm2), D2$p.value), 3)

m<-as.data.frame(rbind(m0, m1, m2))
row.names(m) = c("Original", "Primer modelo", "Segundo modelo")
names(m) = c("Minimo", "Q1", "Mediana", "Media", "Q3", "Maximo", "Curtosis",␣
 ↪"Sesgo", "Valor p")
m
```

A data.frame: 3 × 9

	Minimo <dbl>	Q1 <dbl>	Mediana <dbl>	Media <dbl>	Q3 <dbl>	Maximo <dbl>	Curtosis <dbl>	Ses <d
Original	0	4.000	12.000	13.338	19.000	87.000	8.864	1.5
Primer modelo	1	2.236	3.606	3.461	4.472	9.381	2.828	0.1
Segundo modelo	0	2.227	4.368	4.021	5.622	11.923	2.509	-0.

Podemos observar en la tabla que existe una gran diferencia en los valores de curtosis y sesgo de los dos modelos transformados en comparación con los datos originales. Bajamos la curtosis de casi 9 a un poco menos de 3, que son valores aceptables.

De todas formas, podemos probar hacer la transformación eliminando los datos en 0, para ver si obtenemos resultados diferentes. Pero antes de eliminar los datos es importante verlos y analizar si es posible retirarlos para hacer la transformación.

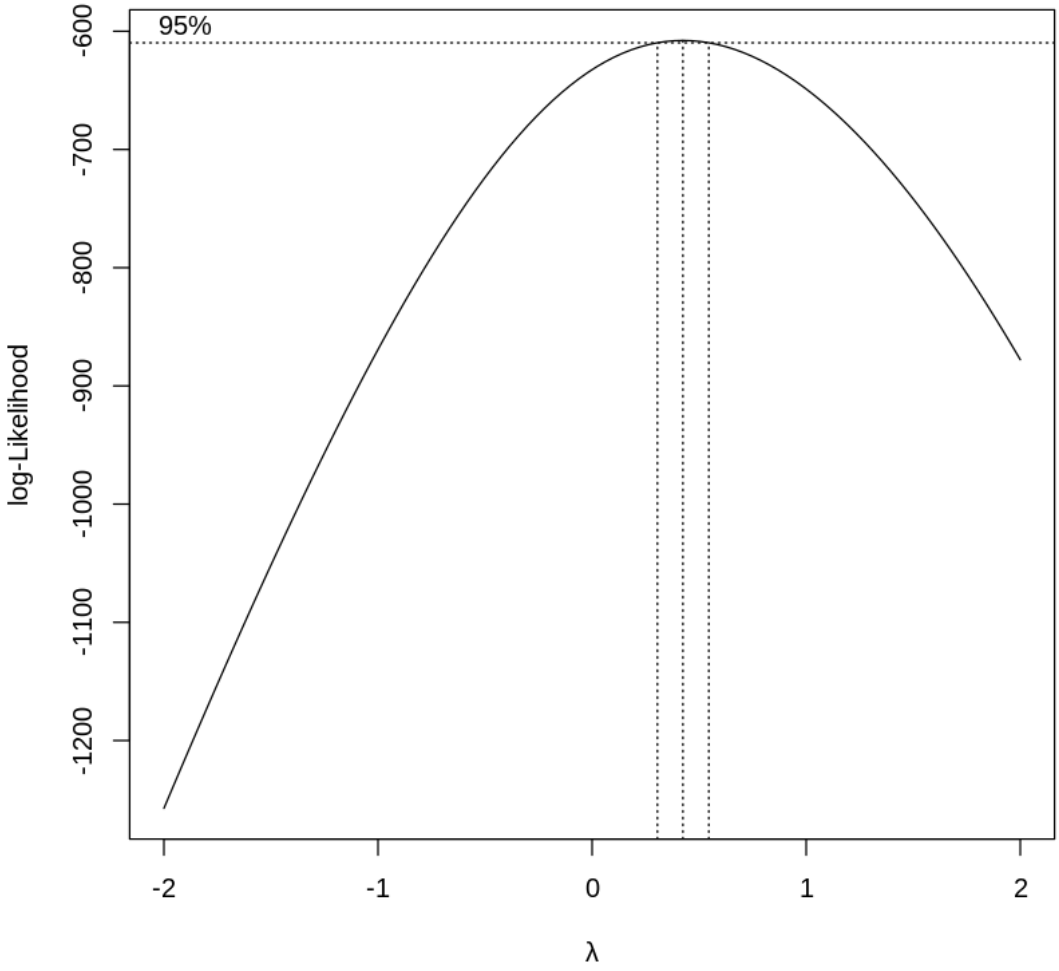
[46]:
```
#Observamos las columnas que estan en 0 para las proteinas
 #M[M$Protein == 0, ]
```

Al seleccionar los valores en 0, nos damos cuenta que no son productos que haga directamente McDonalds, ya que son manzanas, refrescos y cafés. Es por esto, que podemos eliminarlos para hacer el análisis.

