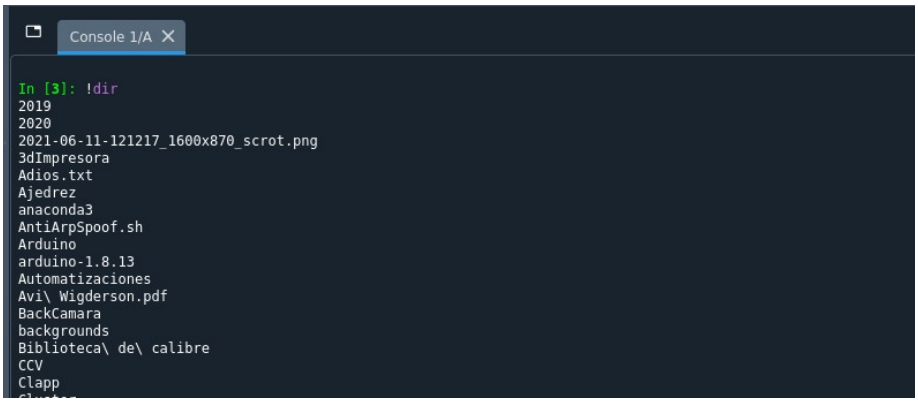
Fecha: 29 Septiembre 2021.
Versión: 0.1

Tip: Ver el contenido del directorio.

> > > !dir



Sigamos jugando con la librería pandas_datareader de [Yahoo! finance]

Lo primero fue instalar la librería.

Nota: https://anaconda.org/anaconda/pandas-datareader

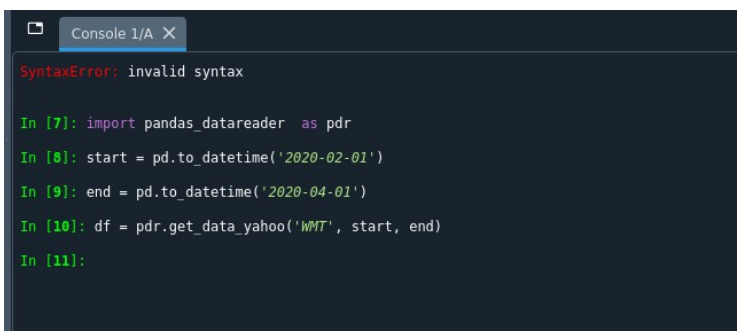
conda install pandas-datareader

> > > **import pandas as pd**
> > > **import pandas_datareader as pdr**

Extraer datos de acciones de Yahoo: (usaremos Walmart como ejemplo)

Símbolo de Walmart = WMT

> > > **start = pd.to_datetime(“2020-02-01”)**
> > > end = pd.to_datetime('2020-04-01')
> > > df = pdr.get_data_yahoo('WMT', start, end)



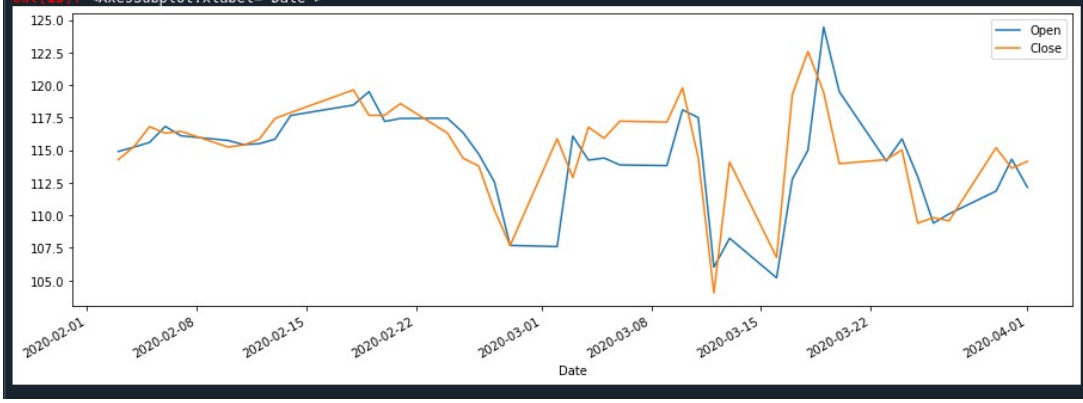
Exploramos el dataframe:

df.info()

```
Console 1/A

In [9]: end = pd.to_datetime('2020-04-01')

In [10]: df = pdr.get_data_yahoo('WMT', start, end)

In [11]: df.info()
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 42 entries, 2020-02-03 to 2020-04-01
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   High       42 non-null     float64
 1   Low        42 non-null     float64
 2   Open       42 non-null     float64
 3   Close      42 non-null     float64
 4   Volume     42 non-null     float64
 5   Adj Close  42 non-null     float64
dtypes: float64(6)
memory usage: 2.3 KB

In [12]:
```

