Practica 2: Limpieza y validación de los datos

Anddy Aldave Valle, Alejandro Pulido Duque

13 de mayo de 2020

1. Descripción del dataset. ¿Por qué es importante y qué pregunta/problema pretende responder?

Los campos del conjunto de datos son los siguientes

x: El número de registro

carat: Peso en kilates del diamante

cut: La calidad del corte del diamante

color: El color del diamante

clarity: Califica la apariencia visual de cada diamante

depth: La altura de un diamante, medida desde el culet, o base, hasta la tabla, o tope superior, divido por su diámetro medio (z / media(x, y))

table: El ancho de la tabla del diamante expresado como un porcentaje de su diámetro medio

price: El precio del diamante

x: longitud del diamante en mm

y: ancho del diamante en mm

z: profundidad del diamante en mm

A continuación, para mayor claridad, se incluyen las partes del diamante

2. Integración y selección de los datos de interés a analizar.

```
# Lectura de datos
diamonds_df <- read.csv("diamonds.csv", header = TRUE, sep=",", dec = ".")
#Ejecutando la funcion summary sobre el dataframe, nos mostrará un resumen de cada variable
summary(diamonds_df)
```

```
##        X             carat               cut        color        clarity
##  Min.   :    1   Min.   :0.2000   Fair      : 1610   D: 6775   SI1    :13065
##  1st Qu.:13486   1st Qu.:0.4000   Good      : 4906   E: 9797   VS2    :12258
##  Median :26971   Median :0.7000   Ideal     :21551   F: 9542   SI2    : 9194
##  Mean   :26971   Mean   :0.7979   Premium   :13791   G:11292   VS1    : 8171
##  3rd Qu.:40455   3rd Qu.:1.0400   Very Good:12082   H: 8304   VVS2   : 5066
##  Max.   :53940   Max.   :5.0100                     I: 5422   VVS1   : 3655
##                                                     J: 2808   (Other): 2531
```

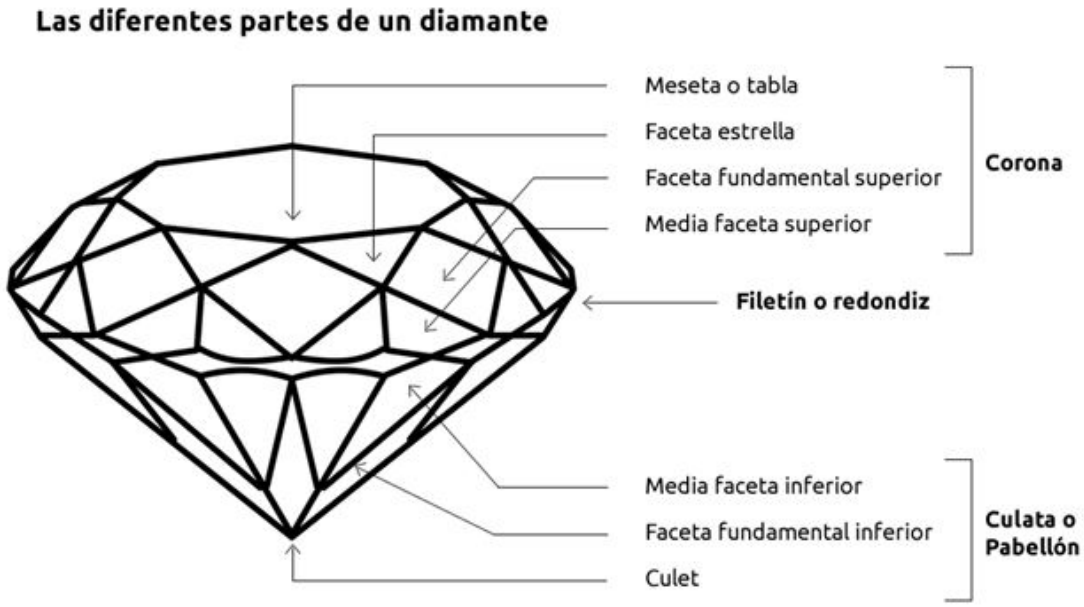


Figure 1: Partes del diamante

```
##      depth           table           price             x
##  Min.   :43.00   Min.   :43.00   Min.   :  326   Min.   : 0.000
##  1st Qu.:61.00   1st Qu.:56.00   1st Qu.:  950   1st Qu.: 4.710
##  Median :61.80   Median :57.00   Median : 2401   Median : 5.700
##  Mean   :61.75   Mean   :57.46   Mean   : 3933   Mean   : 5.731
##  3rd Qu.:62.50   3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540
##  Max.   :79.00   Max.   :95.00   Max.   :18823   Max.   :10.740
##
##        y               z
##  Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 4.720   1st Qu.: 2.910
##  Median : 5.710   Median : 3.530
##  Mean   : 5.735   Mean   : 3.539
##  3rd Qu.: 6.540   3rd Qu.: 4.040
##  Max.   :58.900   Max.   :31.800
##
```

```
str(diamonds_df)
```

```
## 'data.frame':    53940 obs. of  11 variables:
##  $ X      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ carat  : num  0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
##  $ cut    : Factor w/ 5 levels "Fair","Good",..: 3 4 2 4 2 5 5 5 1 5 ...
##  $ color  : Factor w/ 7 levels "D","E","F","G",..: 2 2 2 6 7 7 6 5 2 5 ...
##  $ clarity: Factor w/ 8 levels "I1","IF","SI1",..: 4 3 5 6 4 8 7 3 6 5 ...
##  $ depth  : num  61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
##  $ table  : num  55 61 65 58 58 57 57 55 61 61 ...
##  $ price  : int  326 326 327 334 335 336 336 337 337 338 ...
```

```
##  $ x       : num  3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
##  $ y       : num  3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
##  $ z       : num  2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
```

La lectura del fichero y el resumen de estadísticos de las variables nos muestra que el conjunto de datos consta de 53940 registros y 11 variables o columnas.

La función de importación del fichero csv ha asignado correctamente cada variable con su tipo de datos, los que tienen decimales son números, los que son cualitativos se han importado como factores y el precio es un número entero

De todas ellas la variable X no aporta nada, ya que contiene el número o identificador de registro, por ello, la eliminaremos del conjunto de datos

```
diamonds_df<-diamonds_df[,-1]
```

Consideramos que necesitamos una variable que contenga el volumen de los diamantes, basándonos en sus medidas x, y, z

```
volume<-diamonds_df$x*diamonds_df$y*diamonds_df$z
diamonds_df<-cbind(diamonds_df,volume)
```

3. Limpieza de los datos.

3.1. ¿Los datos contienen ceros o elementos vacíos? ¿Cómo gestionarías cada uno de estos casos?

```
#Comprobamos si hay registros con elementos vacíos o NA
sapply(diamonds_df, anyNA)
```

```
##   carat     cut   color clarity   depth   table   price       x       y       z
##   FALSE   FALSE   FALSE   FALSE   FALSE   FALSE   FALSE   FALSE   FALSE   FALSE
##  volume
##   FALSE
```

Hemos comprobado que no hay ningún registro con elementos vacíos

```
#Comprobamos si hay registros que contienen ceros
sapply(diamonds_df, function(x) any(x==0))
```

```
##   carat     cut   color clarity   depth   table   price       x       y       z
##   FALSE   FALSE   FALSE   FALSE   FALSE   FALSE   FALSE    TRUE    TRUE    TRUE
##  volume
##    TRUE
```

Hemos comprobado que las variables volumen, x, y, z contienen ceros

Un diamante es un objeto que debe tener ancho, altura y profundidad, por lo tanto, asumimos que si alguna de estas variables son 0, es que no se ha medido o hay un error en los datos.

Este tipo de problemas, tiene 2 soluciones: Podríamos eliminar estos registros o calcular un nuevo valor según la semejanza de este registro con otros dentro del mismo conjunto de datos.

Creemos que eliminar estos registros limitaría nuestro conjunto de datos y ocultaría información que queremos analizar, por ello, imputaremos los valores perdidos a través del modelo basado en los k-vecinos más cercanos.

El modelo Knn encuentra los k vecinos más cercanos según la semejanza de los registros perdidos con los demás registros del juego de datos

Solo imputaremos el valor en las variables x, y, z, no tiene sentido aplicar este método a la variable volumen, ya que, es un campo calculado, lo que haremos será que después de calcular los nuevos valores, volveremos a calcular el volumen

```
suppressWarnings(suppressMessages(library(VIM)))
library(VIM)
#Miramos que registros contienen ceros
indicesx <-which(diamonds_df$x==0)
indicesy <-which(diamonds_df$y==0)
indicesz <-which(diamonds_df$z==0)
#Asignamos el valor de NA a todos los campos que contienen ceros
diamonds_df[indicesx,"x"]<-NA
diamonds_df[indicesy,"y"]<-NA
diamonds_df[indicesz,"z"]<-NA
```

Miramos el estado de los registros que contienen ceros en la variable x, antes de la imputación

```
diamonds_df[indicesx,]
```

```
##       carat       cut color clarity depth table price  x    y  z volume
## 11183  1.07     Ideal     F     SI2  61.6    56  4954 NA 6.62 NA      0
## 11964  1.00 Very Good     H     VS2  63.3    53  5139 NA   NA NA      0
## 15952  1.14      Fair     G     VS1  57.5    67  6381 NA   NA NA      0
## 24521  1.56     Ideal     G     VS2  62.2    54 12800 NA   NA NA      0
## 26244  1.20   Premium     D    VVS1  62.1    59 15686 NA   NA NA      0
## 27430  2.25   Premium     H     SI2  62.8    59 18034 NA   NA NA      0
## 49557  0.71      Good     F     SI2  64.1    60  2130 NA   NA NA      0
## 49558  0.71      Good     F     SI2  64.1    60  2130 NA   NA NA      0
```

```
kNNx<-kNN(diamonds_df[,c("cut","color","clarity","depth","table","price","x","y","z")], variable="x",k=
diamonds_df[indicesx,"x"]<-kNNx[indicesx,"x"]
```

Después de la imputación en la variable x

```
diamonds_df[indicesx,]
```

```
##       carat       cut color clarity depth table price    x    y  z volume
## 11183  1.07     Ideal     F     SI2  61.6    56  4954 6.50 6.62 NA      0
## 11964  1.00 Very Good     H     VS2  63.3    53  5139 6.61   NA NA      0
## 15952  1.14      Fair     G     VS1  57.5    67  6381 6.32   NA NA      0
## 24521  1.56     Ideal     G     VS2  62.2    54 12800 7.34   NA NA      0
## 26244  1.20   Premium     D    VVS1  62.1    59 15686 7.01   NA NA      0
## 27430  2.25   Premium     H     SI2  62.8    59 18034 8.42   NA NA      0
## 49557  0.71      Good     F     SI2  64.1    60  2130 6.50   NA NA      0
## 49558  0.71      Good     F     SI2  64.1    60  2130 6.50   NA NA      0
```

Miramos el estado de los registros que contienen ceros en la variable y, antes de la imputación

```
diamonds_df[indicesy,]
```

```
##       carat       cut color clarity depth table price    x  y  z volume
## 11964  1.00 Very Good     H     VS2  63.3    53  5139 6.61 NA NA      0
## 15952  1.14      Fair     G     VS1  57.5    67  6381 6.32 NA NA      0
## 24521  1.56     Ideal     G     VS2  62.2    54 12800 7.34 NA NA      0
## 26244  1.20   Premium     D    VVS1  62.1    59 15686 7.01 NA NA      0
## 27430  2.25   Premium     H     SI2  62.8    59 18034 8.42 NA NA      0
## 49557  0.71      Good     F     SI2  64.1    60  2130 6.50 NA NA      0
## 49558  0.71      Good     F     SI2  64.1    60  2130 6.50 NA NA      0
```

```
kNNy<-kNN(diamonds_df[,c("cut","color","clarity","depth","table","price","x","y","z")], variable="y",k=3
diamonds_df[indicesy,"y"]<-kNNy[indicesy,"y"]
```

Después de la imputación en la variable y

```
diamonds_df[indicesy,]
```

```
##       carat       cut color clarity depth table price    x    y  z volume
## 11964  1.00 Very Good     H     VS2  63.3    53  5139 6.61 6.54 NA      0
## 15952  1.14      Fair     G     VS1  57.5    67  6381 6.32 6.19 NA      0
## 24521  1.56     Ideal     G     VS2  62.2    54 12800 7.34 7.31 NA      0
## 26244  1.20   Premium     D    VVS1  62.1    59 15686 7.01 6.88 NA      0
## 27430  2.25   Premium     H     SI2  62.8    59 18034 8.42 8.37 NA      0
## 49557  0.71      Good     F     SI2  64.1    60  2130 6.50 6.47 NA      0
## 49558  0.71      Good     F     SI2  64.1    60  2130 6.50 6.47 NA      0
```