Eliminamos los datos en 0 y volvemos a ejecutar la transformación de Box-cox

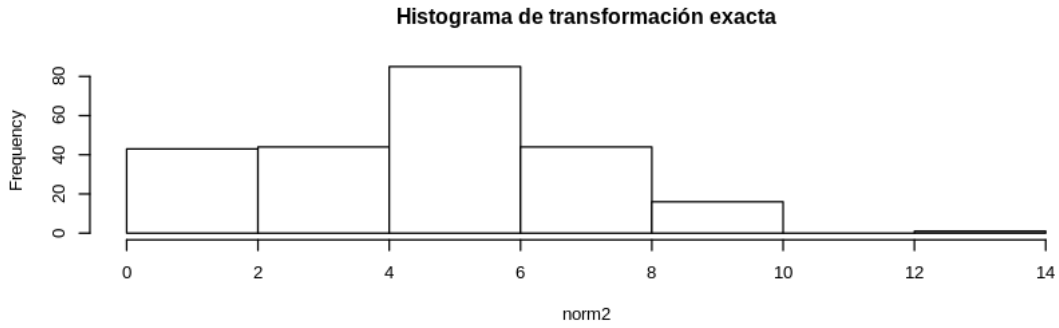
[47]:
```
#Eliminamos registros en 0
x = M$Protein
x = x[x != 0]

#Ejecutamos el Box-com
bc <- MASS::boxcox((x)~1)
l = bc$x[which.max(bc$y)]
l #lambda que optimiza
```

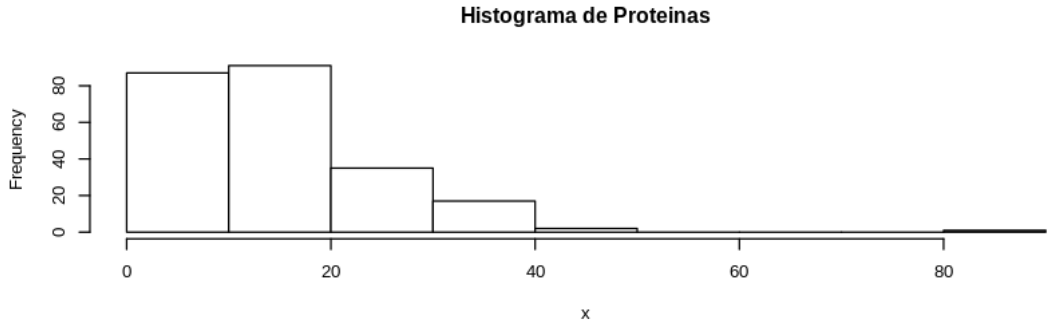
0.424242424242424



Podemos ver que el nuevo valor propuesto para lambda es de 0.424. A continuación se muestra la transformación con la función aproximada, que sigue siendo la misma ya que 0.5 es el valor más cercano. Y también se muestra la transformación exacta.