Veamos ahora la gráfica de Open vs. Close Price:

```
under the Plots pane options menu.

In [13]: df[['Open', 'Close']].plot(figsize=(15,5))
Out[13]: <AxesSubplot:xlabel='Date'>
```




Notemos el uso de lista l= [“Open”, “Close”], ¿Que es df[[“Open”, Close”]]?. Si definimos df2=df[l], ¿Que es df2?. (Usemos type(df2)).


observemos como son los índices en df2. (¿Que tenemos?).

¿Que pasa si usamos df2.plot()?.

… Final de Yahoo Finance, por supuesto es solo una miradita.

Sigamos con el problema de la Multicolinealidad en Regresión lineal Multiple.

Usemos nuestra base de datos ya conocida “Problema3.csv”

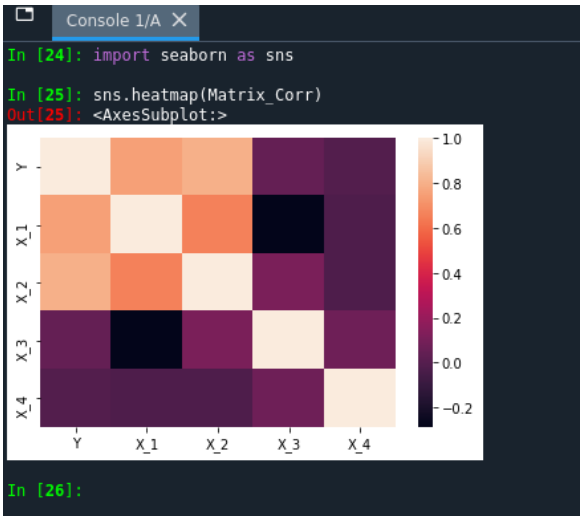
df= pd.read_csv(‘Problema3.csv’)

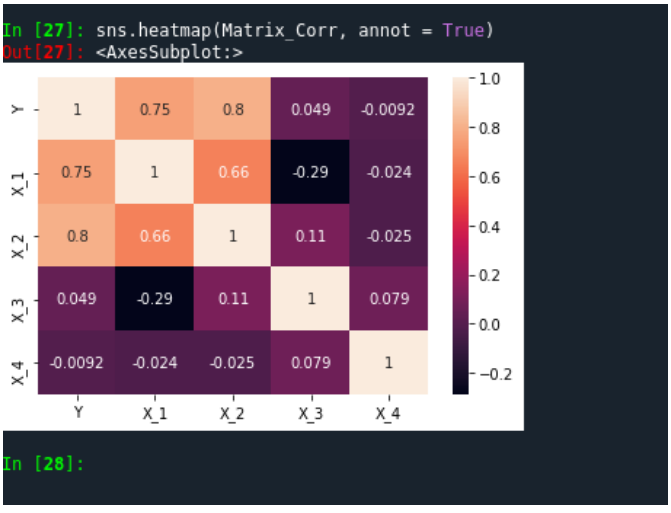
Matrix_Corr= df.corr()

```
In [18]: Matrix_Corr
Out[18]:
              Y       X_1       X_2       X_3       X_4
Y     1.000000  0.751479  0.802857  0.048848 -0.009197
X_1   0.751479  1.000000  0.660456 -0.287566 -0.023559
X_2   0.802857  0.660456  1.000000  0.112739 -0.025328
X_3   0.048848 -0.287566  0.112739  1.000000  0.078914
X_4  -0.009197 -0.023559 -0.025328  0.078914  1.000000

In [19]: type(Matrix_Corr)
Out[19]: pandas.core.frame.DataFrame

In [20]:
```

Notemos que los índices del dataframe Matrix_Corr son los nombres de las columnas.