Miramos el estado de los registros que contienen ceros en la variable z, antes de la imputación

```
diamonds_df[indicesz,]
```

```
##       carat       cut color clarity depth table price    x    y  z volume
## 2208   1.00   Premium     G     SI2  59.1    59  3142 6.55 6.48 NA      0
## 2315   1.01   Premium     H      I1  58.1    59  3167 6.66 6.60 NA      0
## 4792   1.10   Premium     G     SI2  63.0    59  3696 6.50 6.47 NA      0
## 5472   1.01   Premium     F     SI2  59.2    58  3837 6.50 6.47 NA      0
## 10168  1.50      Good     G      I1  64.0    61  4731 7.15 7.04 NA      0
## 11183  1.07     Ideal     F     SI2  61.6    56  4954 6.50 6.62 NA      0
## 11964  1.00 Very Good     H     VS2  63.3    53  5139 6.61 6.54 NA      0
## 13602  1.15     Ideal     G     VS2  59.2    56  5564 6.88 6.83 NA      0
## 15952  1.14      Fair     G     VS1  57.5    67  6381 6.32 6.19 NA      0
## 24395  2.18   Premium     H     SI2  59.4    61 12631 8.49 8.45 NA      0
## 24521  1.56     Ideal     G     VS2  62.2    54 12800 7.34 7.31 NA      0
## 26124  2.25   Premium     I     SI1  61.3    58 15397 8.52 8.42 NA      0
## 26244  1.20   Premium     D    VVS1  62.1    59 15686 7.01 6.88 NA      0
## 27113  2.20   Premium     H     SI1  61.2    59 17265 8.42 8.37 NA      0
## 27430  2.25   Premium     H     SI2  62.8    59 18034 8.42 8.37 NA      0
## 27504  2.02   Premium     H     VS2  62.7    53 18207 8.02 7.95 NA      0
## 27740  2.80      Good     G     SI2  63.8    58 18788 8.90 8.85 NA      0
## 49557  0.71      Good     F     SI2  64.1    60  2130 6.50 6.47 NA      0
## 49558  0.71      Good     F     SI2  64.1    60  2130 6.50 6.47 NA      0
## 51507  1.12   Premium     G      I1  60.4    59  2383 6.71 6.67 NA      0
```

```
kNNz<-kNN(diamonds_df[,c("cut","color","clarity","depth","table","price","x","y","z")], variable="z",k=
diamonds_df[indicesz,"z"]<-kNNz[indicesz,"z"]
```

Después de la imputación en la variable z

```
diamonds_df[indicesz,]
```

```
##       carat       cut color clarity depth table price    x    y    z volume
## 2208   1.00   Premium     G     SI2  59.1    59  3142 6.55 6.48 3.86      0
## 2315   1.01   Premium     H      I1  58.1    59  3167 6.66 6.60 4.05      0
## 4792   1.10   Premium     G     SI2  63.0    59  3696 6.50 6.47 4.05      0
## 5472   1.01   Premium     F     SI2  59.2    58  3837 6.50 6.47 3.82      0
## 10168  1.50      Good     G      I1  64.0    61  4731 7.15 7.04 4.63      0
## 11183  1.07     Ideal     F     SI2  61.6    56  4954 6.50 6.62 3.99      0
## 11964  1.00 Very Good     H     VS2  63.3    53  5139 6.61 6.54 4.14      0
## 13602  1.15     Ideal     G     VS2  59.2    56  5564 6.88 6.83 3.96      0
## 15952  1.14      Fair     G     VS1  57.5    67  6381 6.32 6.19 3.79      0
## 24395  2.18   Premium     H     SI2  59.4    61 12631 8.49 8.45 5.09      0
## 24521  1.56     Ideal     G     VS2  62.2    54 12800 7.34 7.31 4.55      0
## 26124  2.25   Premium     I     SI1  61.3    58 15397 8.52 8.42 5.02      0
## 26244  1.20   Premium     D    VVS1  62.1    59 15686 7.01 6.88 4.16      0
## 27113  2.20   Premium     H     SI1  61.2    59 17265 8.42 8.37 5.13      0
## 27430  2.25   Premium     H     SI2  62.8    59 18034 8.42 8.37 5.23      0
## 27504  2.02   Premium     H     VS2  62.7    53 18207 8.02 7.95 5.06      0
## 27740  2.80      Good     G     SI2  63.8    58 18788 8.90 8.85 5.10      0
## 49557  0.71      Good     F     SI2  64.1    60  2130 6.50 6.47 3.83      0
## 49558  0.71      Good     F     SI2  64.1    60  2130 6.50 6.47 3.83      0
## 51507  1.12   Premium     G      I1  60.4    59  2383 6.71 6.67 4.02      0
```

calculamos nuevamente la variable volumen ya que, esta depende de los valores de x, y, z

```
diamonds_df$volume<-diamonds_df$x*diamonds_df$y*diamonds_df$z
```

Finalmente, volvemos a comprobar si hay registros ceros o elementos vacíos

```
#Comprobamos si hay registros con elementos vacíos o NA
sapply(diamonds_df, anyNA)
```

```
##   carat     cut   color clarity   depth   table   price       x       y       z
##   FALSE   FALSE   FALSE   FALSE   FALSE   FALSE   FALSE   FALSE   FALSE   FALSE
##  volume
##   FALSE
```

```
#Comprobamos si hay registros que contienen ceros
sapply(diamonds_df, function(x) any(x==0))
```

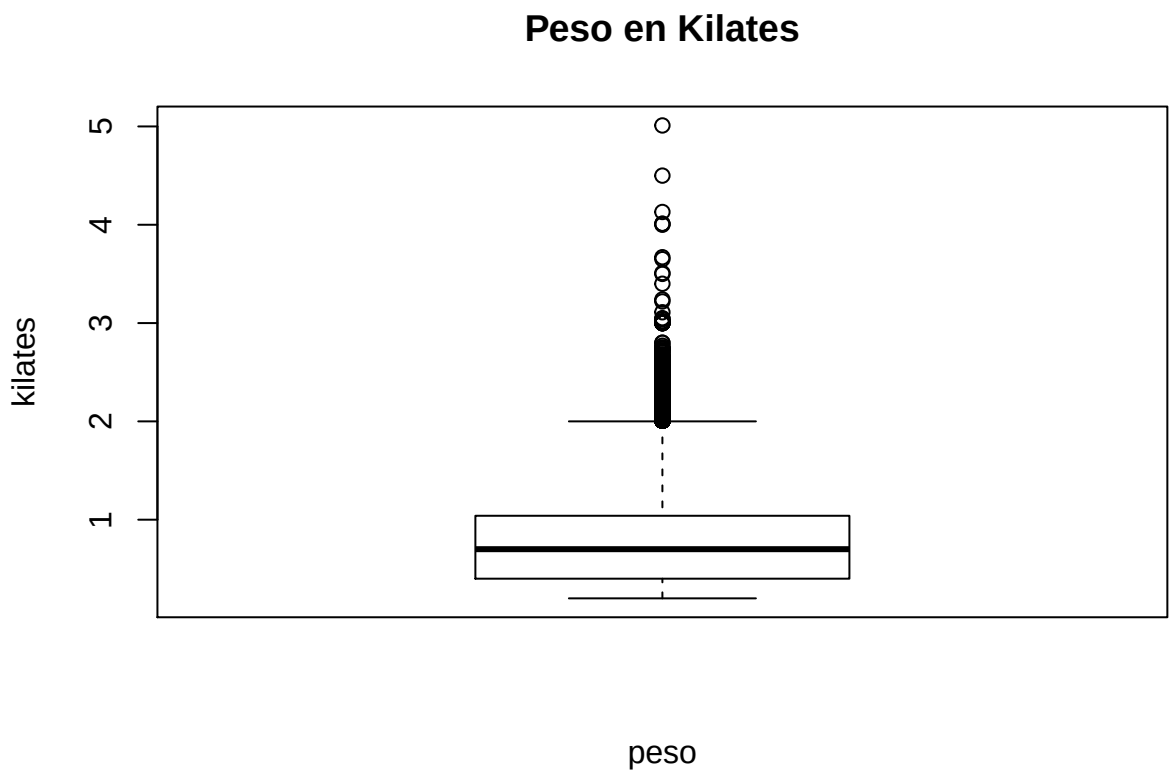
```
##   carat     cut   color clarity   depth   table   price       x       y       z
##   FALSE   FALSE   FALSE   FALSE   FALSE   FALSE   FALSE   FALSE   FALSE   FALSE
##  volume
##   FALSE
```

Efectivamente, ya no hay ningún valor cero ni elementos vacío

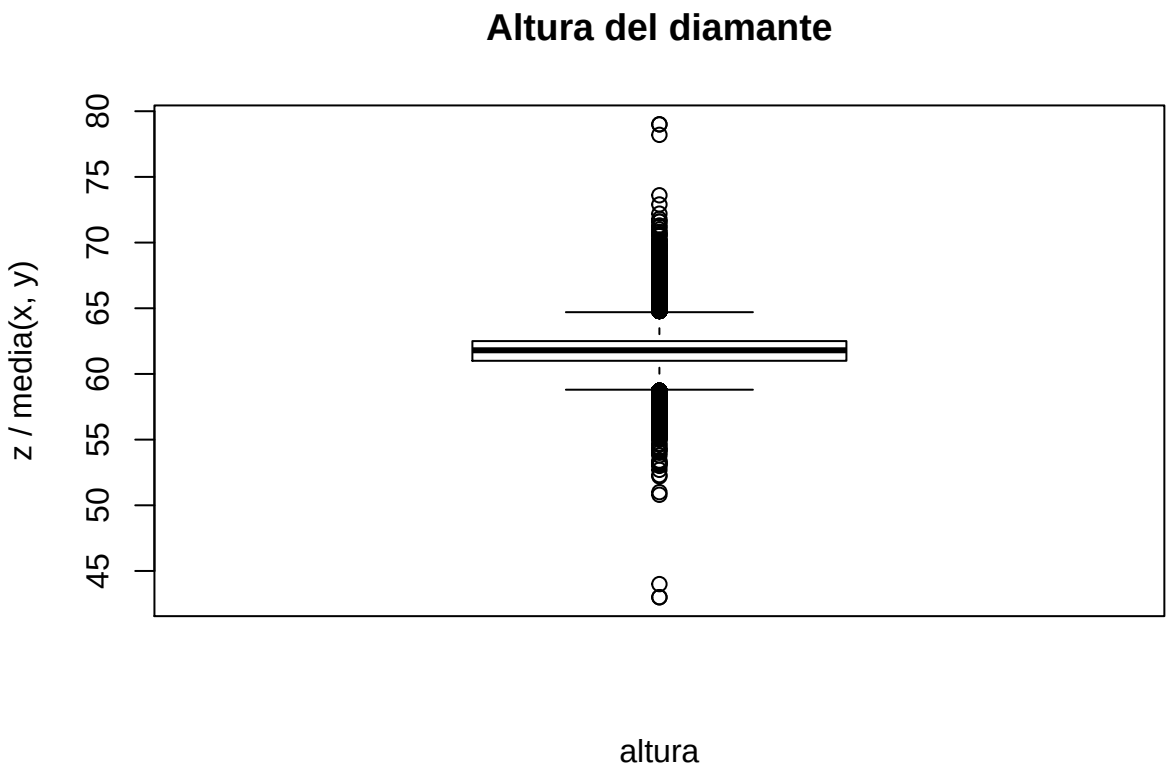
3.2 Identificación y tratamiento de valores extremos.

En primer lugar, en esta categoría analizaremos sólo los valores numéricos, para ello, vamos a realizar representaciones de tipo caja de cada una de las variables.

