

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?

El dataset “Diamonds” ha sido descargado de Kaggle, cuya licencia es *desconocida* tal y como indica el mismo.

Este Set de datos puede ser descargado de: <https://www.kaggle.com/shivam2503/diamonds>

El set de datos contiene 53940 registros de diamantes. Para cada registro, 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, dividido por su diámetro medio ($z / \text{media}(x, y)$)
- 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

Además, se añadirá un atributo secundario que facilitará el análisis. Este atributo secundario, calculado a partir de los descritos en la lista, es:

- Volumen

A continuación, para mayor claridad, se incluyen las partes del diamante, que nos ayudará a localizar los atributos de medida (Altura, Tabla, X, Y, Z)

Las diferentes partes de un diamante

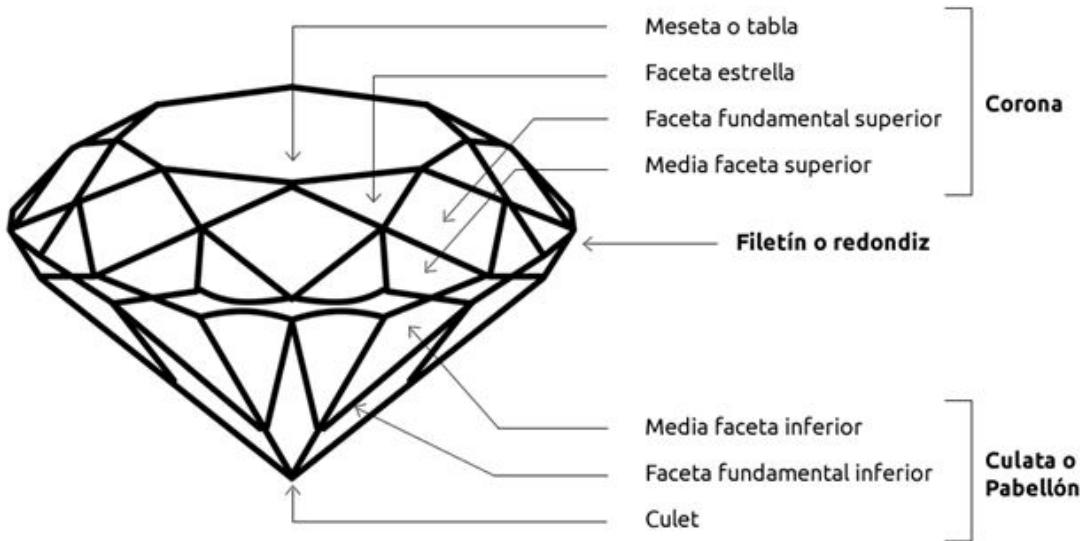


Figure 1: Partes del diamante

Resulta interesante desde un punto de vista de la limpieza y el análisis de los datos obtenidos. En primer lugar, nos permite ejecutar ciertas tareas de limpieza para adecuar los datos a un posterior análisis exhaustivo de las variables.

La principal pregunta que este análisis intentará responder cómo se relacionan las diferentes características físicas de un diamante entre sí, y qué grado de relación guardan con el precio de cada uno de esos diamantes.

Las características de los atributos disponibles nos permitirán hacer varios estudios, a destacar:

- Correlaciones
- Regresiones de varios tipos
- Hipótesis
- Árboles de clasificación

Todos los modelos son característicos del análisis de datos en la Ciencia de Datos, por tanto, la importancia radica en la usabilidad de este dataset y sus aplicaciones prácticas de los modelos anteriormente mencionados.

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 función 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
##
##      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 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.

Antes de proceder con la evaluación de ceros o elementos vacíos, se va a realizar una limpieza de datos duplicados. En primer lugar, veamos si tenemos duplicidades en nuestros datos:

```
nrow(diamonds_df[duplicated(diamonds_df), ])
```

```
## [1] 146
```

Vemos que tenemos 146 registros duplicados, vamos a proceder a su eliminación, ya que para nuestro estudio posterior, podrían alterar los resultados.

```
library(dplyr)
```

```
##  
## Attaching package: 'dplyr'  
  
## The following objects are masked from 'package:stats':  
##  
##     filter, lag  
  
## The following objects are masked from 'package:base':  
##  
##     intersect, setdiff, setequal, union  
  
diamonds_df<-distinct(diamonds_df)
```

Volvemos a comprobar

```
nrow(diamonds_df[duplicated(diamonds_df), ])
```

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

Ya podríamos continuar con el análisis.

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 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 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
## 11157	1.07	Ideal	F	SI2	61.6	56	4954	NA	6.62	NA	0
## 11936	1.00	Very Good	H	VS2	63.3	53	5139	NA	NA	NA	0
## 15915	1.14	Fair	G	VS1	57.5	67	6381	NA	NA	NA	0
## 24465	1.56	Ideal	G	VS2	62.2	54	12800	NA	NA	NA	0
## 26184	1.20	Premium	D	VVS1	62.1	59	15686	NA	NA	NA	0
## 27365	2.25	Premium	H	SI2	62.8	59	18034	NA	NA	NA	0
## 49414	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=1)
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
## 11157	1.07	Ideal	F	SI2	61.6	56	4954	6.50	6.62	NA	0
## 11936	1.00	Very Good	H	VS2	63.3	53	5139	6.61	NA	NA	0
## 15915	1.14	Fair	G	VS1	57.5	67	6381	6.32	NA	NA	0
## 24465	1.56	Ideal	G	VS2	62.2	54	12800	7.34	NA	NA	0
## 26184	1.20	Premium	D	VVS1	62.1	59	15686	7.01	NA	NA	0
## 27365	2.25	Premium	H	SI2	62.8	59	18034	8.42	NA	NA	0
## 49414	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
## 11936	1.00	Very Good	H	VS2	63.3	53	5139	6.61	NA	NA	0
## 15915	1.14	Fair	G	VS1	57.5	67	6381	6.32	NA	NA	0
## 24465	1.56	Ideal	G	VS2	62.2	54	12800	7.34	NA	NA	0
## 26184	1.20	Premium	D	VVS1	62.1	59	15686	7.01	NA	NA	0
## 27365	2.25	Premium	H	SI2	62.8	59	18034	8.42	NA	NA	0
## 49414	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=1)
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
## 11936	1.00	Very Good	H	VS2	63.3	53	5139	6.61	6.54	NA	0
## 15915	1.14	Fair	G	VS1	57.5	67	6381	6.32	6.19	NA	0
## 24465	1.56	Ideal	G	VS2	62.2	54	12800	7.34	7.31	NA	0
## 26184	1.20	Premium	D	VVS1	62.1	59	15686	7.01	6.88	NA	0
## 27365	2.25	Premium	H	SI2	62.8	59	18034	8.42	8.37	NA	0
## 49414	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
## 2202	1.00	Premium	G	SI2	59.1	59	3142	6.55	6.48	NA	0
## 2309	1.01	Premium	H	I1	58.1	59	3167	6.66	6.60	NA	0

```

## 4779 1.10 Premium G SI2 63.0 59 3696 6.50 6.47 NA 0
## 5458 1.01 Premium F SI2 59.2 58 3837 6.50 6.47 NA 0
## 10146 1.50 Good G I1 64.0 61 4731 7.15 7.04 NA 0
## 11157 1.07 Ideal F SI2 61.6 56 4954 6.50 6.62 NA 0
## 11936 1.00 Very Good H VS2 63.3 53 5139 6.61 6.54 NA 0
## 13571 1.15 Ideal G VS2 59.2 56 5564 6.88 6.83 NA 0
## 15915 1.14 Fair G VS1 57.5 67 6381 6.32 6.19 NA 0
## 24339 2.18 Premium H SI2 59.4 61 12631 8.49 8.45 NA 0
## 24465 1.56 Ideal G VS2 62.2 54 12800 7.34 7.31 NA 0
## 26064 2.25 Premium I SI1 61.3 58 15397 8.52 8.42 NA 0
## 26184 1.20 Premium D VVS1 62.1 59 15686 7.01 6.88 NA 0
## 27048 2.20 Premium H SI1 61.2 59 17265 8.42 8.37 NA 0
## 27365 2.25 Premium H SI2 62.8 59 18034 8.42 8.37 NA 0
## 27439 2.02 Premium H VS2 62.7 53 18207 8.02 7.95 NA 0
## 27673 2.80 Good G SI2 63.8 58 18788 8.90 8.85 NA 0
## 49414 0.71 Good F SI2 64.1 60 2130 6.50 6.47 NA 0
## 51362 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=1)
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
## 2202 1.00 Premium G SI2 59.1 59 3142 6.55 6.48 3.86 0
## 2309 1.01 Premium H I1 58.1 59 3167 6.66 6.60 4.05 0
## 4779 1.10 Premium G SI2 63.0 59 3696 6.50 6.47 4.05 0
## 5458 1.01 Premium F SI2 59.2 58 3837 6.50 6.47 3.82 0
## 10146 1.50 Good G I1 64.0 61 4731 7.15 7.04 4.63 0
## 11157 1.07 Ideal F SI2 61.6 56 4954 6.50 6.62 3.99 0
## 11936 1.00 Very Good H VS2 63.3 53 5139 6.61 6.54 4.14 0
## 13571 1.15 Ideal G VS2 59.2 56 5564 6.88 6.83 3.96 0
## 15915 1.14 Fair G VS1 57.5 67 6381 6.32 6.19 3.79 0
## 24339 2.18 Premium H SI2 59.4 61 12631 8.49 8.45 5.09 0
## 24465 1.56 Ideal G VS2 62.2 54 12800 7.34 7.31 4.55 0
## 26064 2.25 Premium I SI1 61.3 58 15397 8.52 8.42 5.02 0
## 26184 1.20 Premium D VVS1 62.1 59 15686 7.01 6.88 4.16 0
## 27048 2.20 Premium H SI1 61.2 59 17265 8.42 8.37 5.13 0
## 27365 2.25 Premium H SI2 62.8 59 18034 8.42 8.37 5.23 0
## 27439 2.02 Premium H VS2 62.7 53 18207 8.02 7.95 5.06 0
## 27673 2.80 Good G SI2 63.8 58 18788 8.90 8.85 5.10 0
## 49414 0.71 Good F SI2 64.1 60 2130 6.50 6.47 3.83 0
## 51362 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
## volume
## FALSE
```