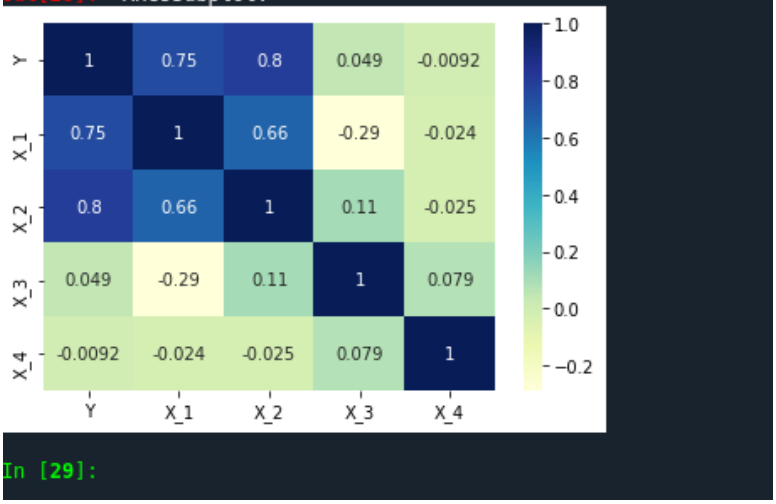
Usemos ahora la librería seaborn y su método heatmap para ver “mejor” la matriz de correlación





Cambiemos la paleta de Colores en heatmap.

```
In [28]: sns.heatmap(Matrix_Corr, annot = True, cmap="YlGnBu")
Out[28]: <AxesSubplot:>
```



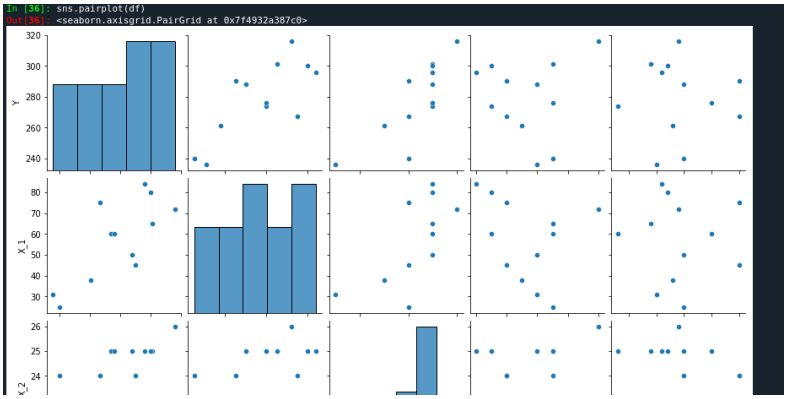

```
In [29]:
```

Finalmente veamos algo muy interesante:
Tip: (Ayudas)

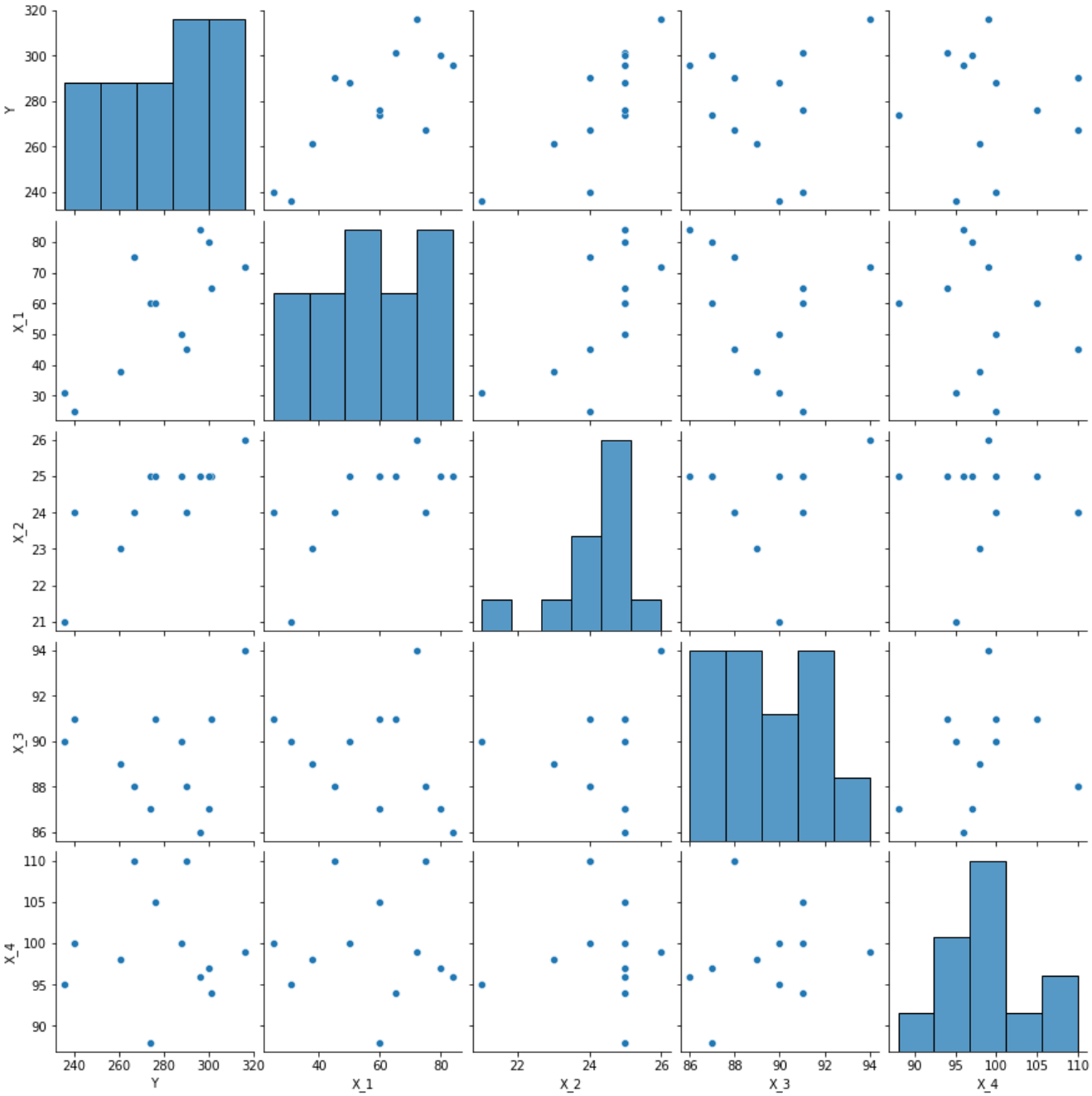
sns.boxplot?