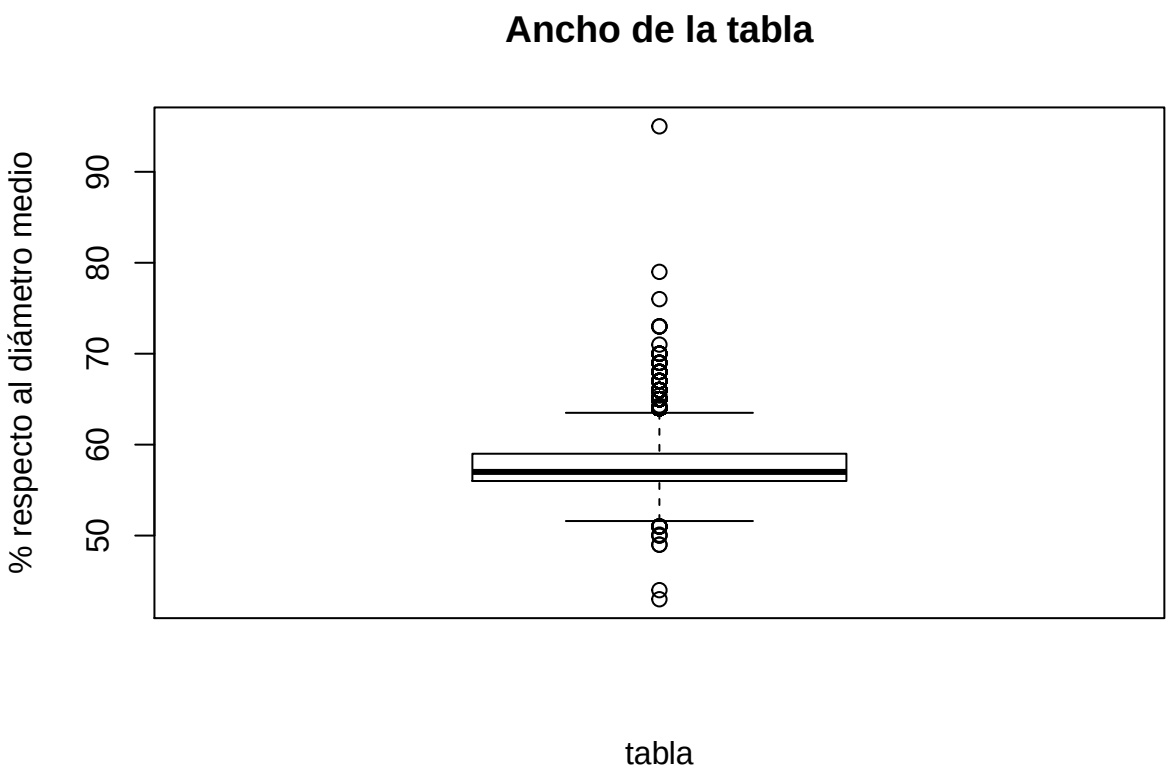
```
#Representamos cada variable
boxplot(diamonds_df$carat, main="Peso en Kilates", xlab="peso", ylab="kilates")
```



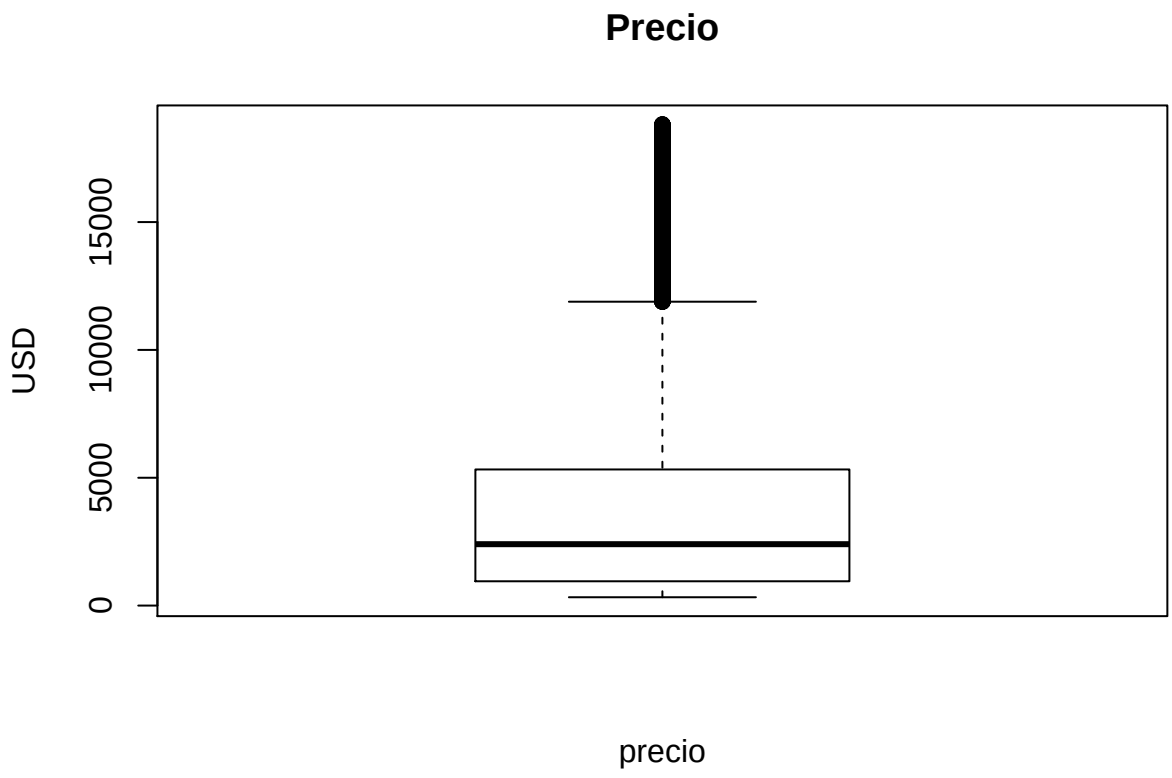

```
boxplot(diamonds_df$depth, main="Altura del diamante", xlab="altura", ylab="z / media(x, y)")
```



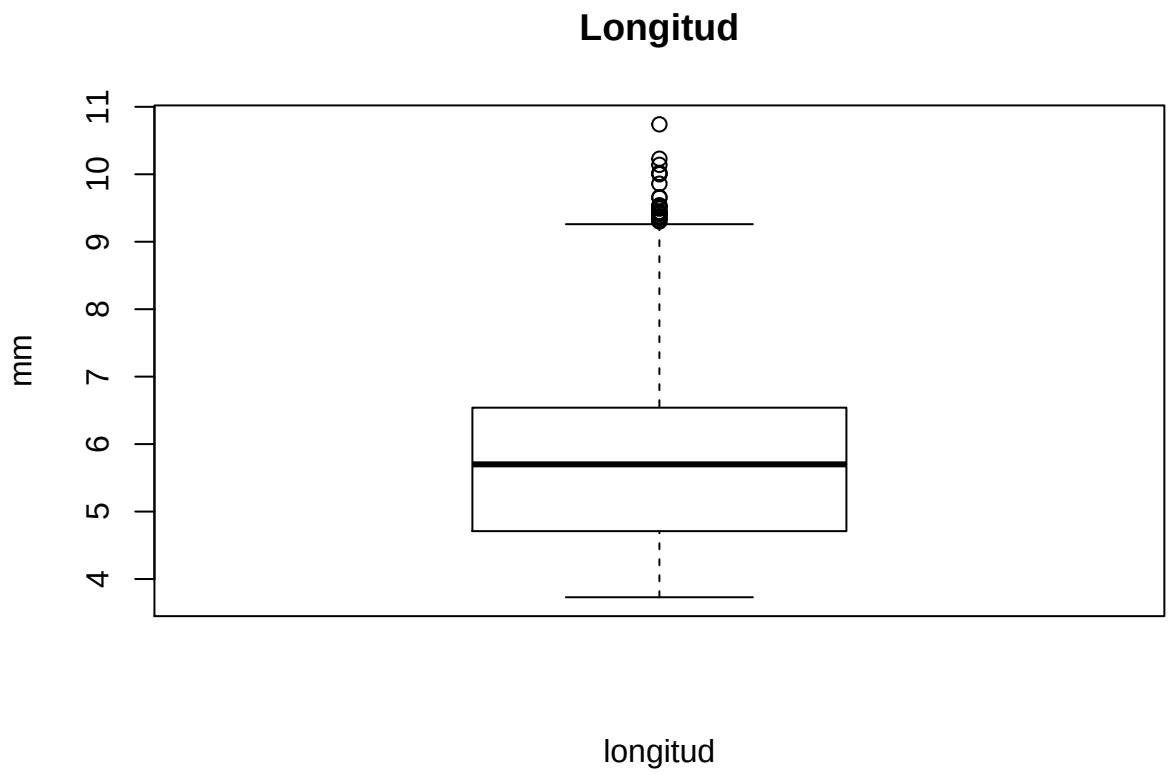

```
boxplot(diamonds_df$table, main="Ancho de la tabla", xlab="tabla", ylab="% respecto al diámetro medio")
```



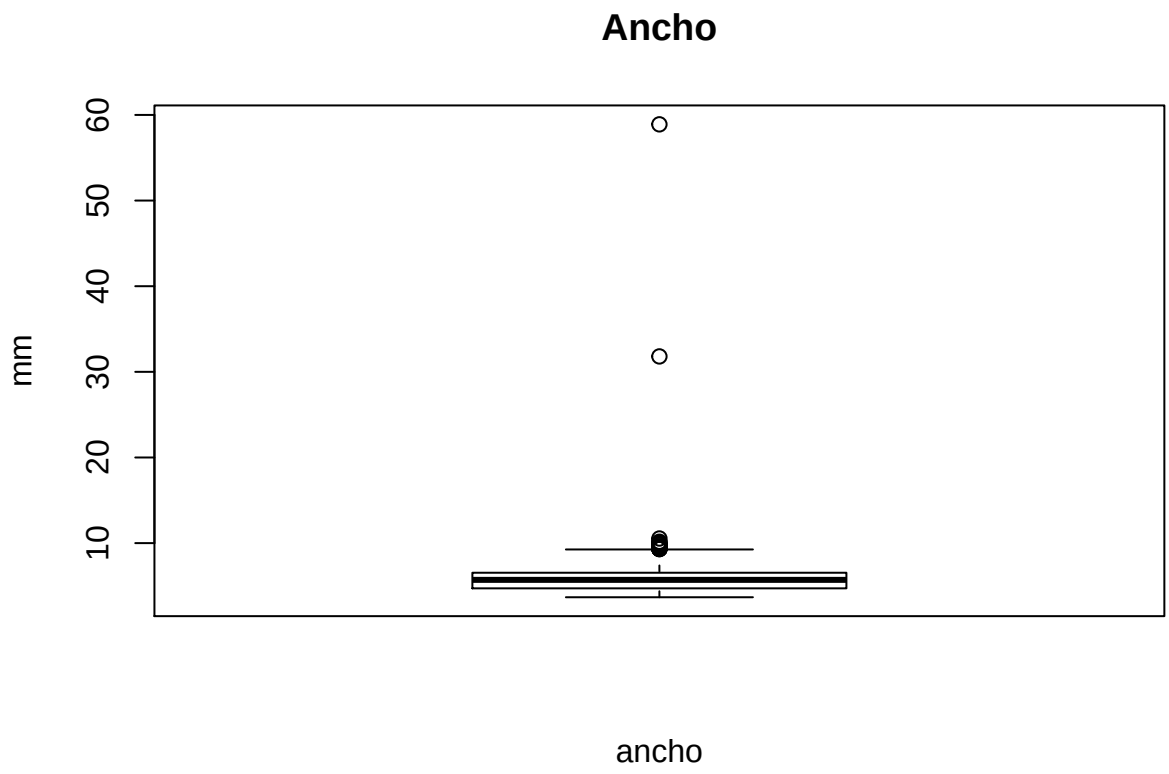

```
boxplot(diamonds_df$price, main="Precio", xlab="precio", ylab="USD")
```