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

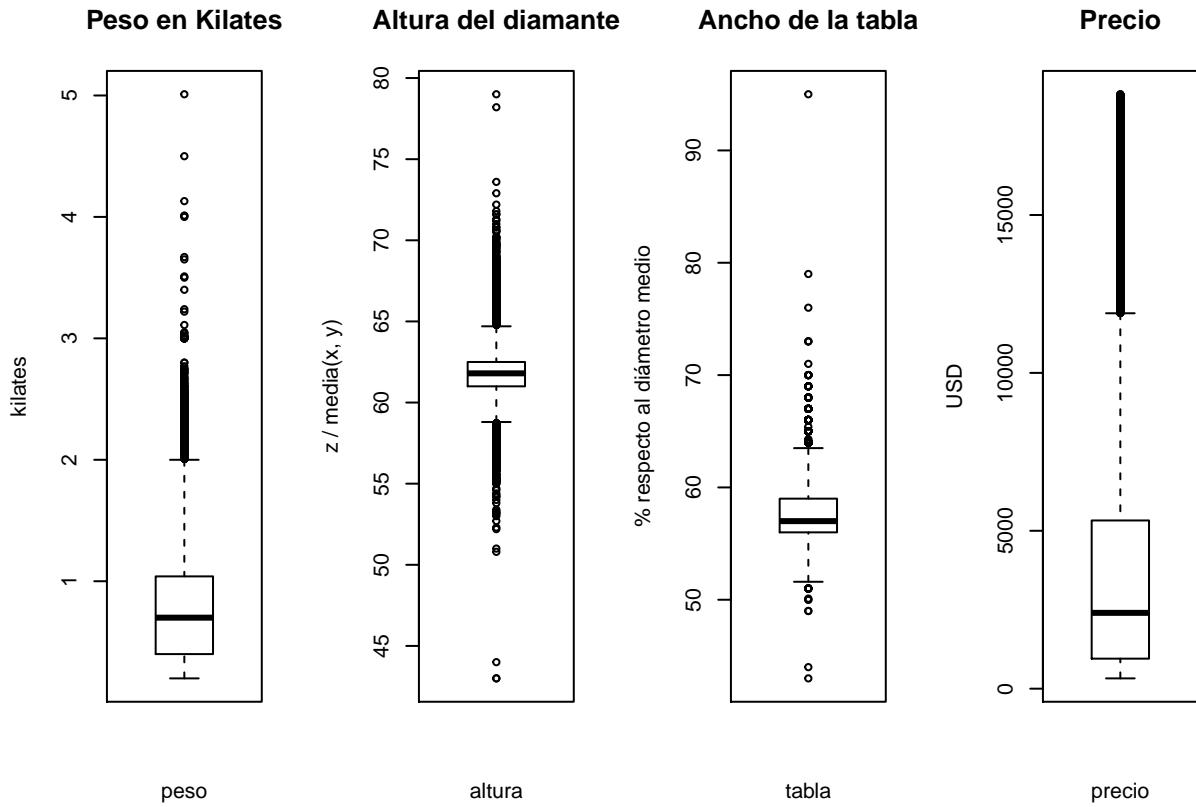
```
##   carat      cut    color clarity   depth   table   price      x      y      z
## FALSE    FALSE    FALSE    FALSE    FALSE    FALSE    FALSE    FALSE    FALSE
## volume
## FALSE
```

Efectivamente, ya no hay ningún valor cero ni elementos vacío

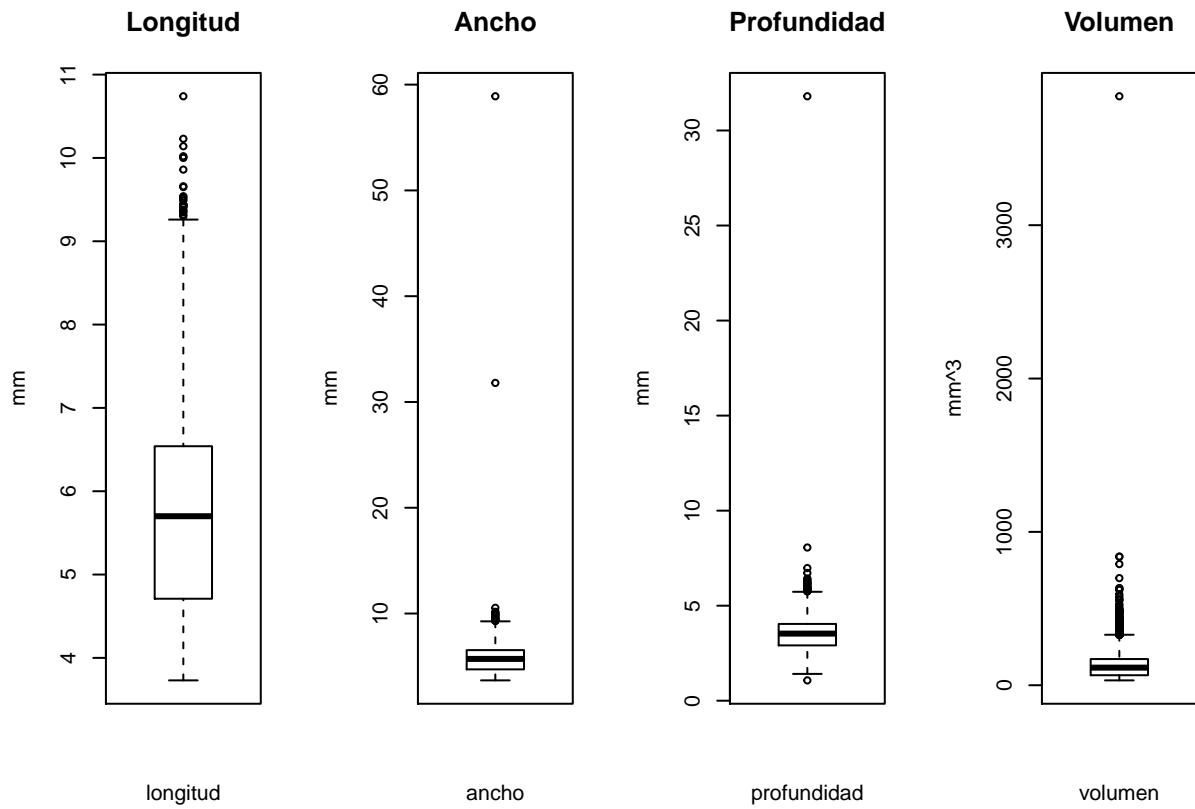
3.2 Identificación y tratamiento de valores extremos.

