Analisis Diamantes _ Estadistica Descriptiva

Ramon Ceballos

10/2/2021

Ejercicio de Diamantes en Python

Se pueden instalar paquetes de Python en RStudio cargando la librería "reticulate" (library(reticulate)) y despues emplear la instrucción py_install().

Una vez instalados, cargamos en RMarkdown diferentes paquetes de Python.

```
import numpy as np
import pandas as pd
import matplotlib
from ggplot import diamonds
```

```
## C:\Users\usuario\ANACON~1\lib\site-packages\ggplot\utils.py:81: FutureWarning: pandas.tslib is depre
## You can access Timestamp as pandas.Timestamp
##   pd.tslib.Timestamp,
## C:\Users\usuario\Documents\R\win-library\3.6\reticulate\python\rpytools\loader.py:24: FutureWarning:
##   level=level
```

```
matplotlib.style.use("ggplot")
```

Una vez cargado el dataset diamonds, lo exploramos.

```
#Dimensiones
print(diamonds.shape)

#Cinco primeras filas
```

```
## (53940, 10)
```

```
print(diamonds.head(5))
```

```
##    carat      cut color clarity  depth  table  price     x     y     z
## 0   0.23    Ideal     E     SI2   61.5   55.0    326  3.95  3.98  2.43
## 1   0.21  Premium     E     SI1   59.8   61.0    326  3.89  3.84  2.31
## 2   0.23     Good     E     VS1   56.9   65.0    327  4.05  4.07  2.31
## 3   0.29  Premium     I     VS2   62.4   58.0    334  4.20  4.23  2.63
## 4   0.31     Good     J     SI2   63.3   58.0    335  4.34  4.35  2.75
```

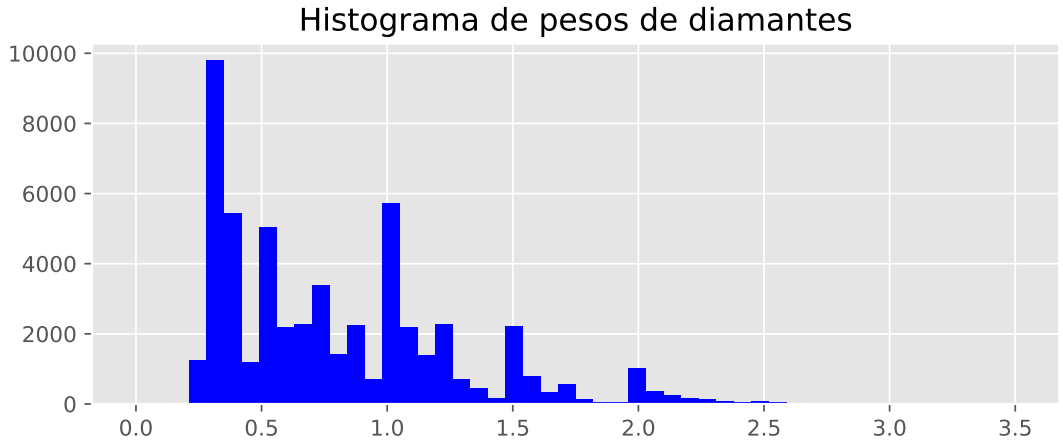
Histograma

Hacer el diagrama de una columna del dataset.

```
#Representamos el histograma de la columna "carat"
diamonds.hist(column="carat",
figsize=(8,3),
color="blue",
bins = 50,
range = (0,3.5))
```

```
## array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000000001F67D7B8>]],
##       dtype=object)
```

```
matplotlib.pyplot.title("Histograma de pesos de diamantes")
matplotlib.pyplot.show()
```



Filtro de outliers

Filtramos en python para aquellos diamantes con un peso superior a 3.5. Estos son muy raros ya que ni aparecen en el histograma anterior.

```
print(diamonds[diamonds["carat"]>3.5])
```

```
##        carat        cut color clarity  depth  table  price      x      y     z
## 23644   3.65       Fair     H      I1   67.1   53.0  11668   9.53   9.48  6.38
## 25998   4.01    Premium     I      I1   61.0   61.0  15223  10.14  10.10  6.17
## 25999   4.01    Premium     J      I1   62.5   62.0  15223  10.02   9.94  6.24
## 26444   4.00  Very Good     I      I1   63.3   58.0  15984  10.01   9.94  6.31
## 26534   3.67    Premium     I      I1   62.4   56.0  16193   9.86   9.81  6.13
## 27130   4.13       Fair     H      I1   64.8   61.0  17329  10.00   9.85  6.43
## 27415   5.01       Fair     J      I1   65.5   59.0  18018  10.74  10.54  6.98
## 27630   4.50       Fair     J      I1   65.8   58.0  18531  10.23  10.16  6.72
## 27679   3.51    Premium     J     VS2   62.5   59.0  18701   9.66   9.63  6.03
```