[48]:
```
norm1 = sqrt(x) # Transformación aproximada
norm2 = ((x)^l - 1) / l #Transformación exacta

#Histogramas
par(mfrow = c(3,1))
hist(norm1, col = 0, main = "Histograma de transformación aproximada")
hist(norm2, col = 0, main = "Histograma de transformación exacta")
hist(x, col = 0, main = "Histograma de Proteinas")
```





A pesar de haber eliminado los datos con 0 proteínas, vemos que las transformaciones no resulta completamente normales. Podemos hacer las pruebas de normalidad para ver si se mejoraron los resultados.

[49]:
```
D0 = nortest::ad.test(x)
D1 = nortest::ad.test(norm1)
D2 = nortest::ad.test(norm2)

m0 = round(c(as.numeric(summary(x)), moments::kurtosis(x), moments::
 ↪skewness(x), D0$p.value), 3)
m1 = round(c(as.numeric(summary(norm1)), moments::kurtosis(norm1), moments::
 ↪skewness(norm1), D1$p.value), 3)
```

```
m2 = round(c(as.numeric(summary(norm2)), moments::kurtosis(norm2), moments::
 ↪skewness(norm2), D2$p.value), 3)

m<-as.data.frame(rbind(m0, m1, m2))
row.names(m) = c("Original", "Primer modelo", "Segundo modelo")
names(m) = c("Minimo", "Q1", "Mediana", "Media", "Q3", "Maximo", "Curtosis",␣
 ↪"Sesgo", "Valor p")
m
```

A data.frame: 3 × 9