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
par (mfrow= c (1,4) )
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")
```



```
par (mfrow= c (1,4))
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.

```

preciomax<-max(diamonds_df$price)
volumenmax<-max(diamonds_df$volume)
cat("El precio máximo registrado es: ",preciomax, "\n El volumen máximo registrado es: ", volumenmax)

## El precio máximo registrado es: 18823
## El volumen máximo registrado es: 3840.598

diamonds_df[diamonds_df$price==preciomax,]

##           carat      cut color clarity depth table price     x     y     z   volume
## 27683  2.29 Premium     I    VS2   60.8   60 18823 8.5 8.47 5.16 371.4942

```

Evaluaremos los máximos de la Carat, Precio, “X”, “Y”, “Z”, Tabla, profundidad y Volumen, ya que, según los gráficos muestran los valores extremos más acusados.

En primer lugar, vamos a filtrar por los valores máximos de Peso, tabla, “X”, “Y”, “Z” y Volumen (variables que, como veremos más adelante, están fuertemente correlacionadas).

```

diamonds_df[diamonds_df$carat==max(diamonds_df$carat) | diamonds_df$table==max(diamonds_df$table) | diamonds_
##           carat depth table price     x     y     z   volume

```

```

## 24013 2.00 58.9 57.0 12210 8.09 58.90 8.06 3840.5981
## 24876 2.01 58.6 95.0 13387 8.32 8.31 4.87 336.7079
## 27351 5.01 65.5 59.0 18018 10.74 10.54 6.98 790.1332
## 48269 0.51 61.8 54.7 1970 5.12 5.15 31.80 838.5024
## 52716 0.50 79.0 73.0 2579 5.21 5.18 4.09 110.3801

```

Si observamos la tabla anterior, podemos ver que cuando tenemos el precio máximo, el tamaño “x” es el máximo también, y de la misma manera su peso en kilates, lo que tiene sentido. Sin embargo, para los otros dos máximos que tenemos para “y” y “z”, como podemos observar en la tabla de resultados al filtrar por los correspondientes máximos, no se corresponden a máximos en kilates ni máximos en precio, por tanto, teniendo en cuenta que, al menos las variables peso, “X”, “Y”, “Z” y volumen (variables físicas relacionadas), no aportan datos coherentes, debemos suponer que los datos son erróneos. Además, en el caso de la Profundidad y la tabla, como veremos a continuación, son variables con correlación negativa, esto es, inversamente proporcionales, por tanto, el máximo que encontramos en Profundidad y Tabla tampoco tendrían sentido, observando los datos en su contexto, comparados con el peso, volumen y el precio.

Vamos a continuar analizando las variables Tabla, Profundidad y ancho (“Y”):

```

suppressWarnings(suppressMessages(library(tidyverse)))
library(tidyverse)
diamonds_df %>% top_n(n=5, y)

```

```

##   carat      cut color clarity depth table price     x     y     z   volume
## 1  2.00 Premium     H    SI2  58.9     57 12210 8.09 58.90 8.06 3840.5981
## 2  4.01 Premium     I     I1  61.0     61 15223 10.14 10.10 6.17 631.8944
## 3  5.01 Fair       J     I1  65.5     59 18018 10.74 10.54 6.98 790.1332
## 4  4.50 Fair       J     I1  65.8     58 18531 10.23 10.16 6.72 698.4553
## 5  0.51 Ideal      E    VS1  61.8     55 2075  5.15 31.80 5.12  838.5024

```

En Y, tenemos otro máximo que no tiene ningún sentido, procedemos al borrado.

```

diamonds_df[diamonds_df$y==31.8, ]

```

```

##      carat      cut color clarity depth table price     x     y     z   volume
## 49048 0.51 Ideal      E    VS1  61.8     55 2075  5.15 31.8 5.12  838.5024

```

Por otra parte, con el resto de variables no se encuentran más incosistencias.

```

diamonds_df %>% top_n(n=5, table)

```

```

##   carat      cut color clarity depth table price     x     y     z   volume
## 1  2.01 Fair      F    SI1  58.6     95 13387 8.32 8.31 4.87 336.70790
## 2  0.70 Fair      H    VS1  62.0     73 2100 5.65 5.54 3.47 108.61447
## 3  0.81 Fair      F    SI2  68.8     79 2301 5.26 5.20 3.58  97.92016
## 4  0.79 Fair      G    SI1  65.3     76 2362 5.52 5.13 3.35  94.86396
## 5  0.71 Fair      D    VS2  55.6     73 2368 6.01 5.96 3.33 119.27927
## 6  0.50 Fair      E    VS2  79.0     73 2579 5.21 5.18 4.09 110.38010

```

```

diamonds_df %>% top_n(n=5, depth)

```

```

##   carat cut color clarity depth table price     x     y     z volume
## 1  1.03 Fair     E      I1  78.2     54 1262 5.72 5.59 4.42 141.3286
## 2  0.99 Fair     J      I1  73.6     60 1789 6.01 5.80 4.35 151.6323
## 3  0.96 Fair     G      SI2  72.2     56 2438 6.01 5.81 4.28 149.4495
## 4  0.50 Fair     E      VS2  79.0     73 2579 5.21 5.18 4.09 110.3801
## 5  0.90 Fair     G      SI1  72.9     54 2691 5.74 5.67 4.16 135.3905

```

Por tanto, resumiendo:

- Máximo en Precio y Peso y medida “X”: los tres valores corresponden al mismo registro, por tanto, son valores coherentes
- Máximo en Profundidad y Tabla: El máximo de Profundidad no corresponde con un mínimo de Tabla y viceversa, como veremos, ambas variables son inversamente proporcionales, por tanto, no son valores coherentes, ya que ademas, no se corresponden con máximos de otras variables
- Maximos en Volumen y medida “y”: En este caso tampoco son coherentes ya que no se corresponden con un máximo de Precio y Peso, siendo estas variables altamente correlacionadas.
- Maximo en “Z”: Tampoco sería coherente, ya que, para ese registro, su volumen es relativamente alto, pero su precio y su peso son mínimos.

Para corregir lo anterior, se van a proceder a borrar los datos que registran máximos incoherentes.

```
diamonds_df<-diamonds_df[-c(24013, 24876, 48269, 52716, 49048), ]
```

4. Análisis de los datos.

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

Vamos a realizar un análisis que responderá cómo están relacionadas las principales características de un diamante.