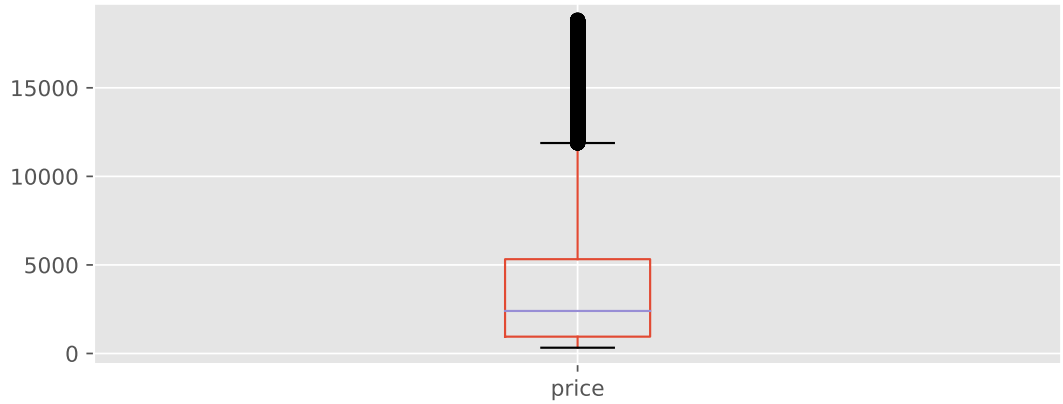
Este tipo de análisis sería conveniente para estudios específicos sobre este tipo de diamantes.

Boxplots

Representar boxplots en Python, para una variable ("precios").

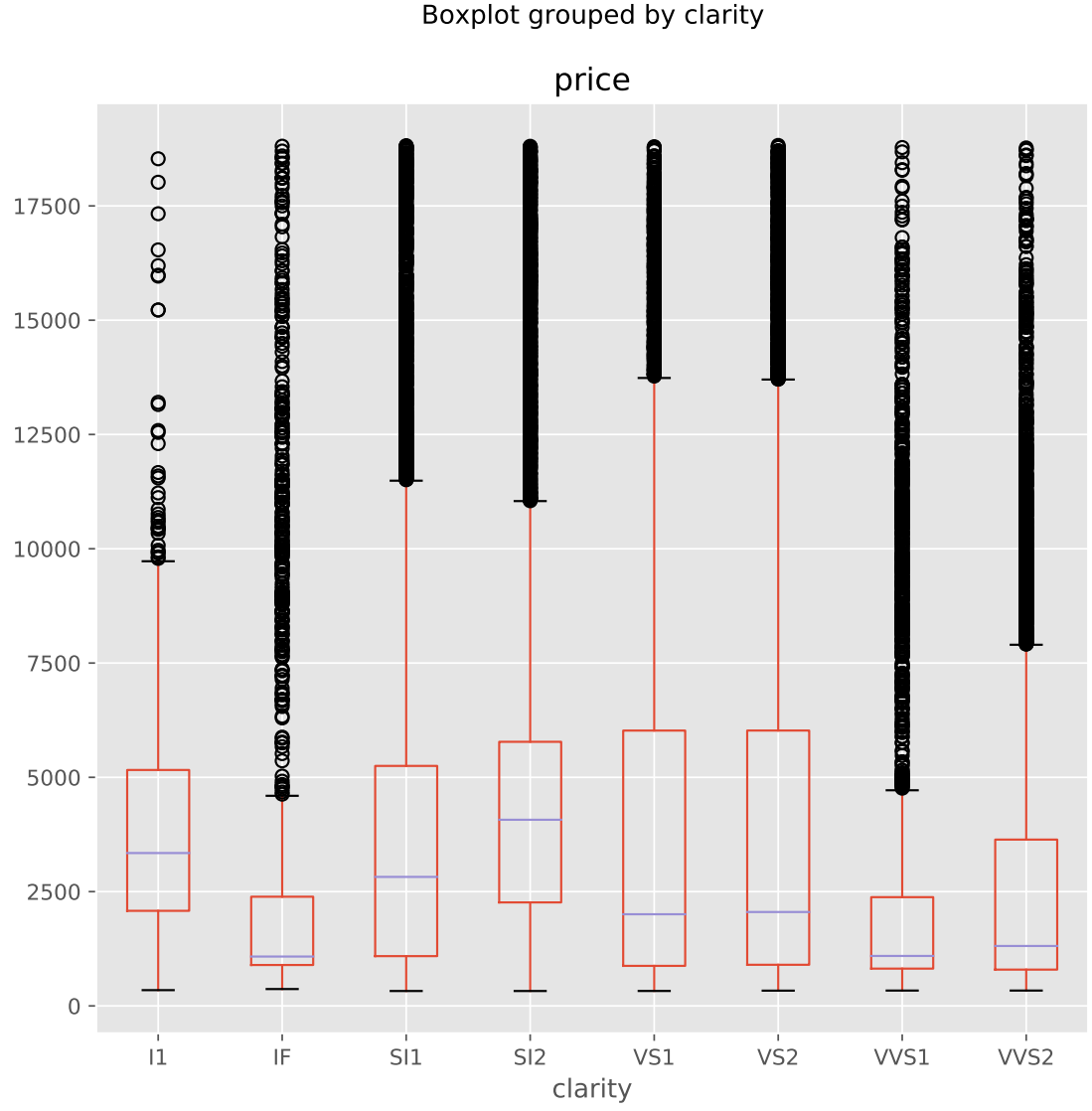
Se podría hacer en función de otra variable, para ver diferentes boxplots comparandolos.

```
#Limpia la figura anterior
matplotlib.pyplot.clf()
diamonds.boxplot(column = "price",
figsize = (8,8))

matplotlib.pyplot.show()
```