	Minimo <dbl>	Q1 <dbl>	Mediana <dbl>	Media <dbl>	Q3 <dbl>	Maximo <dbl>	Curtosis <dbl>	Ses <d
Original	1	8.000	13.000	14.884	20.000	87.000	9.899	1.7
Primer modelo	1	2.828	3.606	3.579	4.472	9.327	3.280	0.1
Segundo modelo	0	3.338	4.641	4.518	6.044	13.318	3.040	-0.

Como podemos ver, la curtosis y son mucho más altos cuando eliminamos los datos que están en 0, para cualquiera de los tres modelos. Aún así, los dos modelos transformados podríamos utilizarlos, ya que tienen un grado de curtosis de aproximadamente 3. A pesar de ser peores que los anteriores, pueden seguir siendo utilizables.

1.2 Transformación con Yeo Johnson

Ahora vamos a realizar el mismo procedimiento utilizando la transformación de Yeo Johnson. La ventaja en comparación con la Box-cox es que esta transformación si permite datos negativos o con valor de 0.

Comenzamos seleccionando la variable nuevamente.

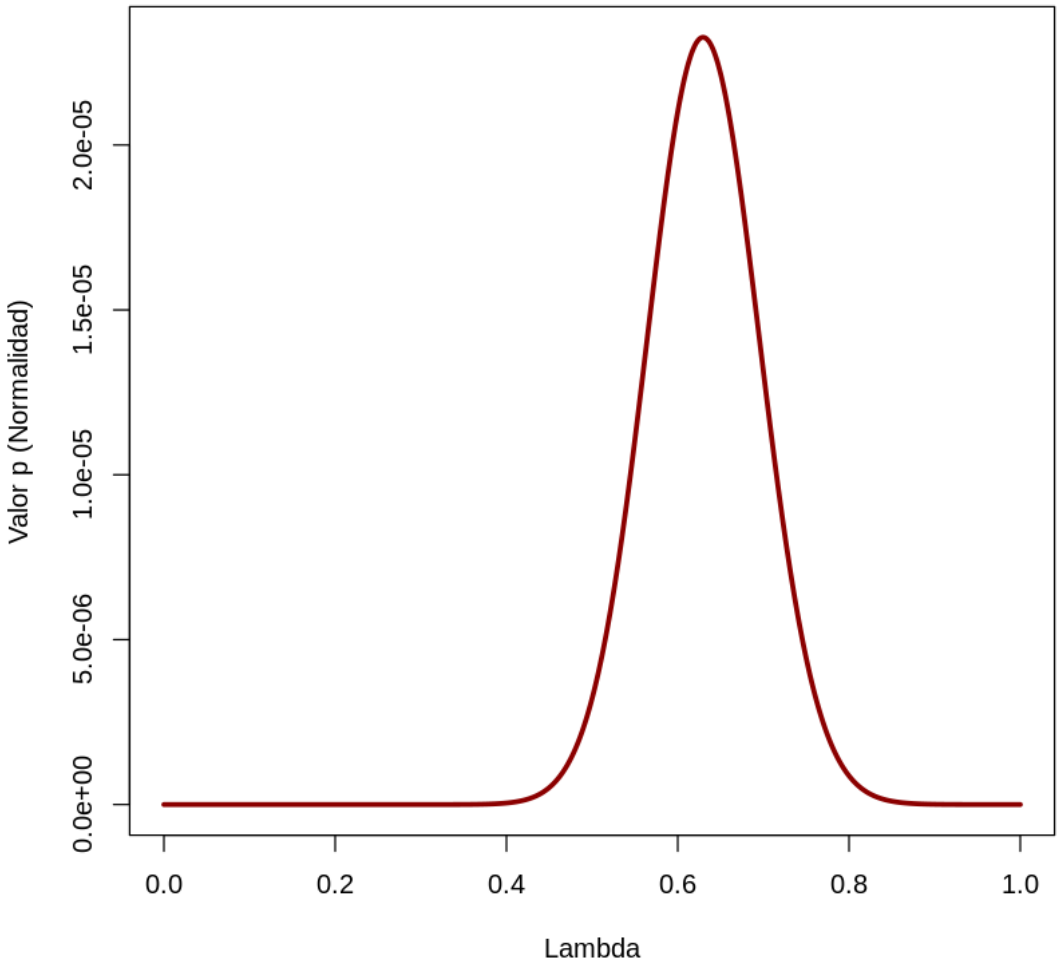
[50]:
```
#Seleccionamos la variable
x = M$Protein
```

Ejecutamos el modelo Yeo Johnson. Es importante mencionar que para ejecutarlo, estamos partiendo de el lambda obtenido en el Box-cox. Para este, debemos hacer varias iteraciones para determinar donde esta el mejor lambda. Nuevamente el mejor lambda se imprime después junto con la gráfica de todos los valores de lambda que se pueden tomar.