Las características que analizaremos son:

- El precio
- El volumen
- Los tipos de corte : Good y Premium
- El color
- La apariencia visual

Los estudios y pruebas que realizaremos sobre los atributos disponibles del conjunto de datos serán:

- Correlaciones
- Regresiones de varios tipos
- Hipótesis
- Árboles de clasificación

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

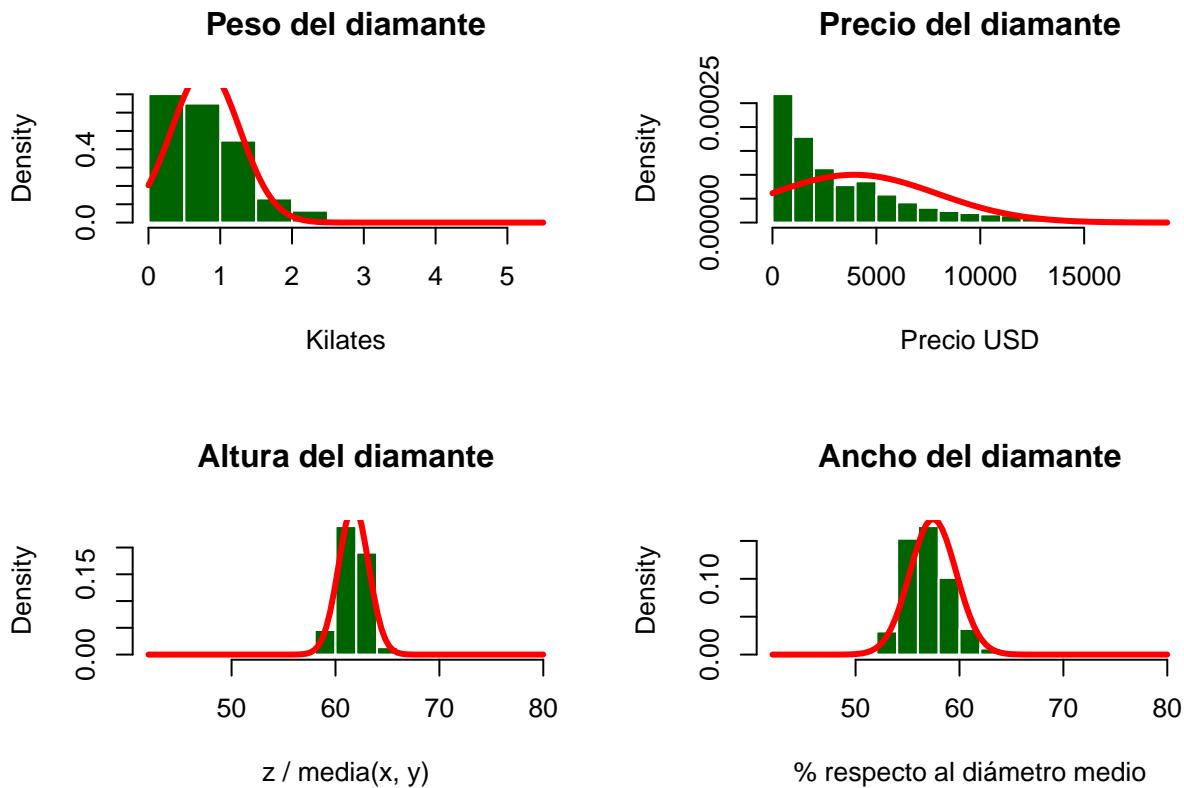
En primer lugar, se realiza un estudio sobre la normalidad de las variables precio, altura, tabla, x, y, z y volumen.

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

```
par(mfrow=c(2,2))
#Variable peso
hist(diamonds_df$carat, main= "Peso del diamante", freq = FALSE, xlab = "Kilates", col = "dark green", lwd=3)
curve(dnorm(x, mean=mean(diamonds_df$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 green", lwd=3)
curve(dnorm(x, mean=mean(diamonds_df$price), sd=sd(diamonds_df$price)), add=TRUE, col="red", lwd=3)

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

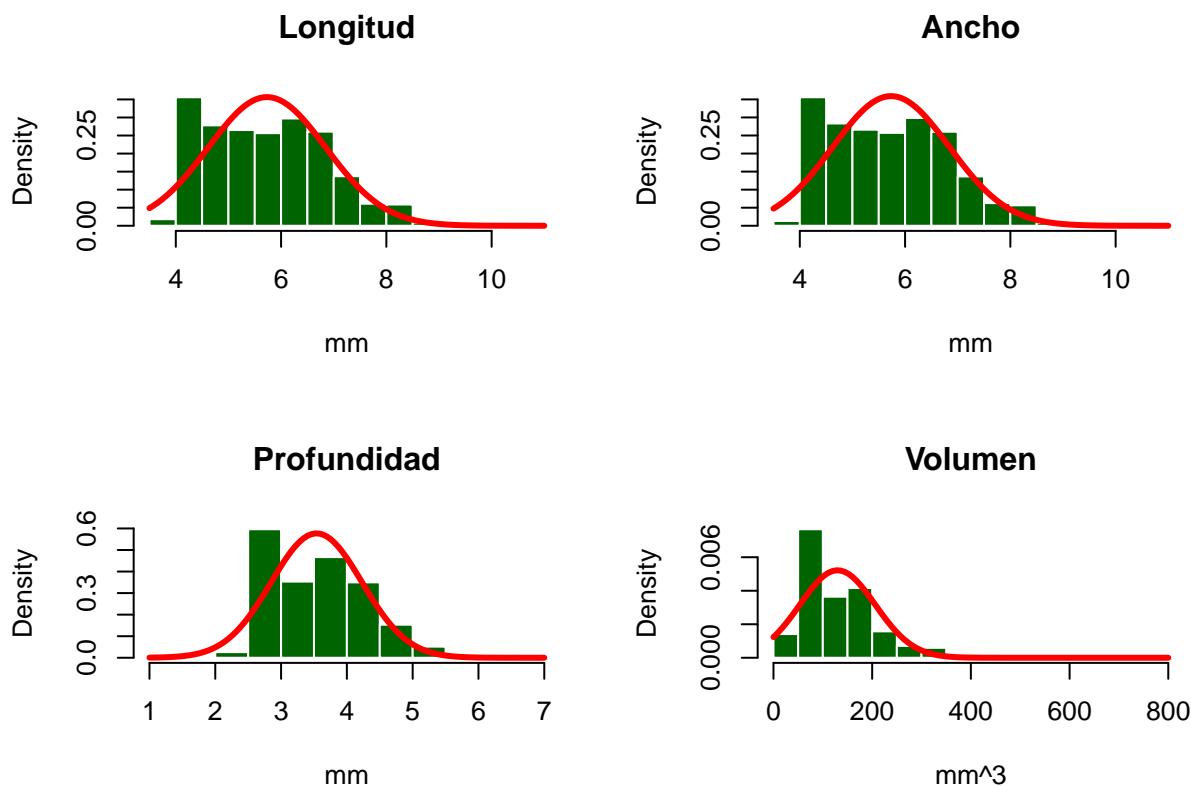


```

par(mfrow=c(2,2))
#Variable "x"
hist(diamonds_df$x, main= "Longitud", freq = FALSE, xlab = "mm", col = "dark green", border="white")
curve(dnorm(x, mean=mean(diamonds_df$x), sd=sd(diamonds_df$x)), add=TRUE, col="red", lwd=3)

#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$volume, main= "Volumen", freq = FALSE, xlab = "mm^3", col = "dark green", border="white")
curve(dnorm(x, mean=mean(diamonds_df$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 = 0.90915, p-value < 2.2e-16

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

##
## Shapiro-Wilk normality test
##
## data: diamonds_df$depth[1:5000]
## W = 0.95418, p-value < 2.2e-16

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

##
## Shapiro-Wilk normality test
##
## data: diamonds_df$table[1:5000]
## W = 0.95044, p-value < 2.2e-16

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

##
## Shapiro-Wilk normality test
##
## data: diamonds_df$y[1:5000]
## W = 0.82562, p-value < 2.2e-16

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

##
## Shapiro-Wilk normality test
##
## data: diamonds_df$z[1:5000]
## W = 0.85945, p-value < 2.2e-16

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

##
## Shapiro-Wilk normality test
##
## data: diamonds_df$volume[1:5000]
## W = 0.90739, 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
##      1596     4891    21487    13747    12068
```

Siendo estos valores:

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