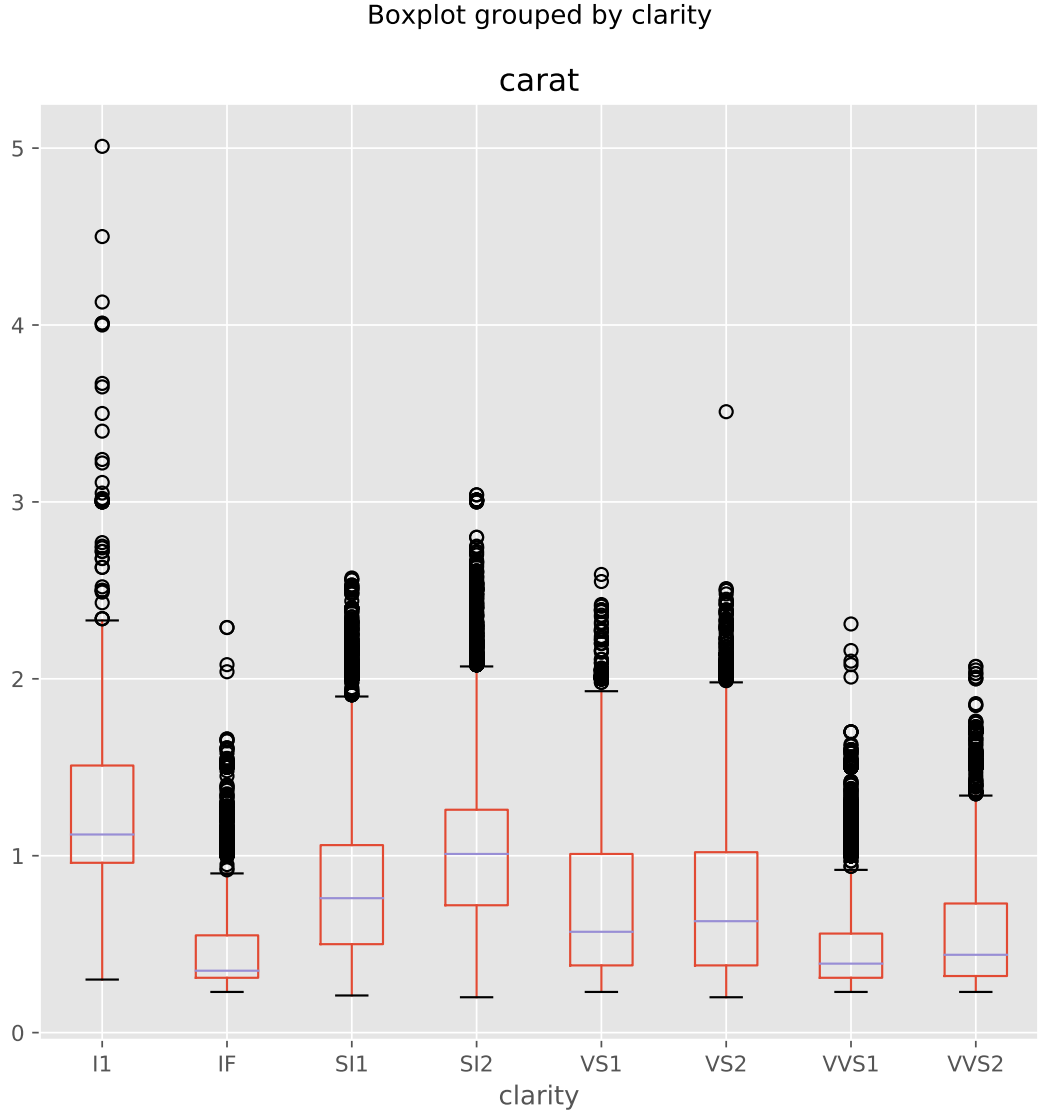
La claridaad de los diamantes vs el precio de los mismos.

```
matplotlib.pyplot.clf()
diamonds.boxplot(column = "price", by = "clarity",
figsize = (8,8))

matplotlib.pyplot.show()
```