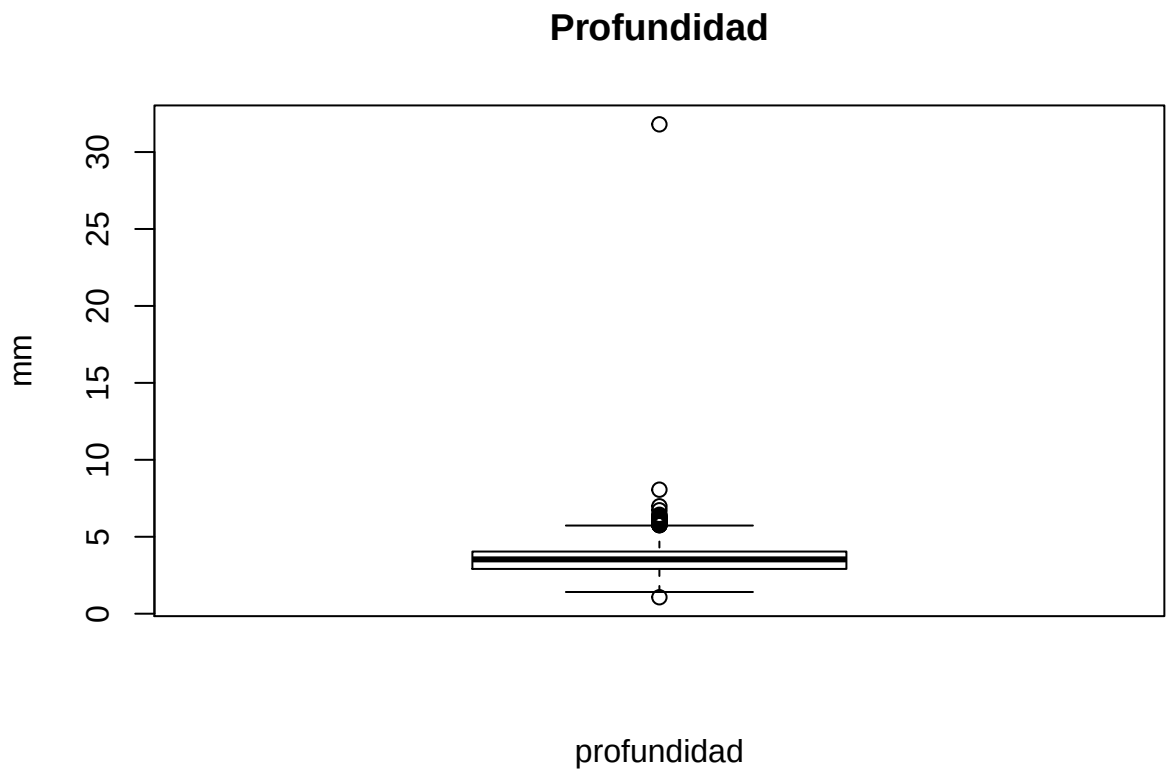

```
boxplot(diamonds_df$x, main="Longitud", xlab="longitud", ylab="mm")
```



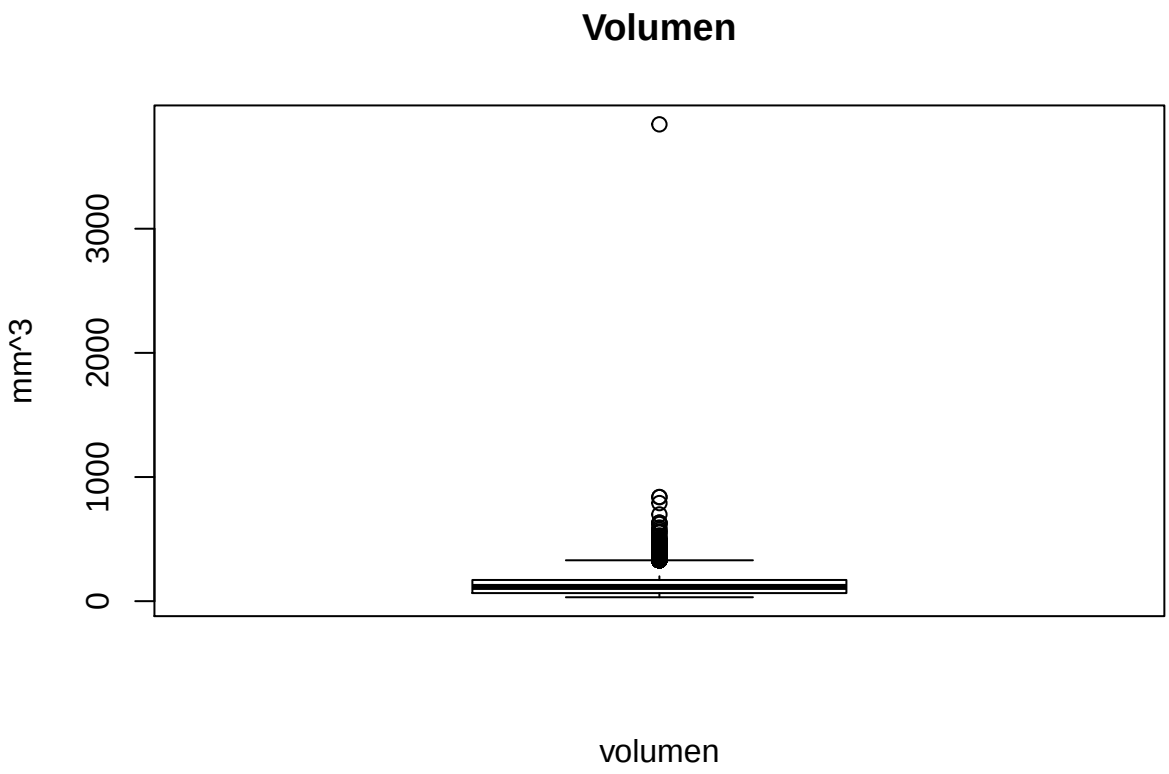

```
boxplot(diamonds_df$y, main="Ancho", xlab="ancho", ylab="mm")
```




```
boxplot(diamonds_df$z, main="Profundidad", xlab="profundidad", ylab="mm")
```




```
boxplot(diamonds_df$volume, main="Volumen", xlab="volumen", ylab="mm^3")
```



Observando los datos, vemos que existen datos que, claramente, suponen valores extremos que no difieren bastante de la media de la muestra, analizamos caso a caso.

4. Análisis de los datos.

4.1. Selección de los grupos de datos que se quieren analizar/comparar (planificación de los análisis a aplicar).

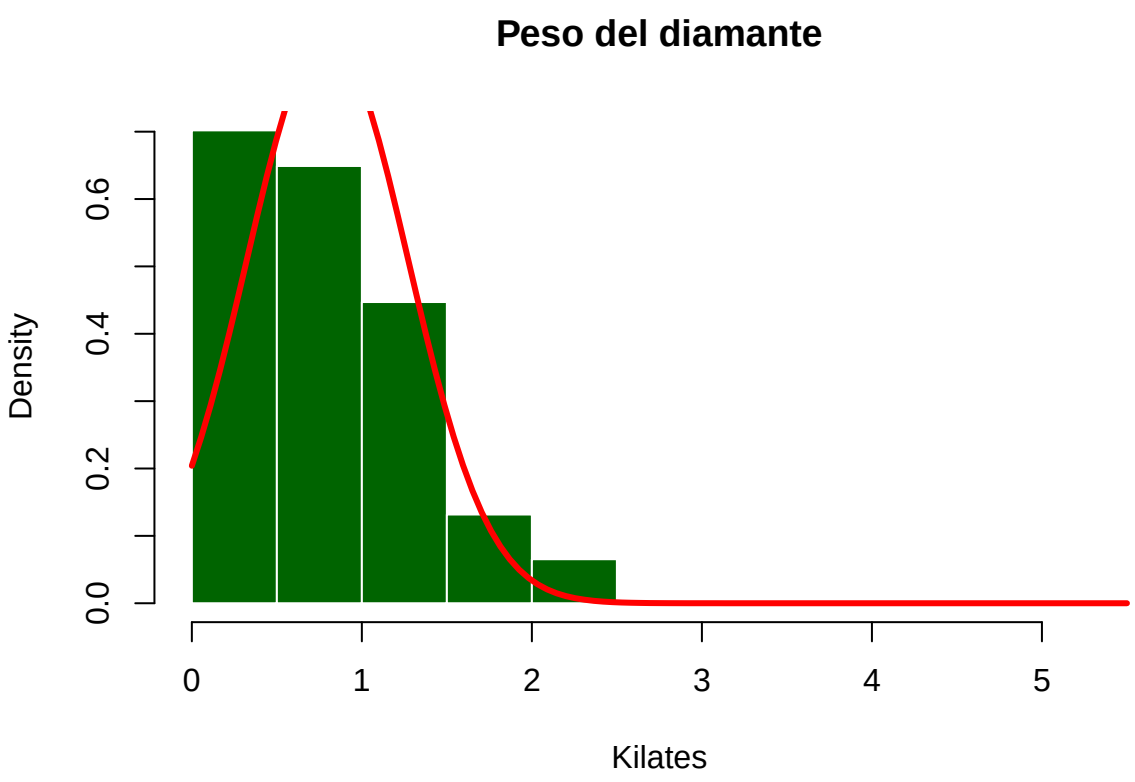
4.2. Comprobación de la normalidad y homogeneidad de la varianza.

En primer lugar, se realiza un estudio sobre la normalidad de las variables precio, altura, tabla, x, y, z y volumen.

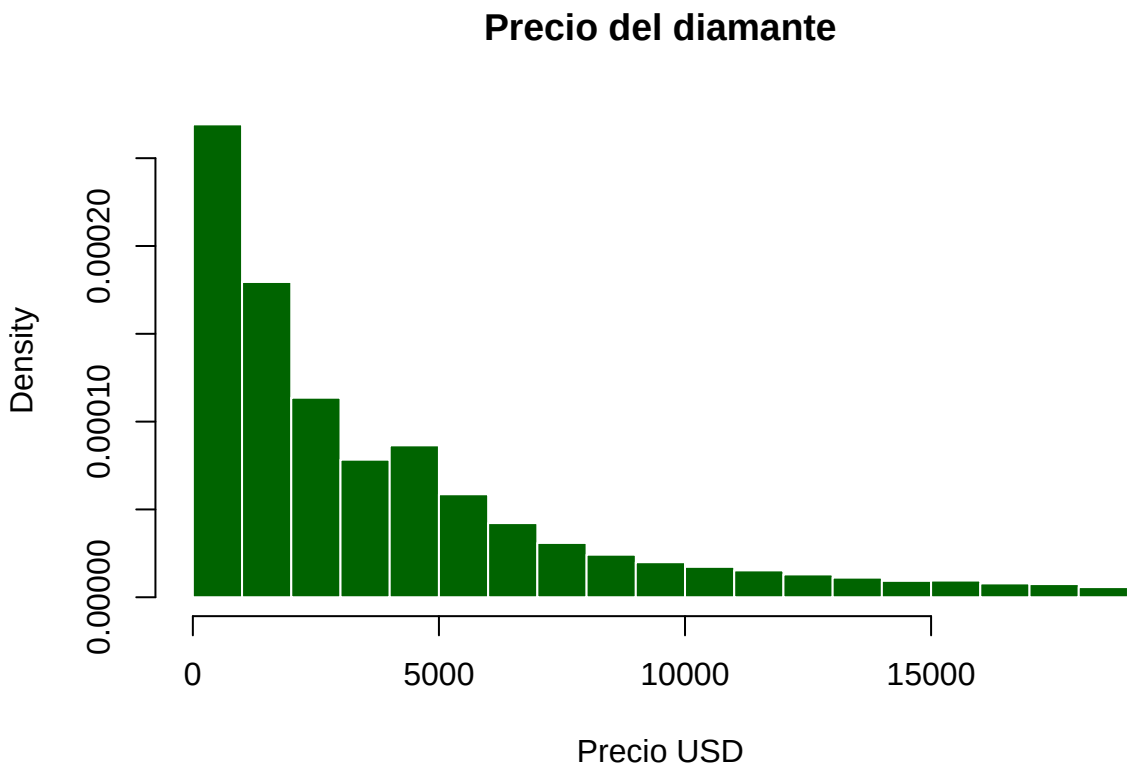
Se realizará un estudio visual y, a continuación, un test Shapiro (considerado de los más potentes para el contraste de la normalidad).