```
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
```

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

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

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

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

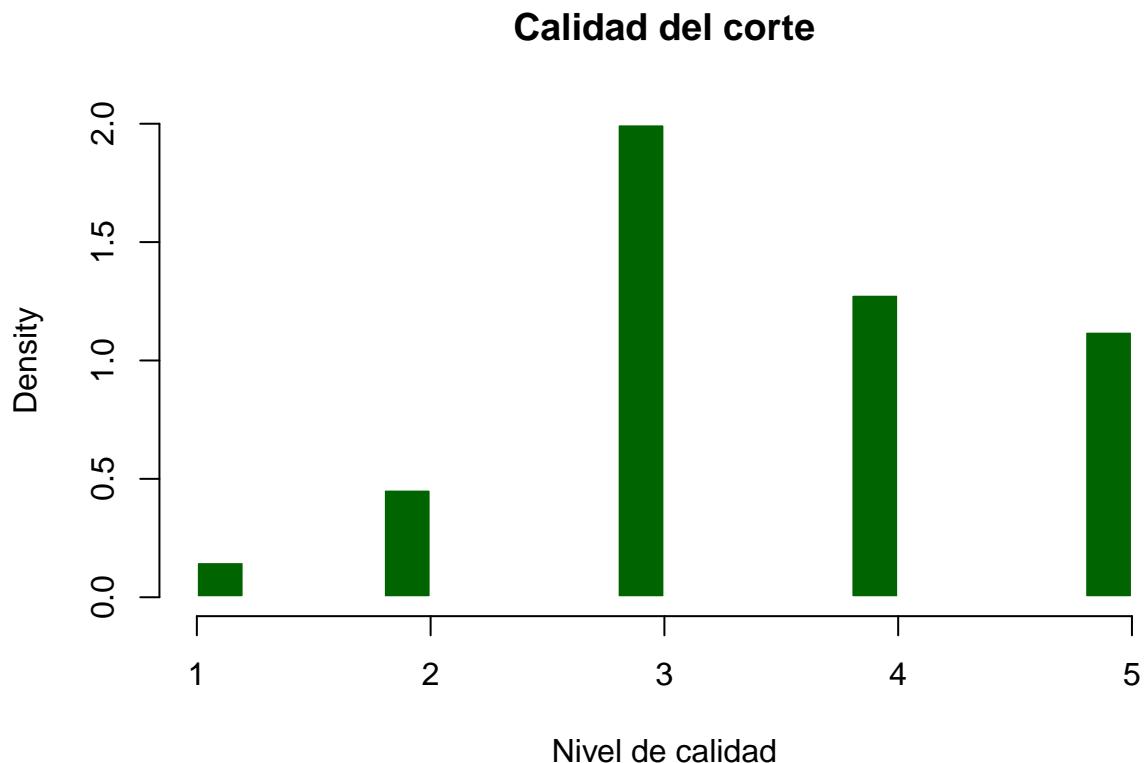
```
level.cut[level.cut=="Premium"]<-"4"
```

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

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

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

```
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 = 0.89186, 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

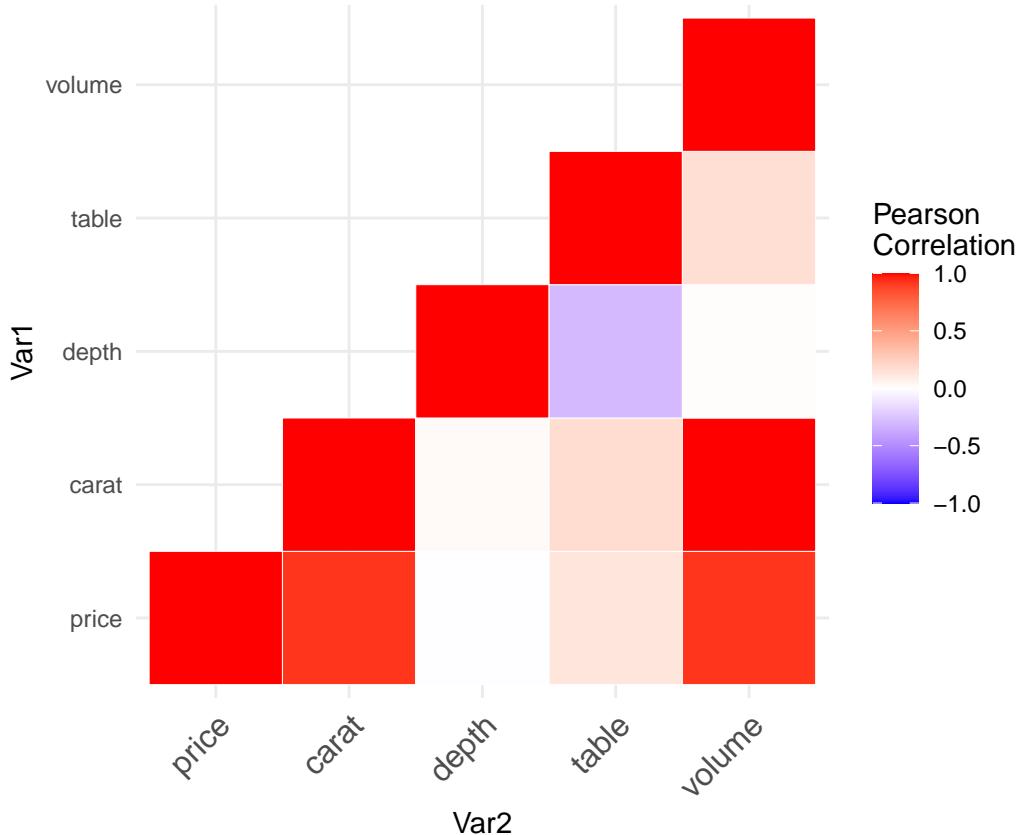
- Tabla
- Volumen

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

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

library(ggplot2)
#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-data
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 dia-

mante. Además, un dato importante que podemos observar es que las variables Profundidad y Tabla son inversamente proporcionales, teniendo una correlación negativa.

Para estas dos variables, vamos a estudiar una a una su índice de correlación:

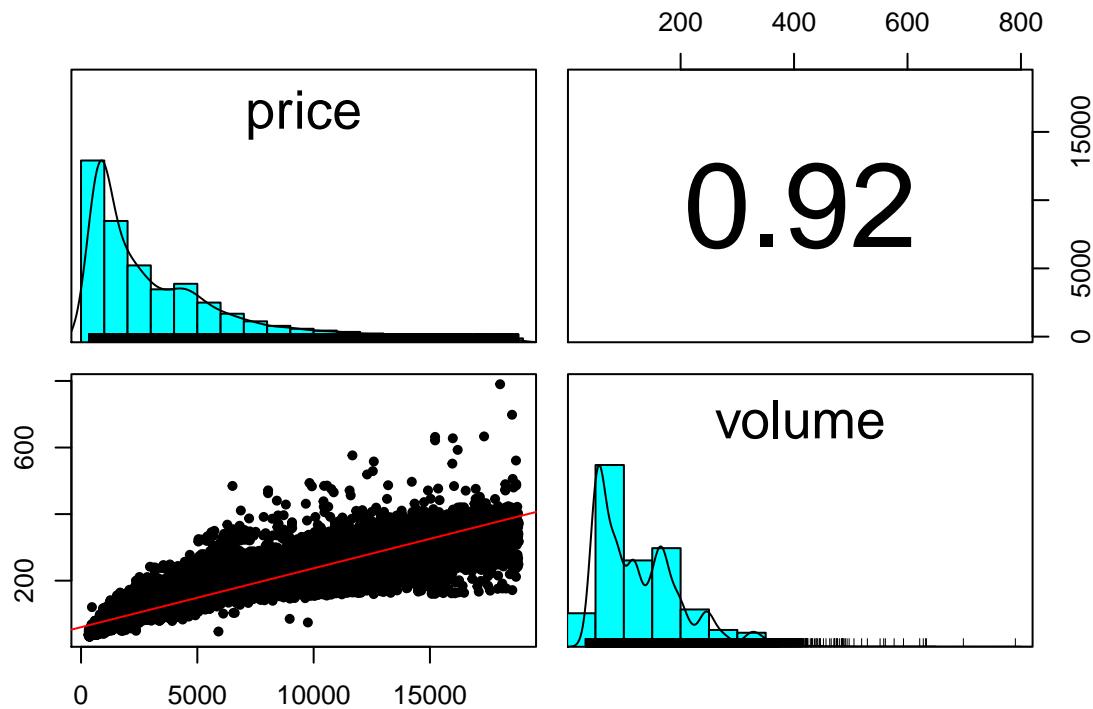
```
library(psych)
```

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

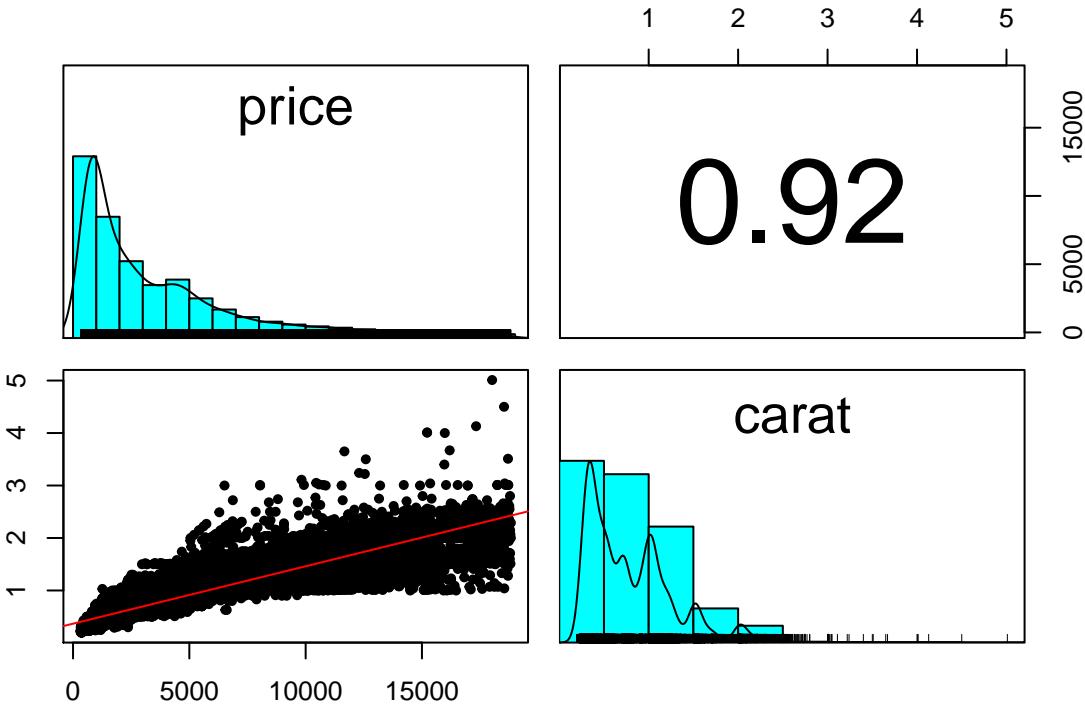
##
## Attaching package: 'psych'

## 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(7,11)], 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 chisquare, 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
- Cuarto cuartil: Alto

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

## cuartil.price
##      Bajo Medio-Bajo Medio-Alto      Alto
## 13442     13432     13467     13448
```

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 1055  6275   2900      3124
##  Medio-Bajo  457 1080  6318   3053      2524
##  Medio-Alto  664 1636  4296   3496      3375
##    Alto     387 1120  4598   4298      3045

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

## 
## cuartil.price D E F G H I J
##    Bajo      1868 2784 2263 2906 2011 1184 426
##  Medio-Bajo  2038 2971 2616 2951 1445  887 524
##  Medio-Alto 1699 2467 2533 2225 2303 1454 786
##    Alto     1150 1551 2107 3180 2512 1882 1066

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

## 
## cuartil.price I1 IF SI1 SI2 VS1 VS2 VVS1 VVS2
##    Bajo      55 614 2920 1023 2289 3375 1388 1778
##  Medio-Bajo 189 729 2847 1512 2178 3100 1352 1525
##  Medio-Alto 318 171 4121 4097 1437 2197  458 668
##    Alto     178 270 3143 2517 2250 3556  449 1085

```

Donde, de manera visual, parece difícil 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 = 1730.9, 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

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

```
## Cramer V
## 0.1036
```

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

```
## Cramer V
## 0.1131
```

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

```
## Cramer V
## 0.1845
```

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

Vamos a investigar si el precio medio de los diamantes con tipo de corte Premium es igual al de tipo de corte Good

Escribimos la hipótesis nula y la hipótesis alternativa

La Hipótesis nula sería que la esperanza de ambas muestras sean iguales y la Hipótesis alternativa H1 que la esperanza de ambas muestras sea diferente, por lo tanto es bilateral

$H_0: \mu_1 = \mu_2$ $H_1: \mu_1 \neq \mu_2$

Para poder asumir una normalidad en los datos, las muestran deben ser superiores a 30, por ello, imprimireé el dataset del primer apartado, en el que se mostraba un gráfico según el número de muestras para ambos tipos de corte

```
#Nivel de significacion
alpha<-1-0.95

hipdiamonds_df<-diamonds_df[diamonds_df$cut=="Good" | diamonds_df$cut=="Premium", ]

#Datos de Good

dfGood <- diamonds_df[diamonds_df$cut=="Good",]
nGood <- nrow(dfGood)
nGood

## [1] 4891
```

```

medGood <- mean(dfGood$ price)
sdGood <- sd(dfGood$ price)

#Datos de Premium
dfPremium <- diamonds_df[diamonds_df$cut=="Premium",]
nPremium <- nrow(dfPremium)

medPremium <- mean(dfPremium$ price)
sdPremium <- sd(dfPremium$ price)

s<- sqrt(((nGood-1)*sdGood^2+(nPremium-1)*sdPremium^2)/(nGood+nPremium-2))
s

## [1] 4180.785

sx<- s*sqrt(1/nGood + 1/nPremium)
sx

## [1] 69.60728

#Calculo de t
t<-(medGood-medPremium)/sx
t

## [1] -9.536733

#Calculo el valor crítico
tcritico<-qt((1-alpha/2),df=nGood+nPremium-2)
tcritico

## [1] 1.960091

#Calculo el pvalue
pvalue<-2*(1- (pt(abs(t),df=nGood+nPremium-2)))
pvalue

## [1] 0

```

El valor de pvalue es 0 es muy inferior al nivel de significación, por lo tanto, rechazamos la hipótesis nula, ya que la esperanza del precio medio es diferente entre ambos tipos de corte. Con respecto al valor crítico expresa el mismo resultado: el valor crítico es -1.96 hasta 1.96 y el valor del estadístico t es de -9 que está totalmente fuera de este intervalo, estamos muy alejados de la zona de aceptación de la Hipótesis nula. Después de ver los resultados, a través del p value y de valor crítico, se deduce que hay diferencias significativas en el precio entre los corte Premium y Good

4.3.3. Regrestiones

```

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

```

```

## 
## Call:
## lm(formula = price ~ volume, data = diamonds_df)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17748.8   -800.3    -22.0    545.2  12598.8
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -2.325e+03  1.301e+01 -178.8   <2e-16 ***
## volume       4.821e+01  8.637e-02   558.2   <2e-16 ***
## ---      
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 1530 on 53787 degrees of freedom
## Multiple R-squared:  0.8528, Adjusted R-squared:  0.8528 
## F-statistic: 3.116e+05 on 1 and 53787 DF,  p-value: < 2.2e-16
```

```
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
## -17274.2   -797.0    -23.2    543.9  12538.4
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -2347.457     13.108 -179.09   <2e-16 ***
## volume       71.740      1.876   38.24   <2e-16 ***
## carat       -3801.216    302.736  -12.56   <2e-16 ***
## ---      
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 1528 on 53786 degrees of freedom
## Multiple R-squared:  0.8532, Adjusted R-squared:  0.8532 
## F-statistic: 1.563e+05 on 2 and 53786 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
## -16588.4   -640.4   -107.7    485.3  11068.2
## 
```

```

## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6823.770    49.952 -136.61 < 2e-16 ***
## volume      47.312     1.586   29.82 < 2e-16 ***
## carat       807.859    256.442   3.15  0.00163 **
## clarityIF   5342.712    57.247   93.33 < 2e-16 ***
## claritySI1  3597.553    48.995   73.43 < 2e-16 ***
## claritySI2  2752.849    49.298   55.84 < 2e-16 ***
## clarityVS1  4461.038    50.019   89.19 < 2e-16 ***
## clarityVS2  4246.097    49.244   86.23 < 2e-16 ***
## clarityVVS1 5032.466    52.914   95.11 < 2e-16 ***
## clarityVVS2 5007.498    51.506   97.22 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1283 on 53779 degrees of freedom
## Multiple R-squared:  0.8965, Adjusted R-squared:  0.8965
## F-statistic: 5.175e+04 on 9 and 53779 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
## -16583.0   -676.8   -189.9    470.3  10227.3
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6617.779    46.886 -141.145 < 2e-16 ***
## carat        1635.296   231.865    7.053 1.77e-12 ***
## volume       44.756     1.434   31.216 < 2e-16 ***
## clarityIF   5555.616    51.841   107.167 < 2e-16 ***
## claritySI1  3670.235    44.296   82.857 < 2e-16 ***
## claritySI2  2713.066    44.563   60.882 < 2e-16 ***
## clarityVS1  4639.524    45.243   102.546 < 2e-16 ***
## clarityVS2  4330.126    44.517   97.269 < 2e-16 ***
## clarityVVS1 5204.741    47.859   108.753 < 2e-16 ***
## clarityVVS2 5085.336    46.565   109.209 < 2e-16 ***
## colorE       -215.967   18.378   -11.751 < 2e-16 ***
## colorF       -314.752   18.572   -16.948 < 2e-16 ***
## colorG       -506.319   18.183   -27.846 < 2e-16 ***
## colorH       -976.496   19.341   -50.487 < 2e-16 ***
## colorI      -1436.906   21.722   -66.149 < 2e-16 ***
## colorJ      -2336.931   26.811   -87.162 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1159 on 53773 degrees of freedom
## Multiple R-squared:  0.9155, Adjusted R-squared:  0.9155
## F-statistic: 3.884e+04 on 15 and 53773 DF, p-value: < 2.2e-16