Tamaño de los diamantes en relación a la claridad de los mismos.

```
matplotlib.pyplot.clf()
diamonds.boxplot(column = "carat", by = "clarity",
figsize = (8,8))

matplotlib.pyplot.show()
```



Densidades

Densidad de la distribución subyacente para la columna carat de pesos.

```
matplotlib.pyplot.clf()

diamonds["carat"].plot(kind="density",
figsize=(8,8),
xlim=(0,5))

matplotlib.pyplot.show()
```

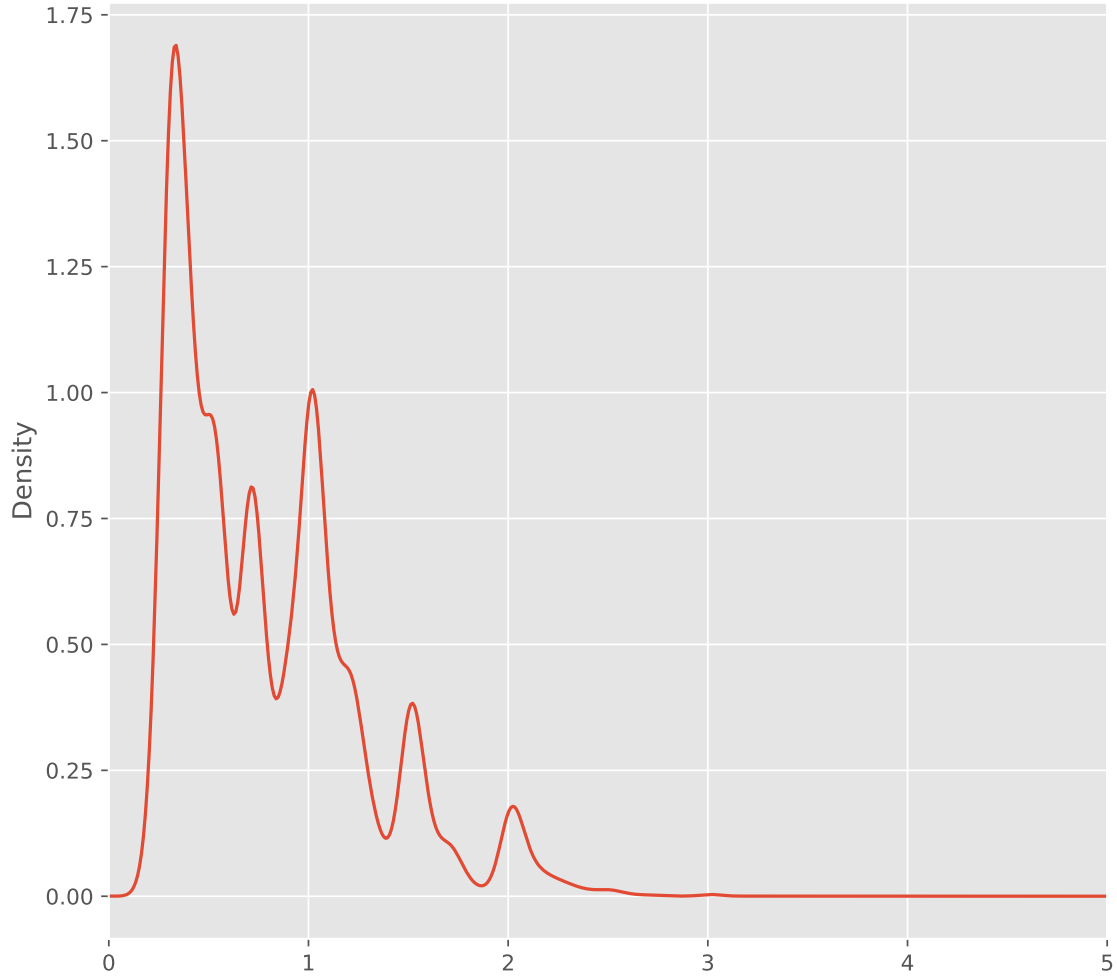


Tabla de frecuencias y Barplot

Con pandas para tablas de frecuencias.

Vamos a hacer la tabla de frecuencias absolutas y a posteirori dibujamos el barplot.

```
#Tabla de frecuencias absolutas
carat_table = pd.crosstab(index=diamonds["clarity"], columns="count")

print(carat_table)
```

```
## col_0    count
## clarity
```

```
## I1          741
## IF         1790
## SI1       13065
## SI2        9194
## VS1        8171
## VS2       12258
## VVS1       3655
## VVS2       5066
```

```
matplotlib.pyplot.clf()

carat_table.plot(kind="bar", figsize=(8,8))

matplotlib.pyplot.show()
```