> En el estudio visual se utilizarán histogramas de frecuencias relativas con curva de normalidad superpuesta

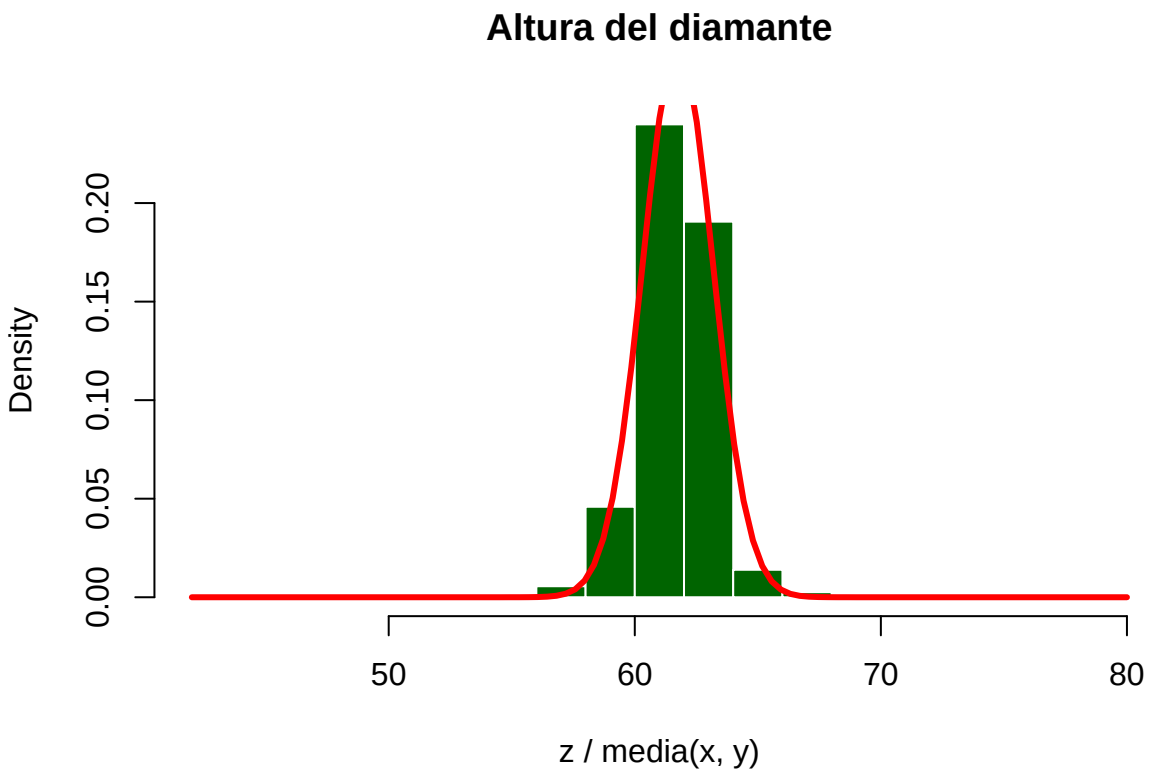
```
#Variable peso
hist(diamonds_df$carat, main= "Peso del diamante", freq = FALSE, xlab = "Kilates", col = "dark green", l
curve(dnorm(x, mean=mean(diamonds_df$carat), sd=sd(diamonds_df$carat)), add=TRUE, col="red", lwd=3)
```



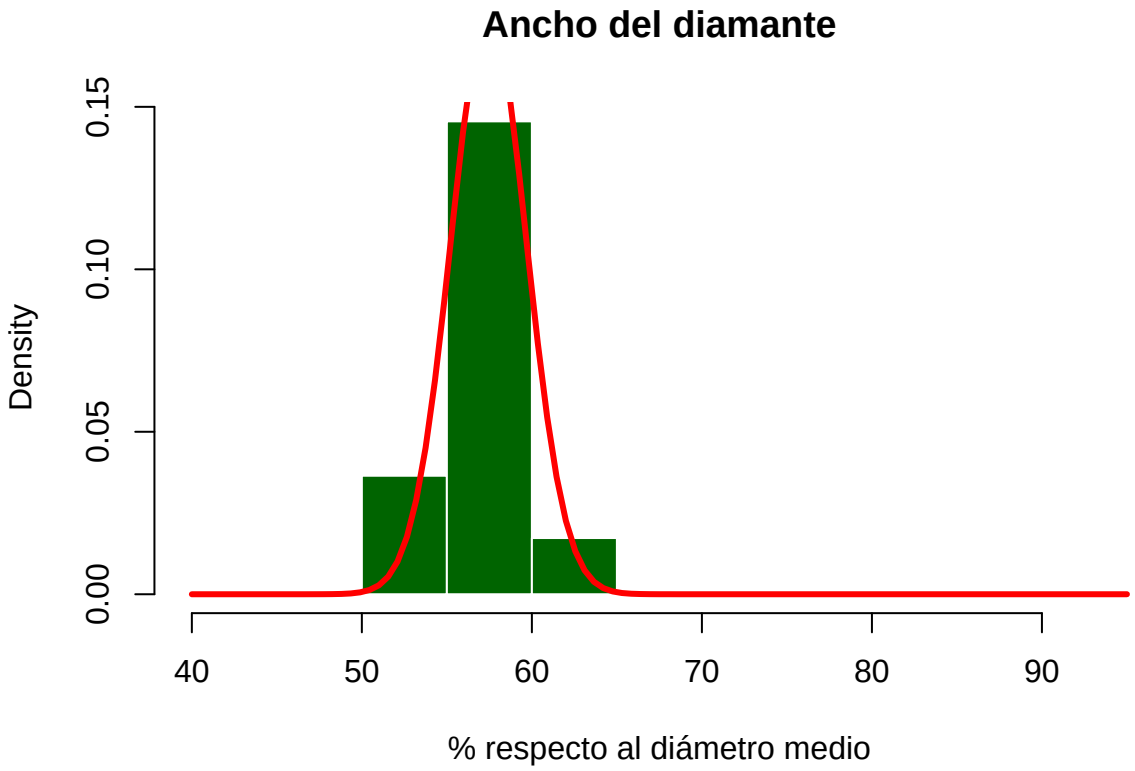

```
#Variable precio
hist(diamonds_df$price, main= "Precio del diamante", freq = FALSE, xlab = "Precio USD", col = "dark gre
```



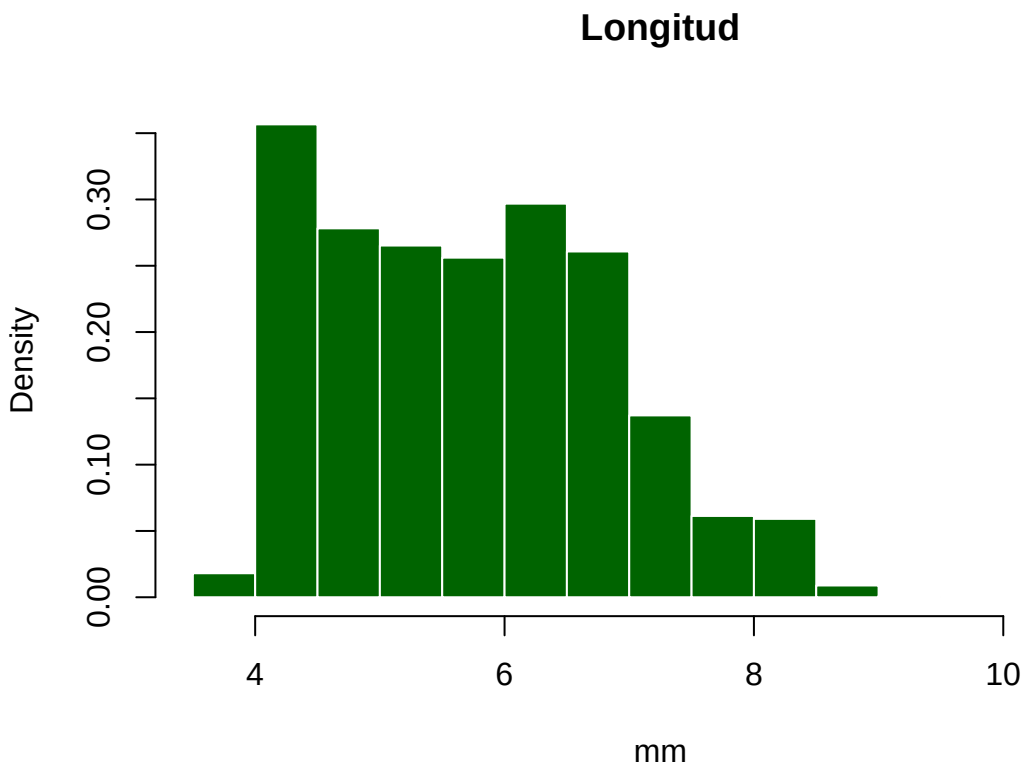

```
#Variable altura
hist(diamonds_df$depth, main= "Altura del diamante", freq = FALSE, xlab = "z / media(x, y)", col = "dark
curve(dnorm(x, mean=mean(diamonds_df$depth), sd=sd(diamonds_df$depth)), add=TRUE, col="red", lwd=3)
```



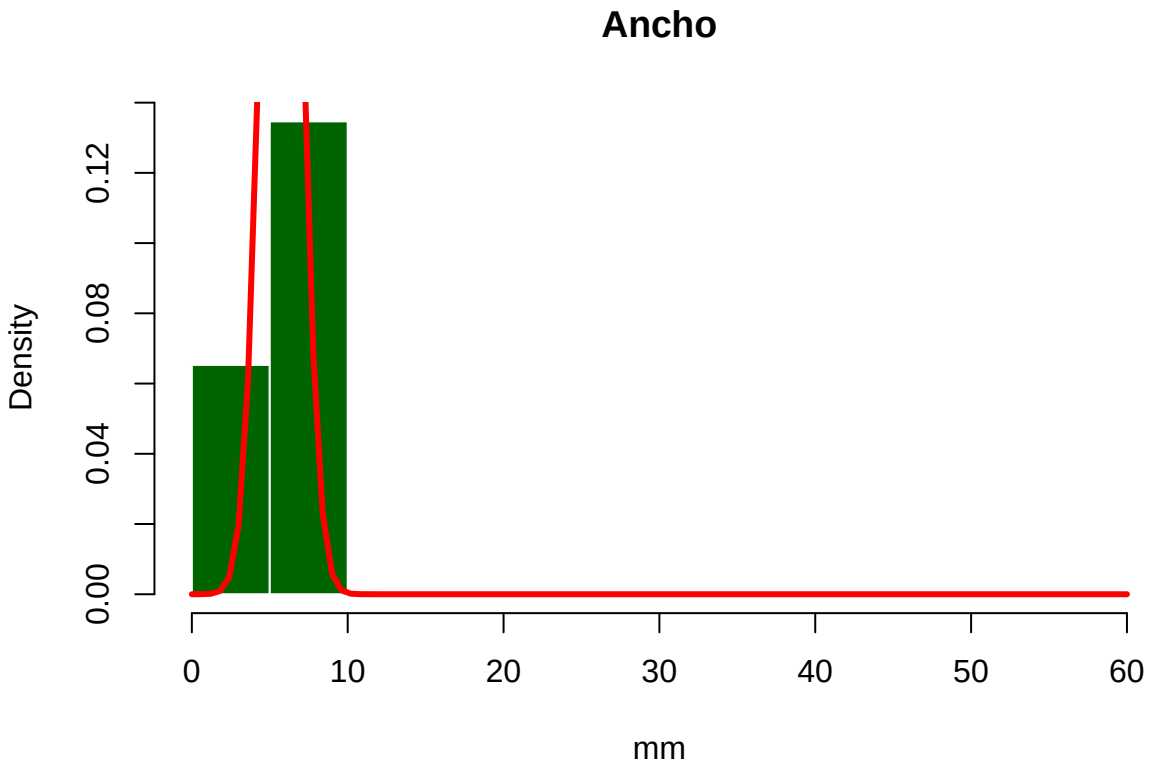

```
#Variable ancho
hist(diamonds_df$table, main= "Ancho del diamante", freq = FALSE, xlab = "% respecto al diámetro medio"
curve(dnorm(x, mean=mean(diamonds_df$table), sd=sd(diamonds_df$table)), add=TRUE, col="red", lwd=3)
```



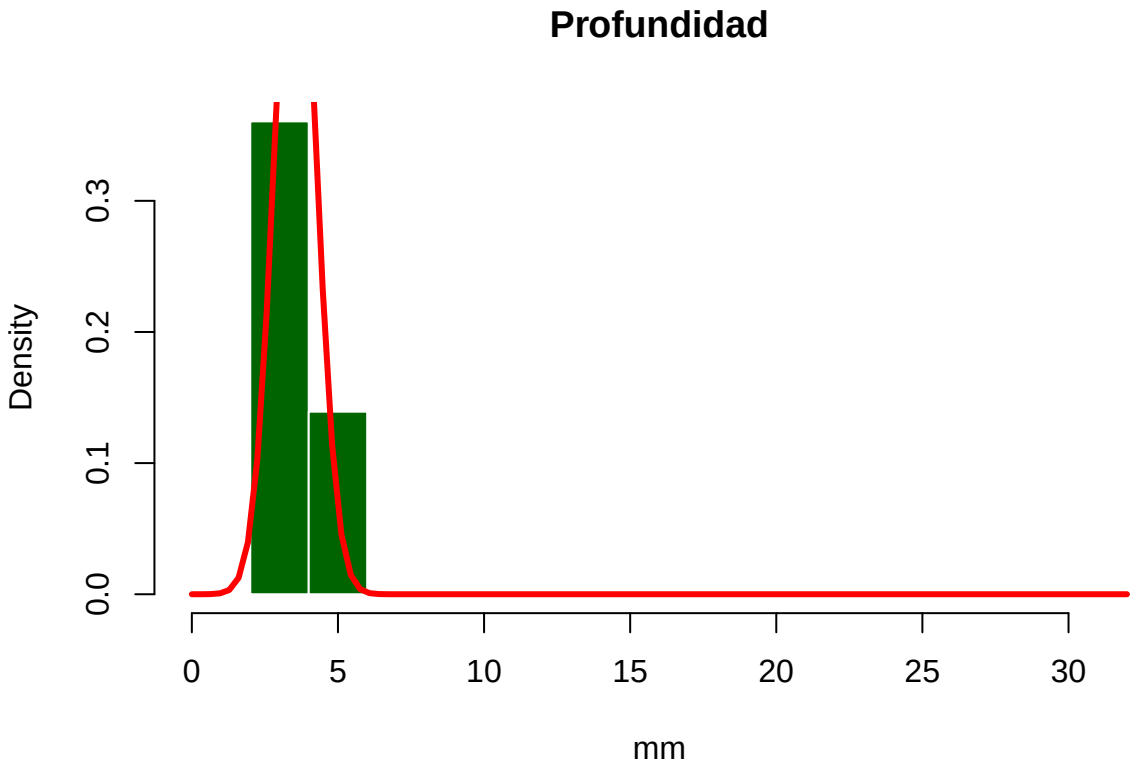

```
#Variable "x"
hist(diamonds_df$x, main= "Longitud", freq = FALSE, xlab = "mm", col = "dark green", border="white")
```