```

Para conseguir el modelo óptimo, construimos 4 modelos, a los que hemos ido añadiendo 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 utilizamos 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
## 16714.28
```

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 16714.28 dólares.

Ahora realizaremos un estudio de regresión logística, intentaremos predecir si el diamante tiene precio alto en función del peso, volumen En primer lugar reralizaremos un modelo donde tenemos en cuenta el precio alto o no del diamante y también si su peso es alto o no

```
diamonds_df$PA<-ifelse(diamonds_df$price>=uname(quantile(diamonds_df$price,.75)),1,0)
diamonds_df$CA<-ifelse(diamonds_df$carat>=uname(quantile(diamonds_df$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
## -1.8024   -0.3489   -0.3489    0.6625    2.3788
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.76845    0.02122 -130.5   <2e-16 ***
## CA          4.17322    0.03013   138.5   <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: 60497  on 53788  degrees of freedom
## Residual deviance: 31646  on 53787  degrees of freedom
## AIC: 31650
##
## Number of Fisher Scoring iterations: 5

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

```
## [1] "Valor OR: 64.9244545611017"
```

En el segundo modelo añadimos el volumen, de tal forma que ahora nuestro modelo tendrá el peso al to o no y el volumen alto o no para predecir si el precio es alto

```
diamonds_df$VA<-ifelse(diamonds_df$volume>=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
## -1.8494 -0.3462 -0.3462  0.6316  2.3852
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.78456   0.02129 -130.78  <2e-16 ***
## CA          2.22150   0.08562   25.95  <2e-16 ***
## VA          2.07377   0.08578   24.18  <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: 60497  on 53788  degrees of freedom
## Residual deviance: 31082  on 53786  degrees of freedom
## AIC: 31088
##
## Number of Fisher Scoring iterations: 5
```

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

```
## [1] "Valor OR: 9.22117595709824"
```

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

```
## [1] "Valor OR: 7.95476115565293"
```

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 curva 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 = 6.1112e-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

```

```

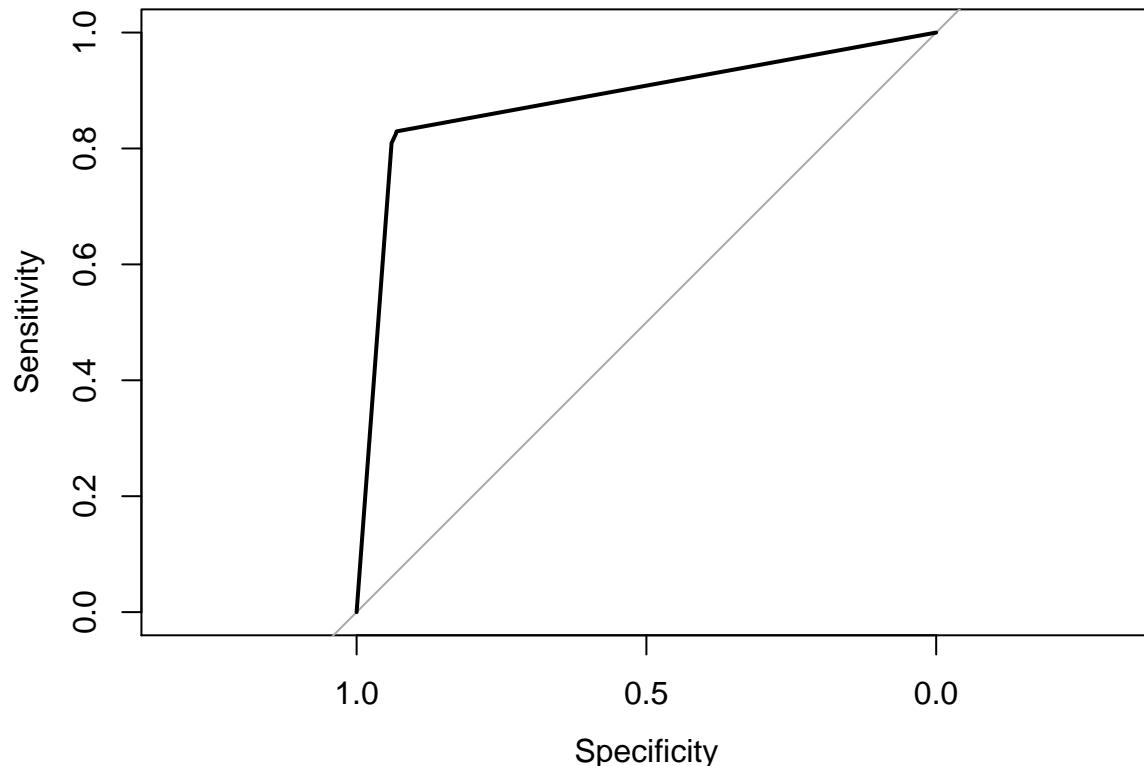
## Setting direction: controls < cases

```

```

plot (g)

```



```

auc (g)

```

```

## Area under the curve: 0.8831

```

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
## 0.8191669
```

Nos da una probabilidad de un 82% de que si el volumen y el peso son altos, el precio también será alto.

4.3.4. Análisis de multicolinealidad

A continuación, vamos a realizar un estudio sobre la multicolinealidad, siendo ésta, la posible dependencia de unas variables con respecto a otras. En este caso, como hemos estudiado, tenemos una fuerte correlación entre las variables peso y volumen, que al tratarse de diamantes de relativa pureza, la relación entre dichas variables no se vería afectada por posibles residuos en los diamantes.

El estudio de multicolinealidad se realiza mediante la raíz cuadrada del resultado de la función “VIF” que encontramos en la librería “faraway”

```
library(faraway)
```

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

```
## Registered S3 methods overwritten by 'lme4':
##   method           from
##   cooks.distance.influence.merMod car
##   influence.merMod        car
##   dfbeta.influence.merMod    car
##   dfbetas.influence.merMod   car
```

```
##
## Attaching package: 'faraway'
```

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

```
## The following object is masked from 'package:VIM':
## 
##   diabetes
```

```
sqrt(vif(modelo_logistico_2))
```

```
##          CA          VA
## 8.673653 8.614915
```

Con los datos disponibles, vemos que tenemos cierto grado de multicolinealidad, a pesar de ello, consideramos que, debido a que el volumen lo hemos calculado nosotros, con los datos de X, Y y Z, y dicha variable no tiene en cuenta otros factores, se mantendrá la variable Volumen para los cálculos anteriores y posteriores.