[51]:
```
yj <- VGAM::yeo.johnson((x), lambda = l)
lp<-seq(0,1,0.001)
nlp <-length(lp)
n=length(x)
D<-matrix(as.numeric(NA), ncol = 2, nrow=nlp)
d <- NA
for(i in 1:nlp){
  d = VGAM::yeo.johnson((x), lambda = lp[i])
  p = nortest::ad.test(d)
  D[i,] = c(lp[i], p$p.value)
}

N=as.data.frame(D)
```

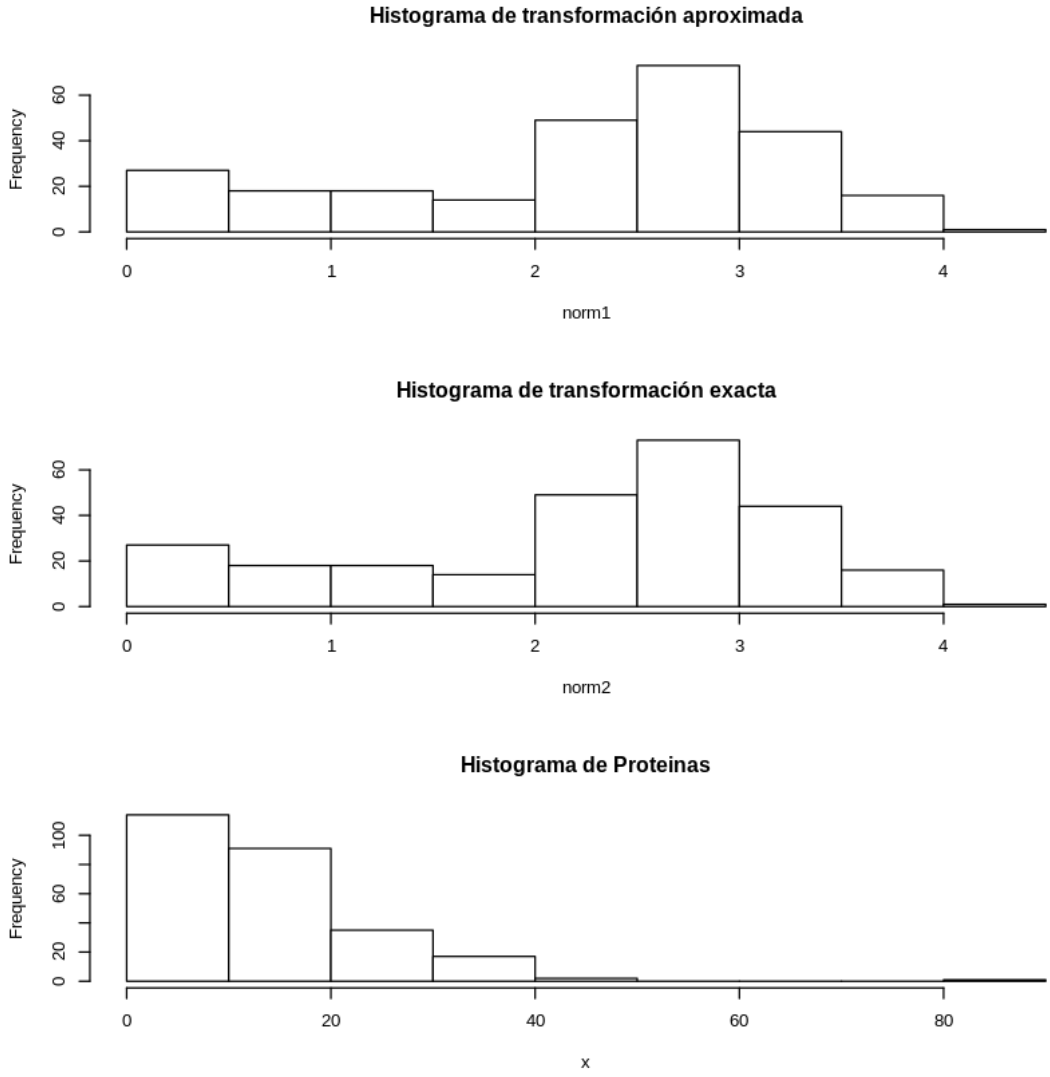
```r
plot(N$V1, N$V2,
type = "l", col = "darkred", lwd = 3,
xlab = "Lambda", ylab = "Valor p (Normalidad)")
l = max(N$V2)
l
```

2.32655168644656e-05



Podemos ver que el mejor valor es muy cercano a 0. El valor exacto es de aproximadamente 0.0000232. Útilizamos este valor para la transformación exacta y para la exacta tomamos el valor en la tabla para un lambda de 0, que es log(x)

[60]:
```
norm1 = log(x+1)
norm2 = ((x+1)^l-1)/l
par(mfrow = c(3,1))
hist(norm1, col = 0, main = "Histograma de transformación aproximada")
hist(norm2, col = 0, main = "Histograma de transformación exacta")
hist(x, col = 0, main = "Histograma de Proteinas")
```



Podemos ver en los histogramas son más normales que el original. Podemos hacer una prueba de normalidad para confirmarlo.