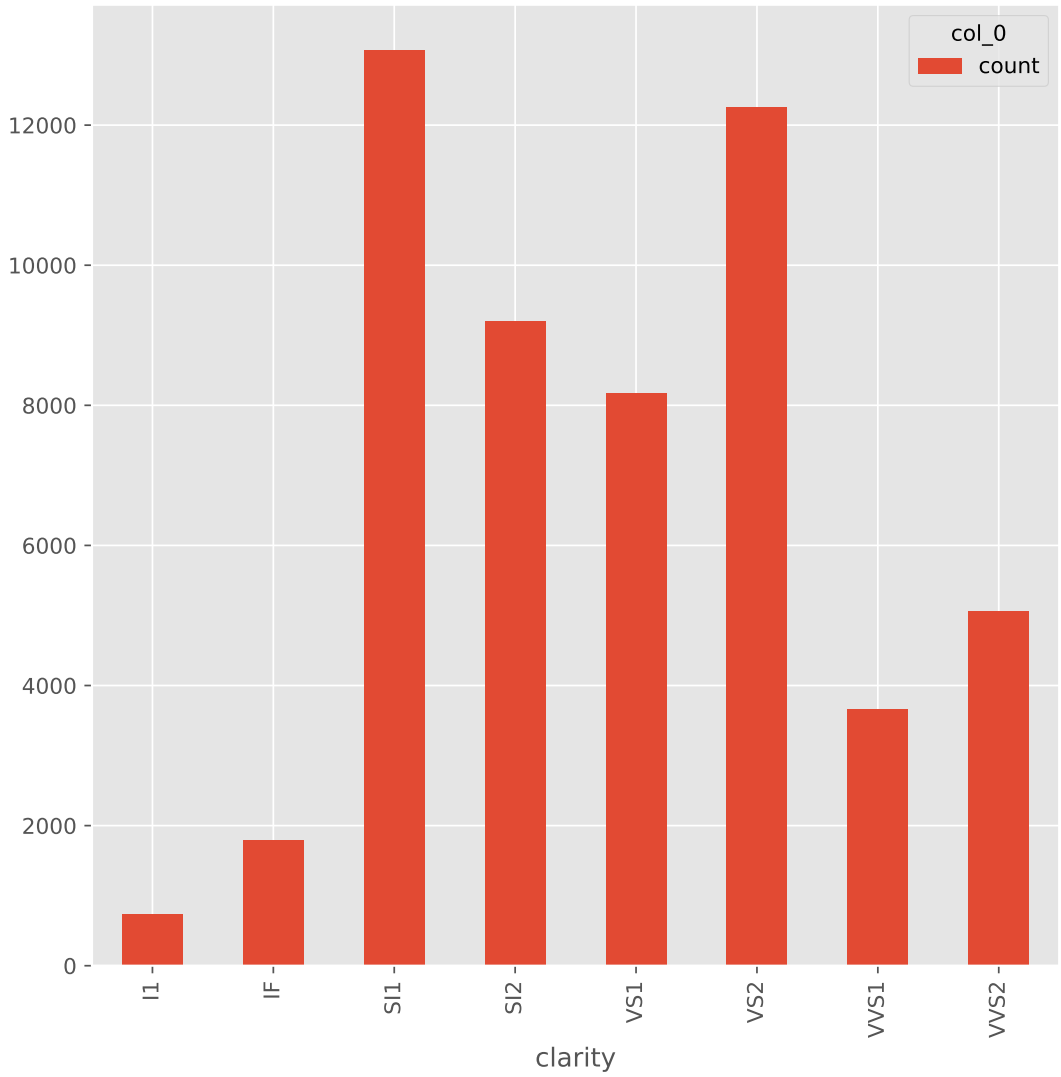
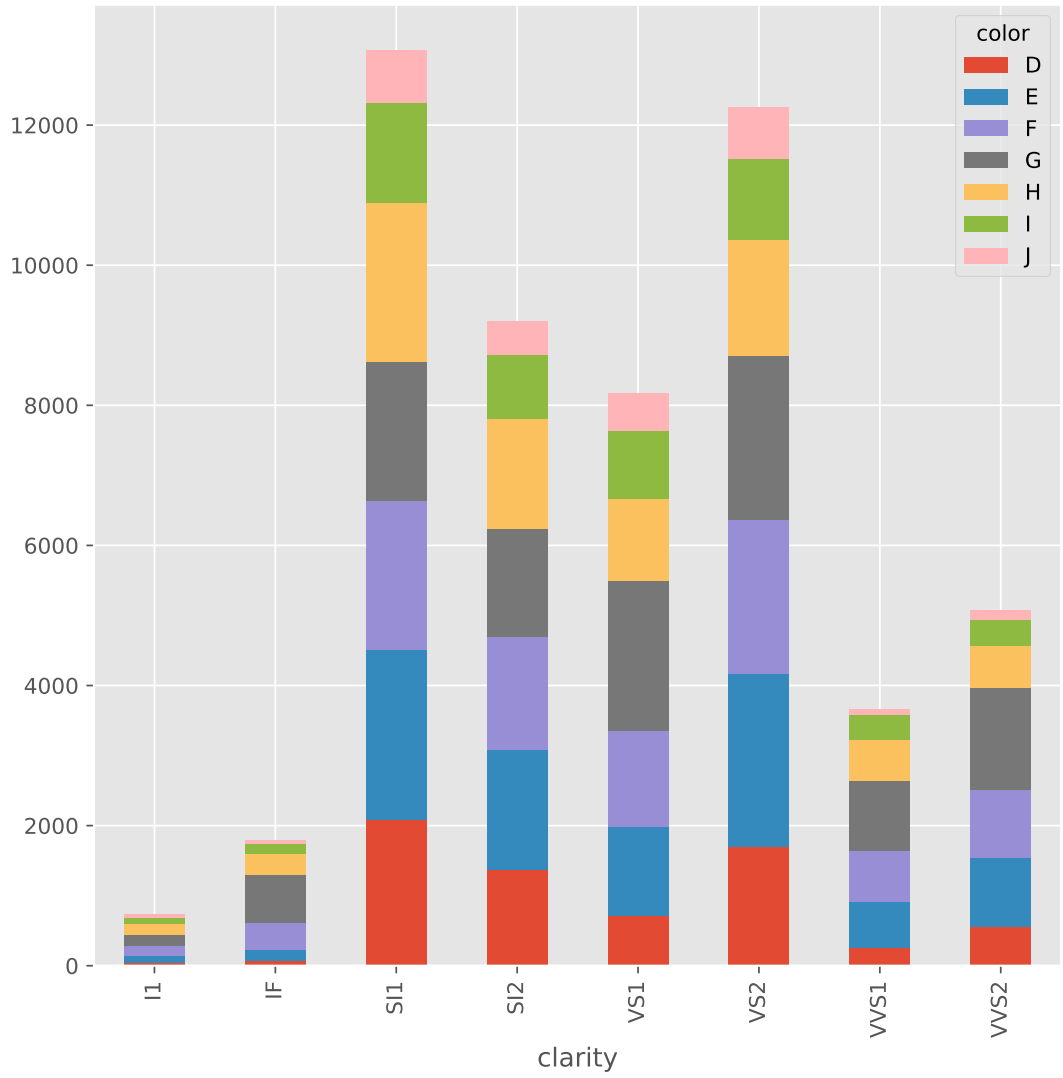


Tabla bidimensional de dos variables (claridad y color).