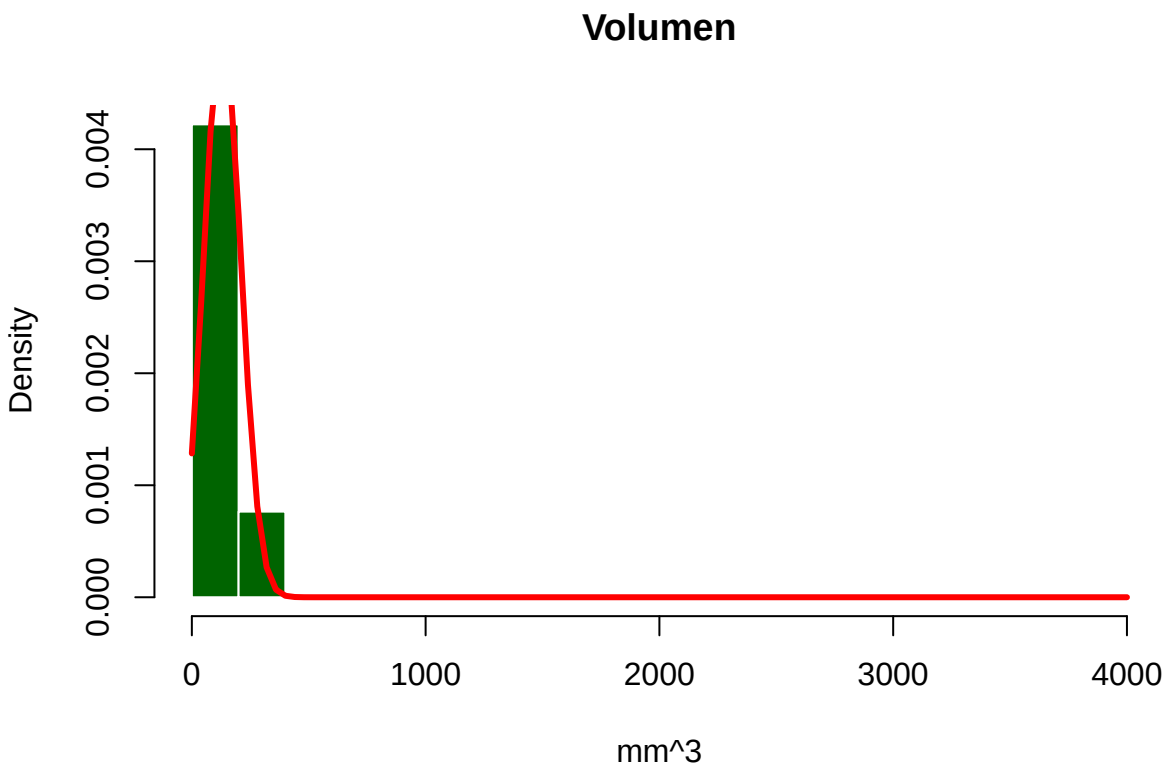

```
#Variable "y"
hist(diamonds_df$y, main= "Ancho", freq = FALSE, xlab = "mm", col = "dark green", border="white")
curve(dnorm(x, mean=mean(diamonds_df$y), sd=sd(diamonds_df$y)), add=TRUE, col="red", lwd=3)
```




```
#Variable "z"
hist(diamonds_df$z, main= "Profundidad", freq = FALSE, xlab = "mm", col = "dark green", border="white")
curve(dnorm(x, mean=mean(diamonds_df$z), sd=sd(diamonds_df$z)), add=TRUE, col="red", lwd=3)
```




```
#Volumen
hist(diamonds_df$volume, main= "Volumen", freq = FALSE, xlab = "mm^3", col = "dark green", border="whit
curve(dnorm(x, mean=mean(diamonds_df$volume), sd=sd(diamonds_df$volume)), add=TRUE, col="red", lwd=3)
```



Vemos que en el caso del precio seguiría una distribución exponencial negativa, por otra parte, la longitud no seguiría ninguna distribución apreciable.

Vamos a realizar el Test Shapiro a las variables que podrían asemejarse a una distribución normal, si atendemos a su gráfico.

```
shapiro.test(diamonds_df$carat[1:5000])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  diamonds_df$carat[1:5000]
## W = 0.90902, p-value < 2.2e-16
```

```
shapiro.test(diamonds_df$depth[1:5000])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  diamonds_df$depth[1:5000]
## W = 0.95416, p-value < 2.2e-16
```

```
shapiro.test(diamonds_df$table[1:5000])
```

```
##
```

```
##  Shapiro-Wilk normality test
##
## data:  diamonds_df$table[1:5000]
## W = 0.95047, p-value < 2.2e-16
```

```
shapiro.test(diamonds_df$y[1:5000])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  diamonds_df$y[1:5000]
## W = 0.82575, p-value < 2.2e-16
```

```
shapiro.test(diamonds_df$z[1:5000])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  diamonds_df$z[1:5000]
## W = 0.85949, p-value < 2.2e-16
```

```
shapiro.test(diamonds_df$volume[1:5000])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  diamonds_df$volume[1:5000]
## W = 0.9072, p-value < 2.2e-16
```

A continuación, vamos a comprobar si la varianza es homogénea, es decir, cumple con la homocedasticidad, para ello, ya que las muestras nos son normales, vamos a utilizar el test de Fligner-Killeen

Vamos a realizar un estudio de normalidad sobre la variable corte, para ello vamos a asignar un valor numérico a cada uno de los niveles de la variable categorica "cut".

```
table(diamonds_df$cut)
```

```
##
##      Fair      Good     Ideal   Premium Very Good
##      1610      4906     21551     13791     12082
```

Siendo estos valores:

- Fair: 1
- Good: 2
- Very Good: 3
- Premium: 4
- Ideal: 5

```r
level.cut<-diamonds_df$cut
level.cut[level.cut="Fair"]<-"1"
```

```
## Warning in `[<-.factor`(`*tmp*`, level.cut = "Fair", value = "1"): invalid
## factor level, NA generated
```

```r
level.cut[level.cut="Good"]<-"2"
```

```
## Warning in `[<-.factor`(`*tmp*`, level.cut = "Good", value = "2"): invalid
## factor level, NA generated
```

```r
level.cut[level.cut="Very Good"]<-"3"
```

```
## Warning in `[<-.factor`(`*tmp*`, level.cut = "Very Good", value = "3"): invalid
## factor level, NA generated
```

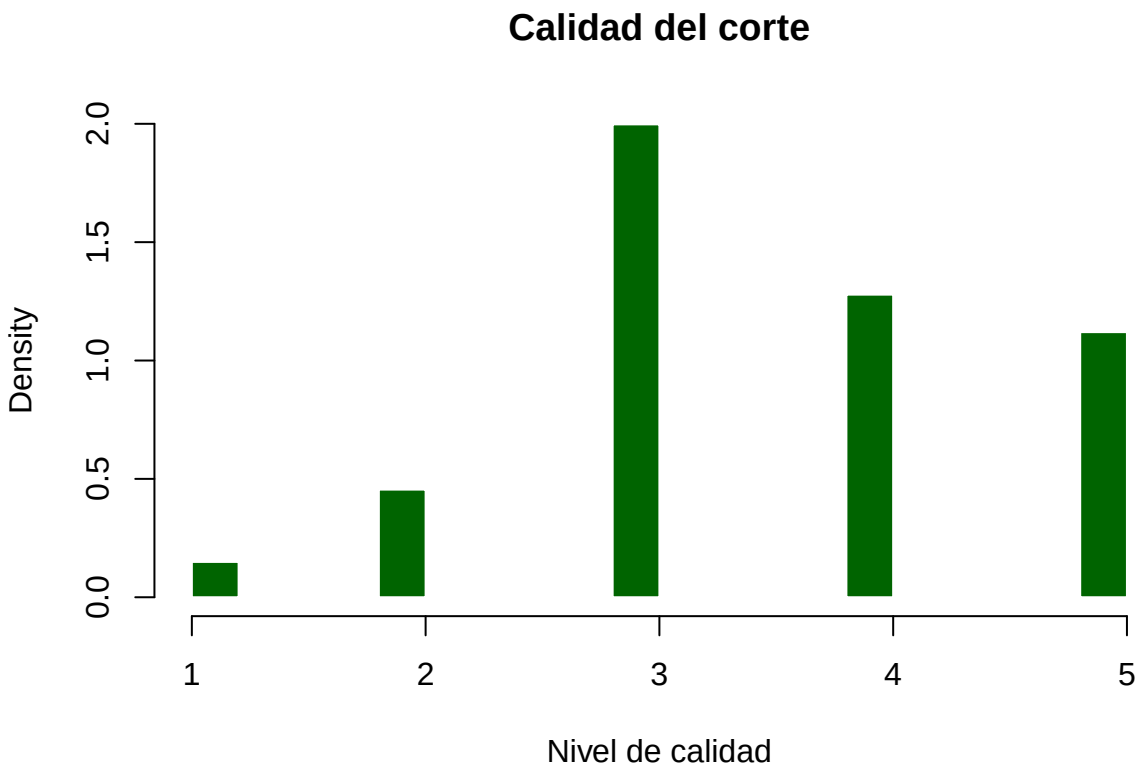
```r
level.cut[level.cut="Premium"]<-"4"
```

```
## Warning in `[<-.factor`(`*tmp*`, level.cut = "Premium", value = "4"): invalid
## factor level, NA generated
```

```r
level.cut[level.cut="Ideal"]<-"5"
```

```
## Warning in `[<-.factor`(`*tmp*`, level.cut = "Ideal", value = "5"): invalid
## factor level, NA generated
```

```r
level.cut<-as.numeric(level.cut)

#Variable corte
hist(level.cut, main= "Calidad del corte", freq = FALSE, xlab = "Nivel de calidad", col = "dark green",
```




```
#curve(dnorm(x, mean=mean(level.cut), sd=sd(level.cut)), add=TRUE, col="red", lwd=3)

#Test Shapiro
shapiro.test(level.cut[1:5000])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  level.cut[1:5000]
## W = 0.89184, p-value < 2.2e-16
```

4.3. Aplicación de pruebas estadísticas para comparar los grupos de datos. En función de los datos y el objetivo del estudio, aplicar pruebas de contraste de hipótesis, correlaciones, regresiones, etc. Aplicar al menos tres métodos de análisis diferentes.