4.3.5. Árbol de clasificación

Como hemos observado en las secciones anteriores, tenemos una fuerte correlación entre las variables Precio, peso y volumen, por tanto, vamos a realizar un árbol de clasificación basado en esas variables para intentar predecir la categoría de precio del diamante, para ello, vamos a utilizar las categorías que habíamos descrito en el apartado 4.3.1

Para ello, creamos un nuevo dataset con las columnas a utilizar en el análisis y dividir los datos en entrenamiento y test, 2/3 para entrenar y 1/3 para probar.

```
#Creamos el Data Set
diamonds_tree<-cbind(diamonds_df, cuartil.price)

#Lo desordenamos
diamonds_tree<-diamonds_tree[sample(nrow(diamonds_tree)),]
head(diamonds_tree,10)

##      carat      cut color clarity depth table price     x     y     z volume
## 25402  1.55    Ideal     F    SI1   61.9    55 14220 7.47 7.42 4.61 255.52031
## 36940  0.35    Ideal     E    VVS1   61.8    56   967 4.53 4.50 2.79 56.87415
## 23337  0.36 Very Good     E    SI1   62.9    56   631 4.52 4.57 2.86 59.07730
## 2107   0.83 Very Good     F    SI1   63.1    58  3118 5.98 5.94 3.76 133.55971
## 53383  0.74    Ideal     I    VS1   62.5    56  2690 5.76 5.80 3.61 120.60288
## 11406  1.11    Ideal     E    SI2   62.8    55  5004 6.62 6.60 4.15 181.32180
## 51012  0.71 Very Good     F    SI1   64.0    58  2345 5.59 5.66 3.60 113.90184
## 4174   0.91    Ideal     G    SI2   62.7    57  3557 6.15 6.19 3.87 147.32510
## 48364  0.74      Fair     J    VS1   58.8    68  1982 5.91 5.80 3.44 117.91632
## 40612  0.54      Good     E    SI2   63.4    56  1163 5.14 5.18 3.27  87.06440
##      PA CA VA cuartil.price
## 25402  1  1  1      Alto
## 36940  0  0  0  Medio-Bajo
## 23337  0  0  0      Bajo
## 2107   0  0  0  Medio-Alto
## 53383  0  0  0  Medio-Alto
## 11406  0  1  1  Medio-Alto
## 51012  0  0  0  Medio-Bajo
## 4174   0  0  0  Medio-Alto
## 48364  0  0  0  Medio-Bajo
## 40612  0  0  0  Medio-Bajo

#Dividimos los datos de entrenamiento y test
indexes = sample(1:nrow(diamonds_tree), size=floor((2/3)*nrow(diamonds_tree)))

diamonds_train<-diamonds_tree[indexes,]
diamonds_test<-diamonds_tree[-indexes,]
```

A continuación, modelamos el árbol:

```
library(rpart)

## Warning: package 'rpart' was built under R version 3.6.3

##
## Attaching package: 'rpart'
```

```

## The following object is masked from 'package:faraway':
##
##      solder

library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 3.6.3

#Modelado

model.tree <- rpart(formula = cuartil.price ~ carat + volume + clarity + color , data = diamonds_train,
model.tree

## n= 35859
##
## node), split, n, loss, yval, (yprob)
##      * denotes terminal node
##
##  1) root 35859 26795 Alto (2.467163e-01 2.501743e-01 2.503416e-01 2.527678e-01)
##  2) volume< 112.9232 17158 8312 Bajo (5.155613e-01 4.519758e-01 3.228815e-02 1.748456e-04)
##     4) volume< 65.96801 9205 1394 Bajo (8.485606e-01 1.513308e-01 0.000000e+00 1.086366e-04) *
##     5) volume>=65.96801 7953 1591 Medio-Bajo (1.301396e-01 7.999497e-01 6.965925e-02 2.514774e-04)
##  3) volume>=112.9232 18701 9640 Alto (5.347308e-05 6.502326e-02 4.504037e-01 4.845195e-01)
##     6) volume< 188.5887 11830 4011 Medio-Alto (8.453085e-05 1.027050e-01 6.609467e-01 2.362637e-01)
##        12) carat< 0.985 5905 1269 Medio-Alto (1.693480e-04 1.998307e-01 7.850974e-01 1.490262e-02) *
##        13) carat>=0.985 5925 2742 Medio-Alto (0.000000e+00 5.907173e-03 5.372152e-01 4.568776e-01)
##           26) clarity=I1,SI1,SI2 3436 623 Medio-Alto (0.000000e+00 1.018626e-02 8.186845e-01 1.71129
##           27) clarity=IF,VS1,VS2,VVS1,VVS2 2489 370 Alto (0.000000e+00 0.000000e+00 1.486541e-01 8.5
##     7) volume>=188.5887 6871 605 Alto (0.000000e+00 1.455392e-04 8.790569e-02 9.119488e-01) *

#Reglas aplicadas
rpart.rules(model.tree)

```

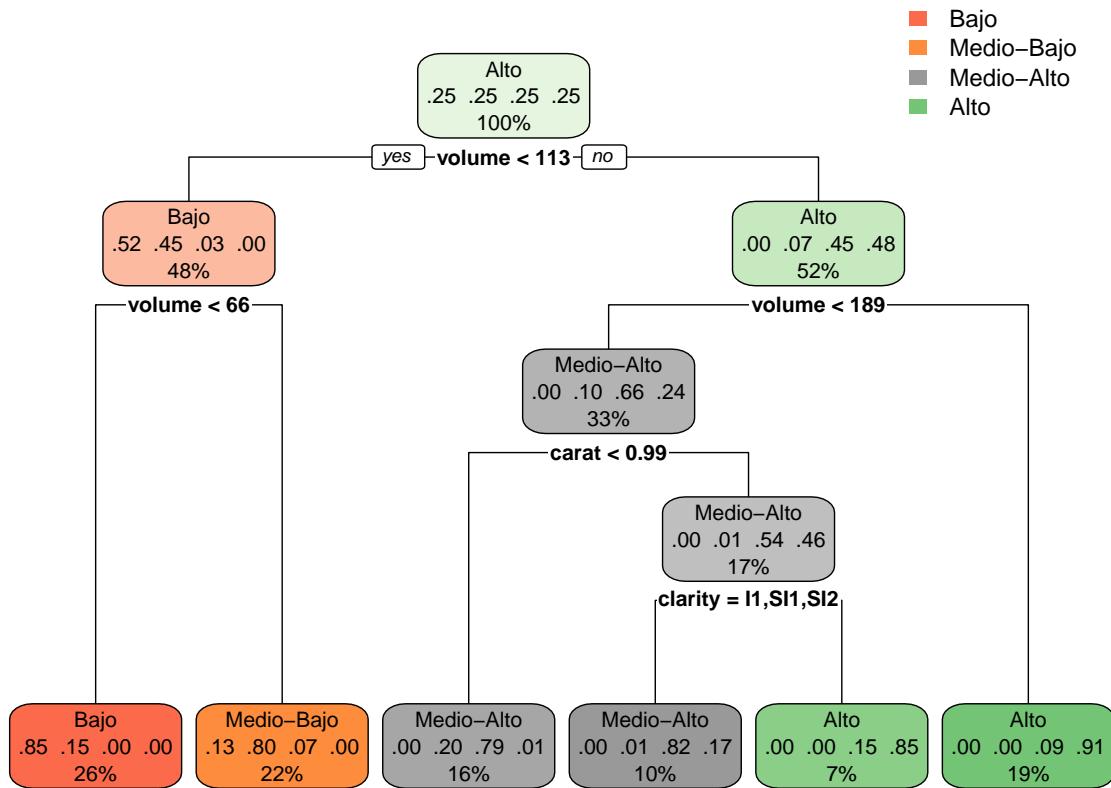
```

##  cuartil.price  Baj Med Med Alt
##      Bajo [.85 .15 .00 .00] when volume <    66
##      Medio-Bajo [.13 .80 .07 .00] when volume is  66 to 113
##      Medio-Alto [.00 .20 .79 .01] when volume is 113 to 189 & carat <  0.99
##      Medio-Alto [.00 .01 .82 .17] when volume is 113 to 189 & carat >= 0.99 & clarity is
##          Alto [.00 .00 .15 .85] when volume is 113 to 189 & carat >= 0.99 & clarity is IF or VS1 or
##          Alto [.00 .00 .09 .91] when volume >=      189

```

Representamos el árbol, para hacernos una idea:

```
rpart.plot(model.tree)
```



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?