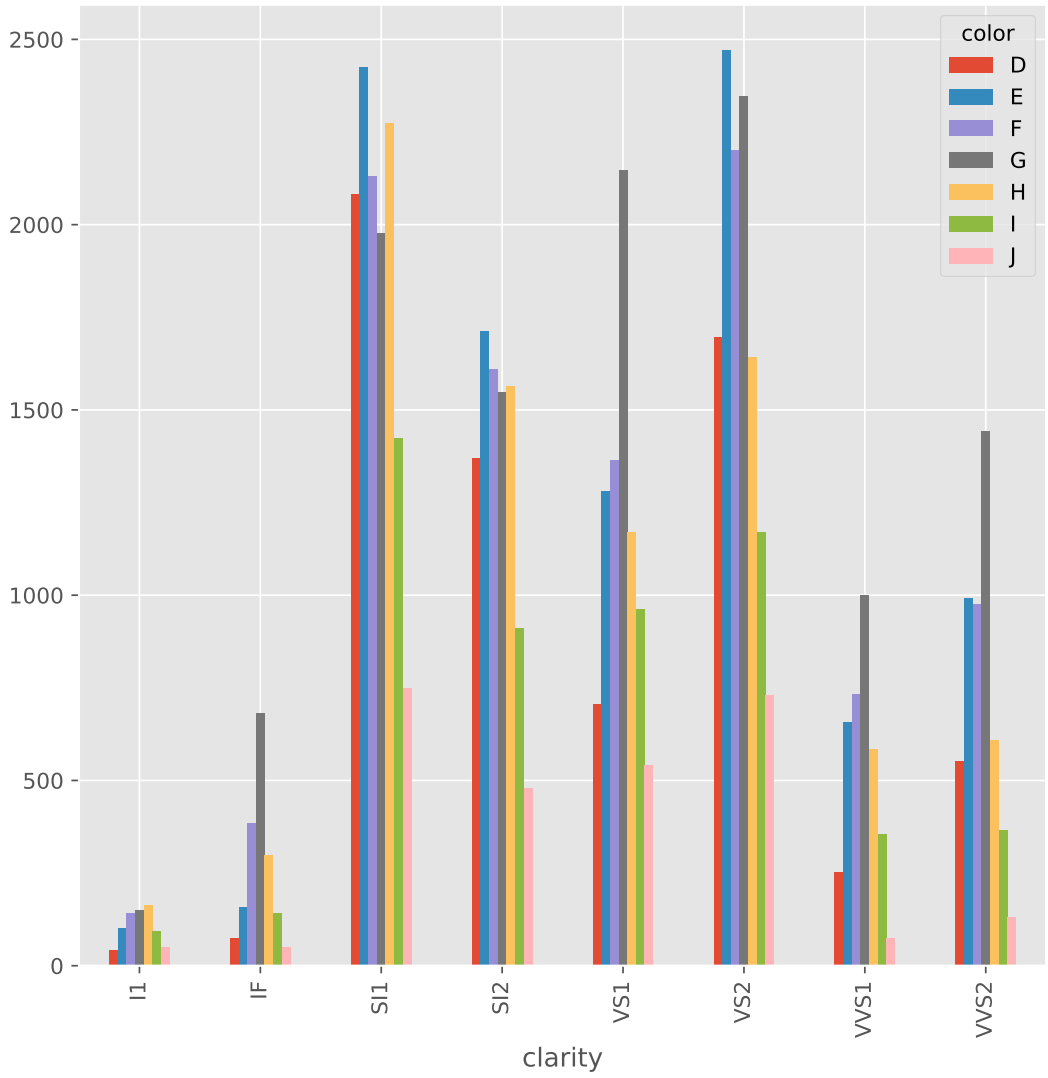
```
#Frecuencias absolutas para tabla bidimensional
carat_table_2 = pd.crosstab(index=diamonds["clarity"], columns=diamonds["color"])

print(carat_table_2)
```

```
## color       D     E     F     G     H     I    J
## clarity
## I1         42   102   143   150   162    92   50
## IF         73   158   385   681   299   143   51
## SI1      2083  2426  2131  1976  2275  1424  750
## SI2      1370  1713  1609  1548  1563   912  479
## VS1       705  1281  1364  2148  1169   962  542
## VS2      1697  2470  2201  2347  1643  1169  731
## VVS1      252   656   734   999   585   355   74
## VVS2      553   991   975  1443   608   365  131
```

```
matplotlib.pyplot.clf()

#Barras apiladas (stacked=True)
carat_table_2.plot(kind="bar", figsize=(8,8), stacked=True)

matplotlib.pyplot.show()
```




```
matplotlib.pyplot.clf()

#Barras no apiladas
carat_table_2.plot(kind="bar", figsize=(8,8), stacked=False)

matplotlib.pyplot.show()
```