4.3.1. Análisis de correlaciones

A continuación, vamos a realizar los análisis de correlaciones, para ello, vamos a utilizar el precio como principal indicador, correlacionando éste con todas las variables, numéricas y categoricas, para las numéricas realizaremos un análisis de correlación Pearson, para las categóricas realizaremos el test chi-square

Variables Numéricas

- Precio
- Peso

- Profundidad
- Tabla
- Volumen

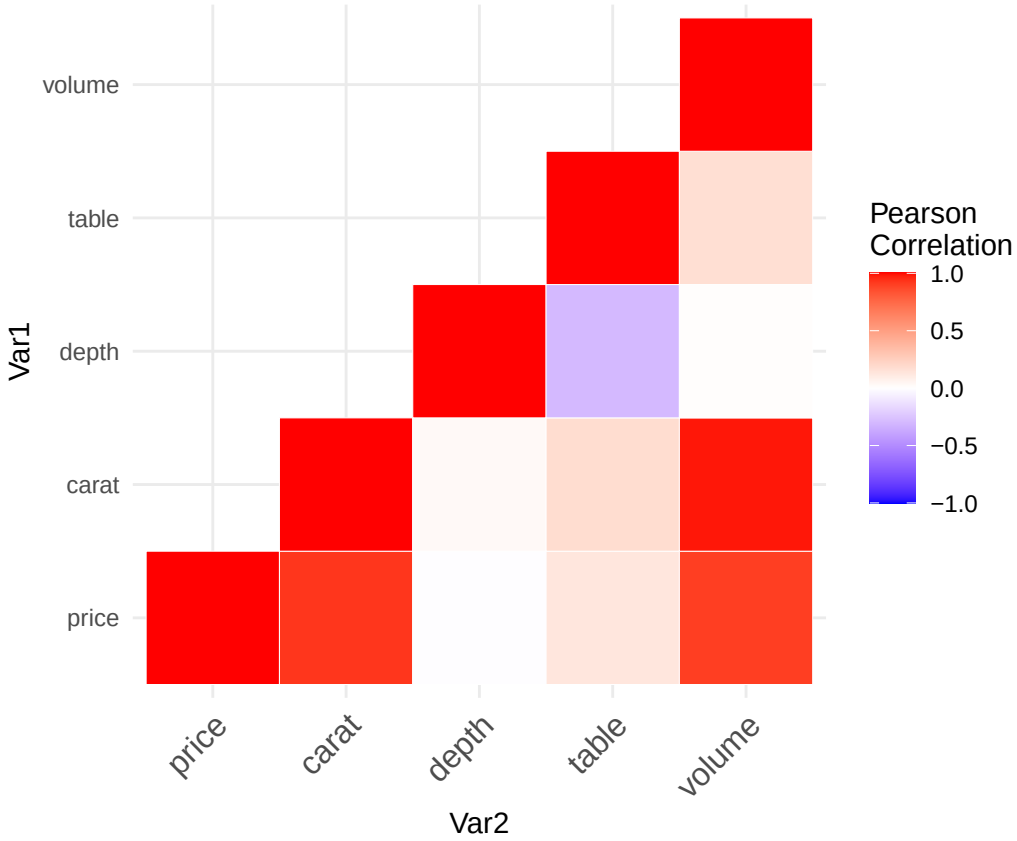
```
library (reshape2, warn.conflicts = FALSE)
```

```
## Warning: package 'reshape2' was built under R version 3.6.3
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```
#Creamos una matriz de correlación con los datos descritos
corr.mat<-round(cor(diamonds_df[,c(7,1,5,6,11)], method = "pearson"),2)

#Visualizamos los datos mediante un mapa de calor
corr.mat[lower.tri(corr.mat)]<- NA

m.corr.mat<-melt(corr.mat, na.rm=TRUE)

# Fuente: http://www.sthda.com/english/wiki/ggplot2-quick-correlation-matrix-heatmap-r-software-and-dat
ggplot(data = m.corr.mat, aes(Var2, Var1, fill = value))+
 geom_tile(color = "white")+
 scale_fill_gradient2(low = "blue", high = "red", mid = "white",
   midpoint = 0, limit = c(-1,1), space = "Lab",
   name="Pearson\nCorrelation") +
  theme_minimal()+
 theme(axis.text.x = element_text(angle = 45, vjust = 1,
    size = 12, hjust = 1))+
 coord_fixed()
```



Vemos que las variables más correlacionadas con el precio son el peso en kilates y el volumen del diamante. Para estas dos variables, vamos a estudiar una a una su índice de correlación:

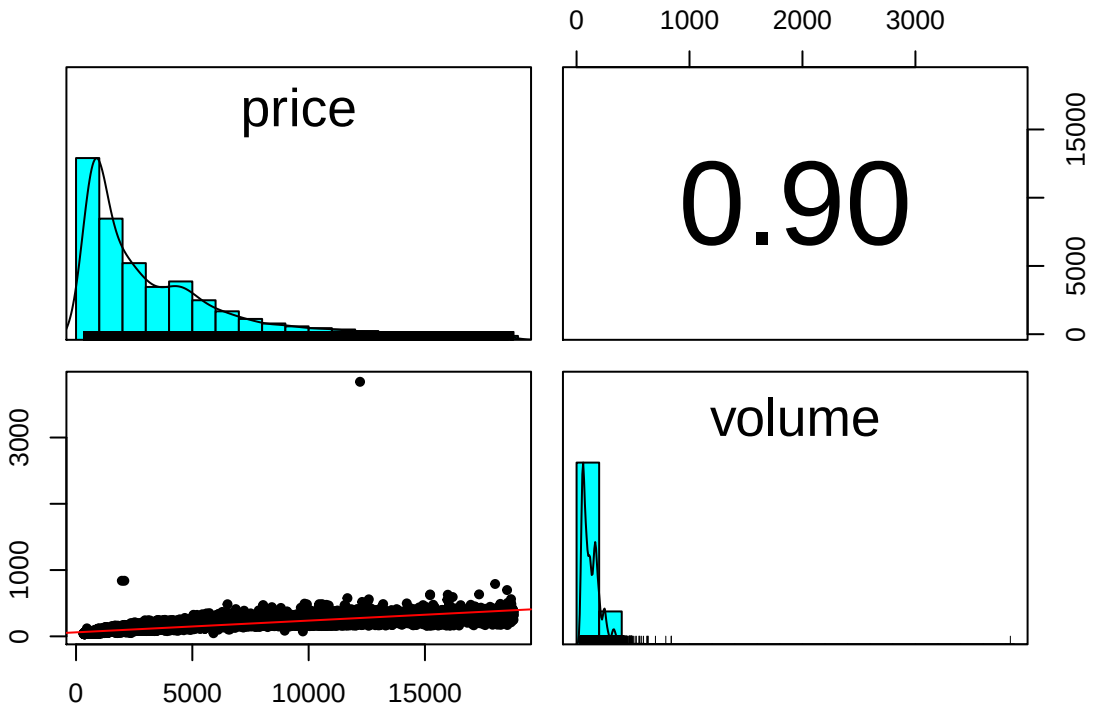
```
library(psych)
```

```
## Warning: package 'psych' was built under R version 3.6.3
```

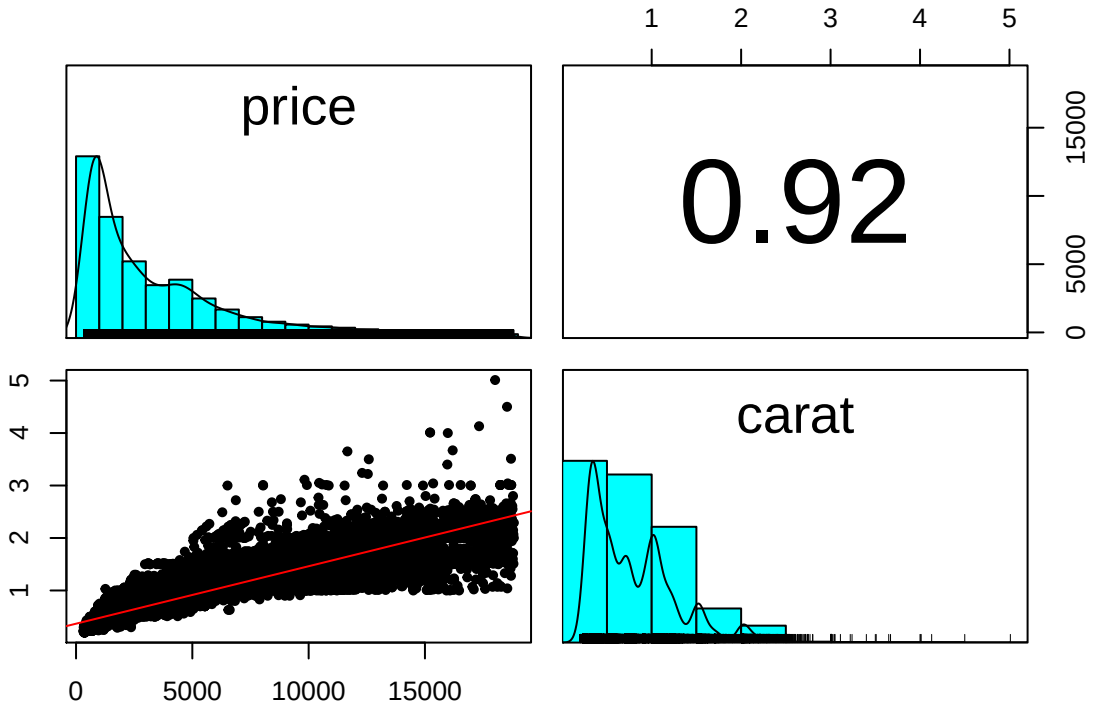
```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
pairs.panels(diamonds_df[,c(7,11)], ellipses = FALSE, lm=TRUE, method = "pearson")
```




```
pairs.panels(diamonds_df[,c(7,1)], ellipses = FALSE, lm=TRUE, method = "pearson")
```



Vemos que la correlación entre el Precio y el Peso, y entre el Precio y su volumen es bastante alta.

Variables Categóricas

A continuación, vamos a realizar un estudio de correlación para variables categóricas utilizando el test chi-square, para ello, en primer lugar, tenemos que crear una tabla de frecuencias de las variables y luego estudiar su relación mediante dicho test.

Pero antes, debemos discretizar la variable precio. lo haremos mediante divisiones en función de la frecuencia, para tener grupos lo más parecidos posibles, tendremos 4 niveles de precio:

- Primer cuartil: Bajo
- Segundo cuartil: Medio-bajo
- Tercer cuartil: Medio-alto
- Tercer cuartil: Alto

```
suppressWarnings(suppressMessages(library(arules)))
library(arules)
cuartil.price<-discretize(diamonds_df$price, method="frequency",breaks=4,labels=c("Bajo","Medio-Bajo","

table(cuartil.price)
```

```
## cuartil.price
##       Bajo Medio-Bajo Medio-Alto       Alto
##      13483      13476      13496      13485
```

Una vez realizada la discretización, procedemos a crear las tablas de contingencia para ver la relación entre el Precio y las variables cualitativas.

```
#Variable "cut"
tbl.cut<-table(cuartil.price,diamonds_df$cut)
tbl.cut
```

```
##
## cuartil.price Fair Good Ideal Premium Very Good
##    Bajo         88 1057  6304    2905      3129
##    Medio-Bajo  459 1082  6342    3067      2526
##    Medio-Alto  670 1639  4303    3508      3376
##    Alto        393 1128  4602    4311      3051
```

```
#Variable "Color"
tbl.color<-table(cuartil.price,diamonds_df$color)
tbl.color
```

```
##
## cuartil.price    D    E    F    G    H    I    J
##    Bajo       1875 2787 2268 2917 2023 1183  430
##    Medio-Bajo 2048 2985 2626 2959 1446  888  524
##    Medio-Alto 1700 2473 2536 2230 2310 1460  787
##    Alto       1152 1552 2112 3186 2525 1891 1067
```

```
#Variable "Clarity"
tbl.clarity<-table(cuartil.price,diamonds_df$clarity)
tbl.clarity
```

```
##
## cuartil.price   I1   IF  SI1  SI2  VS1  VS2 VVS1 VVS2
##    Bajo         55  616 2931 1024 2296 3389 1392 1780
##    Medio-Bajo  189  733 2850 1524 2185 3107 1356 1532
##    Medio-Alto  319  171 4130 4110 1438 2202  458  668
##    Alto        178  270 3154 2536 2252 3560  449 1086
```

Donde, de manera visual, parece dificil establecer relaciones entre el Precio y el resto de las variables.

A continuación, realizamos los tests chi-square para cada una de las tablas, con el fin de comprobar si existe relación.

```
chisq.test(cuartil.price, diamonds_df$cut)
```

```
##
##  Pearson's Chi-squared test
##
## data:  cuartil.price and diamonds_df$cut
## X-squared = 1748.1, df = 12, p-value < 2.2e-16
```

```
#chisq.test(tbl.clarity)
#chisq.test(tbl.color)
```

Alternativa utilizando el método V de Cramer, este método da un valor entre 0 y 1, donde 0 es nada relacionado y 1 totalmente relacionado

```r
suppressWarnings(suppressMessages(library(rcompanion)))
library(rcompanion)
cramerV(cuartil.price, diamonds_df$cut)
```

```
## Cramer V
##   0.1039
```

```r
cramerV(cuartil.price, diamonds_df$color)
```

```
## Cramer V
##   0.1134
```

```r
cramerV(cuartil.price, diamonds_df$clarity)
```

```
## Cramer V
##   0.1846
```

Y vemos que no podemos establecer una relación directa entre el Precio y las variables de Color, Corte y Claridad.