[61]:
```
D0 = nortest::ad.test(x)
D1 = nortest::ad.test(norm1)
D2 = nortest::ad.test(norm2)
```

```
m0 = round(c(as.numeric(summary(x)), moments::kurtosis(x), moments::
 ↪skewness(x), D0$p.value), 3)
m1 = round(c(as.numeric(summary(norm1)), moments::kurtosis(norm1), moments::
 ↪skewness(norm1), D1$p.value), 3)
m2 = round(c(as.numeric(summary(norm2)), moments::kurtosis(norm2), moments::
 ↪skewness(norm2), D2$p.value), 3)

m<-as.data.frame(rbind(m0, m1, m2))
row.names(m) = c("Original", "Primer modelo", "Segundo modelo")
names(m) = c("Minimo", "Q1", "Mediana", "Media", "Q3", "Maximo", "Curtosis",␣
 ↪"Sesgo", "Valor p")
m
```

A data.frame: 3 × 9

	Minimo <dbl>	Q1 <dbl>	Mediana <dbl>	Media <dbl>	Q3 <dbl>	Maximo <dbl>	Curtosis <dbl>	Ses <d
Original	0	4.000	12.000	13.338	19.000	87.000	8.864	1.5
Primer modelo	0	1.609	2.565	2.230	2.996	4.477	2.601	-0.
Segundo modelo	0	1.609	2.565	2.231	2.996	4.478	2.601	-0.

Podemos ver que la curtosis sí se reduce a valores óptimos, cercanos a 3, junto con el sesgo. Aún así, son valores parecidos a los que ya habíamos obtenido con la transformación Box-cox.

1.3 Conclusión y comparaciones

Para definir la mejor transformación para usar, podemos decir que es alguna de las dos del Yeo Johnson, ya que estás son las que tienen la media más cercana a la mediana. Además, tienen un buen valor de curtosis y de sesgo. A pesar que el modelo aproximado tiene una curtosis más alta, puede ser mejor utilizarlo por ser una función más sencilla.

Por lo tanto, la transformación elegida es log(x)

El problema con los resultados obtenidos es que todos tienen un valor p muy bajo, por lo que nos provocaría que rechaza la hipótesis. Esto puede ser ocacionado por los valores en 0 o porque en los histogramas observamos que el comportamiento no es completamente normal.

1.3.1 Transformación Box-Cox vs. Yeo Johnson

Una de las principales diferencias entre estos dos métodos es que el primero no acepta valores negativos o de 0, mientras que el YJ sí los puede manejar.

Por otro lado, BC lo podemos utilizar sin conocer un valor previo de lambda, mientras que YJ necesita iniciar con un valor de lambda y hacer iteraciones para encontrar el mejor.

1.3.2 Transformaciones vs. escalamiento

Transformaciones

Las transformaciones las utilizamos cuando tenemos un conjunto de datos que no es normal. Al aplicar alguno de los métodos vistos arriba, podemos convertir los datos a una distribución normal. El problema de esto es que los datos transformados pierden totalmente el sentido. Lo que se hace

es transformar los datos, calcular alguna regresión, y sustituir la transformación en la regresión para que los resultados vuelvan a tener sentido.

Las transformaciones se pueden realizar tanteando o con los métodos vistos arriba, así como obtenemos dos resultados, uno exacto, con una función compleja y uno aproximado con una función sencilla.

Escalamiento

El escalamiento conserva la distribución de los datos, pero nos ayuda a comparar variables o a interpretar los datos de diferentes formas. Por ejemplo, con el escalamiento min-max, podemos saber en que porcentaje se encuentran los datos. Con el escalamiento estándar podemos saber cuantos datos se encuentran alejados de la media y qué tan alejados están (en términos de la desviación estándar)

Diferencias

Como podemos ver, las transformaciones nos ayudan a normalizar los datos, perdiendo el sentido de la escala y valores que se tienen en los datos perse. Pero que nos ayuda a preparar los datos para métodos que necesitan datos normalizados, como una regresión.

Y, por el otro lado, el escalamiento nos ayuda a cambiar la escala de los datos, manteniendo su significado, para hacer análisis de las características de los datos o para compararlos con otras variables.