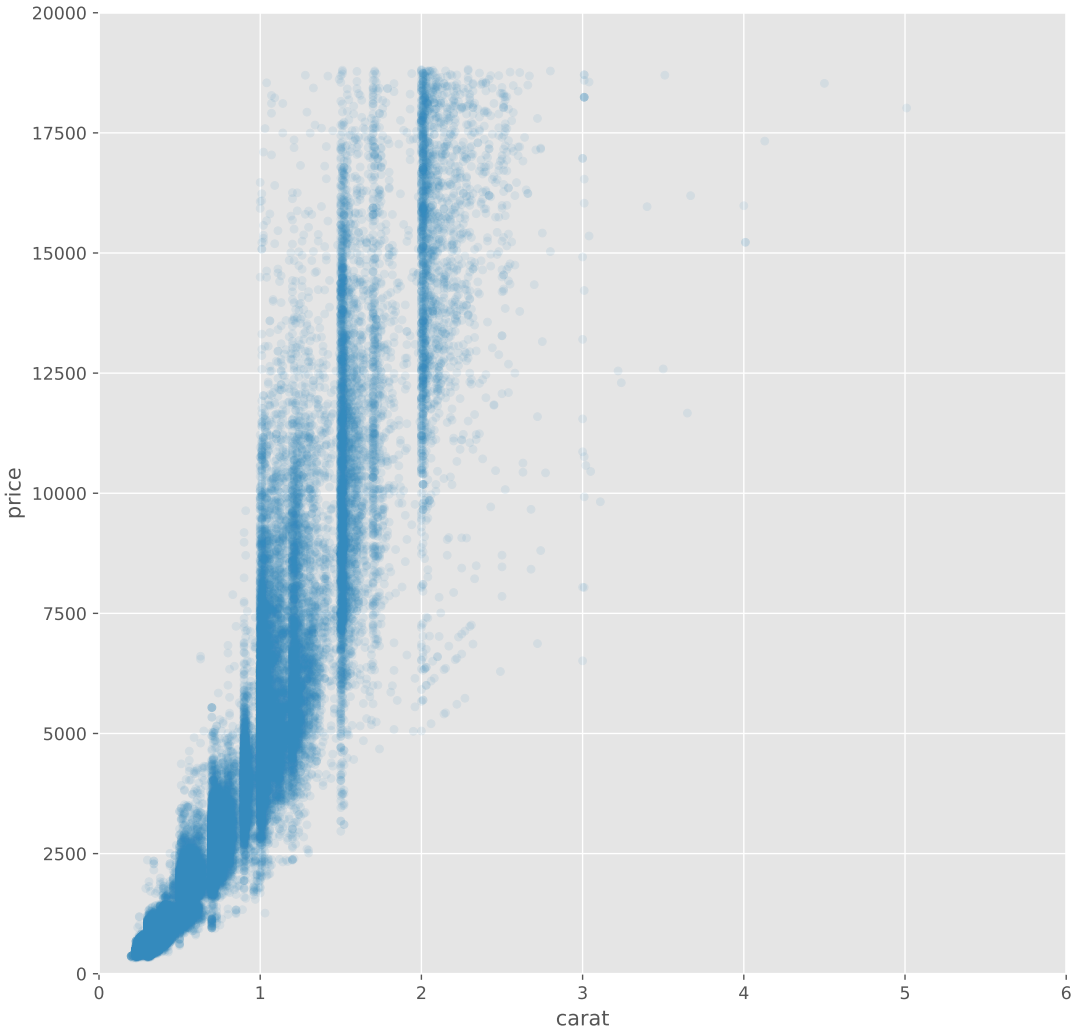


Scatterplot

Un gráfico de ptos de dispersión parat peso vs precio.

Poner transparencia a los ptos para visualizar mejor las zonas donde se aglutinan ptos.

```
matplotlib.pyplot.clf()

diamonds.plot(kind="scatter", x = "carat", y = "price", figsize=(10,10), ylim=(0,20000), xlim = (0,6), a
matplotlib.pyplot.show()
```


20000
17500
15000
12500
10000
7500
5000
2500
0
price
0
1
2
3
4
5
6
carat