4.3.2. Contraste de Hipótesis

4.3.3. Regrestiones

```r
modelo1<-lm(price~volume, data=diamonds_df)
summary(modelo1)
```

```
##
## Call:
## lm(formula = price ~ volume, data = diamonds_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -162792    -780     -86     431   12692
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.058e+03  1.422e+01  -144.7   <2e-16 ***
## volume       4.610e+01  9.373e-02   491.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1703 on 53938 degrees of freedom
## Multiple R-squared:  0.8177, Adjusted R-squared:  0.8177
## F-statistic: 2.419e+05 on 1 and 53938 DF,  p-value: < 2.2e-16
```

```r
modelo2<-lm(price~volume+carat, data=diamonds_df)
summary(modelo2)
```

```
##
## Call:
## lm(formula = price ~ volume + carat, data = diamonds_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18521.7   -805.5    -19.5    538.2  12723.8
##
## Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept) -2259.766     13.056 -173.087  < 2e-16 ***
## volume          3.175      0.411    7.727 1.12e-14 ***
## carat        7243.588     67.841  106.774  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1548 on 53937 degrees of freedom
## Multiple R-squared:  0.8495, Adjusted R-squared:  0.8495
## F-statistic: 1.522e+05 on 2 and 53937 DF,  p-value: < 2.2e-16
```

```
modelo3<-lm(price~volume+carat+clarity, data=diamonds_df)
summary(modelo3)
```

```
##
## Call:
## lm(formula = price ~ volume + carat + clarity, data = diamonds_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17318.4   -639.6   -110.1    480.0  11159.1
##
## Coefficients:
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept) -6906.8216    50.2143 -137.547  < 2e-16 ***
## volume          2.0699     0.3435    6.025  1.7e-09 ***
## carat        8105.3266    56.9751  142.261  < 2e-16 ***
## clarityIF    5505.1373    57.3619   95.972  < 2e-16 ***
## claritySI1   3723.4461    49.1556   75.748  < 2e-16 ***
## claritySI2   2872.6472    49.4683   58.070  < 2e-16 ***
## clarityVS1   4606.3902    50.1297   91.890  < 2e-16 ***
## clarityVS2   4382.3745    49.3769   88.754  < 2e-16 ***
## clarityVVS1  5179.3698    53.0446   97.642  < 2e-16 ***
## clarityVVS2  5156.0880    51.6183   99.889  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1293 on 53930 degrees of freedom
## Multiple R-squared:  0.8949, Adjusted R-squared:  0.8949
## F-statistic: 5.104e+04 on 9 and 53930 DF,  p-value: < 2.2e-16
```

```
modelo4<-lm(price~carat+volume+clarity+color, data=diamonds_df)
summary(modelo4)
```

```
##
## Call:
## lm(formula = price ~ carat + volume + clarity + color, data = diamonds_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17277.1   -677.4   -192.5    472.5  10310.0
##
## Coefficients:
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept) -6695.4206    47.1908 -141.880  < 2e-16 ***
## carat        8546.6021    51.7117  165.274  < 2e-16 ***
## volume          1.9139     0.3108    6.158  7.4e-10 ***
## clarityIF    5710.7167    52.0021  109.817  < 2e-16 ***
## claritySI1   3789.8916    44.4909   85.184  < 2e-16 ***
## claritySI2   2826.6116    44.7675   63.140  < 2e-16 ***
## clarityVS1   4778.8817    45.3940  105.276  < 2e-16 ***
## clarityVS2   4460.0146    44.6870   99.806  < 2e-16 ***
## clarityVVS1  5345.0754    48.0293  111.288  < 2e-16 ***
## clarityVVS2  5227.4106    46.7194  111.889  < 2e-16 ***
## colorE       -216.7710    18.5193  -11.705  < 2e-16 ***
## colorF       -314.8760    18.7151  -16.825  < 2e-16 ***
## colorG       -508.9292    18.3223  -27.777  < 2e-16 ***
## colorH       -985.3325    19.4840  -50.571  < 2e-16 ***
## colorI      -1441.3788    21.8913  -65.842  < 2e-16 ***
## colorJ      -2340.1862    27.0199  -86.610  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1170 on 53924 degrees of freedom
## Multiple R-squared:  0.914,  Adjusted R-squared:  0.914
## F-statistic: 3.822e+04 on 15 and 53924 DF,  p-value: < 2.2e-16
```

Hemos elegido el modelo lineal, ya que consideramos que las mejores variables predictoras son el volumen y el peso, además el color y la apariencia visual ayudarán a ajustar más el modelo. Para conseguir el modelo óptimo, construimos 4 modelos, a los que hemos ido añdiendo más variables hasta conseguir un modelo óptimo. En el primer modelo solo correlacionamos el volumen y el precio y el coeficiente de determinación es alto ya que da un 0.81 En el segundo modelo añadimos el peso y el coeficiente de determinación sube hasta dar un 0.85 En el tercer modelo utlizamos el peso, volumen y la apariencia visual y el coeficiente de determinación da un 0.89 Finalmente en el cuarto y último modelo, utilizamos el peso, volumen, la apariencia visual y el color y el coeficiente de determinación da un 0.91. Consideramos que ya es un número bastante alto, además el p-valor es 0, lo que indica que es un buen modelo

Como prueba, vamos a intentar predecir el precio de un diamante según sus características

```
dfPrueba<- data.frame(carat=2.54,volume=400,clarity='SI2',color='I')
predict(modelo4,dfPrueba)
```

```
##        1
## 17163.75
```

La prueba realizada indica que para un peso de 2.54, volumen de 400, apariencia visual SI2 y color I, el modelo predice que el precio sería de 17.164 dólares.

Ahora realizaremos un estudio de regresión logística, intentaremos predecir si el diamante tiene precio alto en función del peso, volumen

```
diamonds_df$PA<-ifelse(diamonds_df$price>=unname(quantile(diamonds_df$price,.75)),1,0)
diamonds_df$CA<-ifelse(diamonds_df$carat>=unname(quantile(diamonds_df$carat,.75)),1,0)

modelo_logistico_1 <- glm(formula=PA~CA , data = diamonds_df, family = "binomial")
summary (modelo_logistico_1)
```

```
##
## Call:
## glm(formula = PA ~ CA, family = "binomial", data = diamonds_df)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.80197  -0.34859  -0.34859  -0.09574   2.37954
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.77034    0.02121  -130.6   <2e-16 ***
## CA           4.17423    0.03009   138.7   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 60665  on 53939  degrees of freedom
## Residual deviance: 31721  on 53938  degrees of freedom
## AIC: 31725
##
## Number of Fisher Scoring iterations: 5
```

```
print(paste0("Valor OR: ",exp(coefficients(modelo_logistico_1)[2])))
```

```
## [1] "Valor OR: 64.9897247442485"
```

```
diamonds_df$VA<-ifelse(diamonds_df$volume>=unname(quantile(diamonds_df$volume,.75)),1,0)


modelo_logistico_2 <- glm(formula=PA~CA+VA , data = diamonds_df, family = "binomial")
summary (modelo_logistico_2)
```

```
##
## Call:
## glm(formula = PA ~ CA + VA, family = "binomial", data = diamonds_df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8495  -0.3459  -0.3459  -0.1015   2.3859
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept) -2.78634    0.02128 -130.93   <2e-16 ***
## CA           2.21945    0.08532   26.01   <2e-16 ***
## VA           2.07786    0.08548   24.31   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 60665  on 53939  degrees of freedom
## Residual deviance: 31150  on 53937  degrees of freedom
## AIC: 31156
##
## Number of Fisher Scoring iterations: 5
```

```
print(paste0("Valor OR: ",exp(coefficients(modelo_logistico_2)[2])))
```

```
## [1] "Valor OR: 9.20228865705978"
```

```
print(paste0("Valor OR: ",exp(coefficients(modelo_logistico_2)[3])))
```

```
## [1] "Valor OR: 7.98732688872872"
```

Hemos conseguido reducir el valor de AIC, nuestro modelo ha mejorado. Todas las variables tienen importancia en el modelo. Según el modelo, volumen alto y peso alto producen un precio alto en el diamante, para saber si esta afirmación es correcta, realizaremos el test de holem para saber si el modelo está bien ajustado y por otro lado, miraremos al curca ROC y el área bajo la curva para poder afirmar que el volumen y peso alto producen un precio alto.

```
suppressWarnings(suppressMessages(library(ResourceSelection)))
library(ResourceSelection)
hoslem.test(diamonds_df$PA, fitted(modelo_logistico_2))
```

```
##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  diamonds_df$PA, fitted(modelo_logistico_2)
## X-squared = 6.5297e-17, df = 8, p-value = 1
```

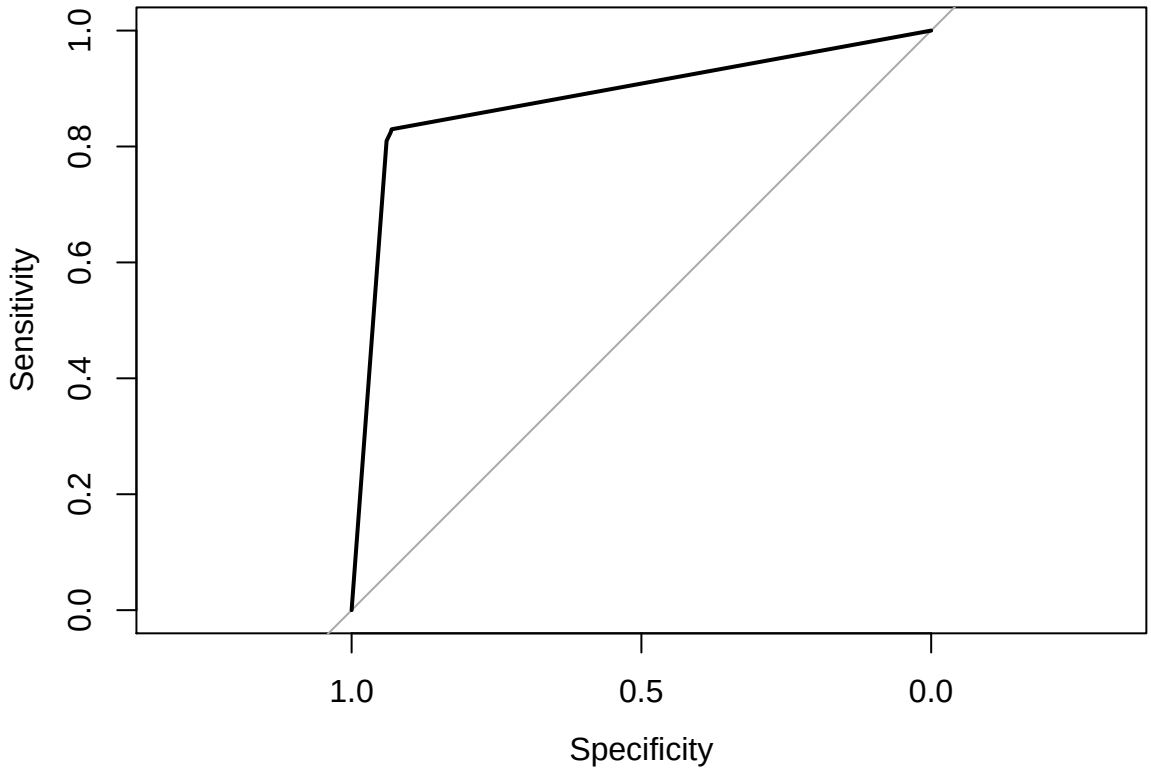
El test de Hoslem nos da un p-valor de 1, indica que el modelo está muy bien ajustado

```
suppressWarnings(suppressMessages(library(pROC)))

library (pROC)
prob_p_alto= predict(modelo_logistico_2, diamonds_df, type="response")

g= roc(diamonds_df$PA, prob_p_alto, data=diamonds_df)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot (g)
```




```
auc (g)
```

```
## Area under the curve: 0.8833
```

El área bajo la curva es de 0.88, es un valor bastante alto. Por lo tanto podemos afirmar que el precio alto está ligado a un volumen alto y peso alto. Intentaremos predecir la probabilidad de tener un precio alto, si el peso y el volumen son altos

```
df_2<-data.frame (CA=1,VA=1)
predict (modelo_logistico_2, df_2 ,type="response")
```

```
##         1
## 0.8192041
```

5. Representación de los resultados a partir de tablas y gráficas.

6. Resolución del problema. A partir de los resultados obtenidos, ¿cuáles son las conclusiones? ¿Los resultados permiten responder al problema?