o
help(sns.boxplot)

> > > sns.pairplot(df)



Resultado Final:



Resumen de Regresión lineal:

```
In [1]: import pandas as pd

In [2]: import statsmodels.formula.api as sm

In [3]: df = pd.read_csv("Problema3.csv")

In [4]: df
Out[4]:
      Y  X_1  X_2  X_3  X_4
0   240   25   24   91  100
1   236   31   21   90   95
2   290   45   24   88  110
3   274   60   25   87   88
4   301   65   25   91   94
5   316   72   26   94   99
6   300   80   25   87   97
7   296   84   25   86   96
8   267   75   24   88  110
9   276   60   25   91  105
10  288   50   25   90  100
11  261   38   23   89   98

In [5]: l="Y ~ X_1 + X_2 + X_3 + X_4"

In [6]: l
Out[6]: 'Y ~ X_1 + X_2 + X_3 + X_4'

In [7]: resultado = sm.ols(formula=l, data=df).fit()
```

Console 1/A

```
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.745
Model:                            OLS   Adj. R-squared:                  0.599
Method:                 Least Squares   F-statistic:                     5.106
Date:                Wed, 29 Sep 2021   Prob (F-statistic):             0.0303
Time:                        18:51:31   Log-Likelihood:                -46.745
No. Observations:                  12   AIC:                             103.5
Df Residuals:                       7   BIC:                             105.9
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   -102.7132    207.859     -0.494      0.636    -594.221     388.795
X_1            0.6054      0.369      1.641      0.145      -0.267       1.478
X_2            8.9236      5.301      1.684      0.136      -3.610      21.457
X_3            1.4375      2.392      0.601      0.567      -4.218       7.093
X_4            0.0136      0.734      0.019      0.986      -1.722       1.749
==============================================================================
Omnibus:                        0.007   Durbin-Watson:                   1.772
Prob(Omnibus):                  0.996   Jarque-Bera (JB):                0.207
Skew:                          -0.039   Prob(JB):                        0.902
Kurtosis:                       2.362   Cond. No.                     6.82e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 6.82e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
"""

In [9]:
```

¿Que ocurre si quitamos X_1 y asumimos que la constante es cero?

```
In [9]: l="Y ~ X_1 + X_3 + X_4 -1"

In [10]: resultado = sm.ols(formula=l, data=df).fit()

In [11]: resultado.summary()
/home/jorge/.local/lib/python3.8/site-packages/scipy/stats/stats.py:1603: UserWarning: kurtosistest only valid for n>=20 ... c
n=12
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
Out[11]:
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared (uncentered):                   0.997
Model:                            OLS   Adj. R-squared (uncentered):              0.997
Method:                 Least Squares   F-statistic:                              1168.
Date:                Wed, 29 Sep 2021   Prob (F-statistic):                    5.63e-12
Time:                        20:37:29   Log-Likelihood:                         -48.832
No. Observations:                  12   AIC:                                      103.7
Df Residuals:                       9   BIC:                                      105.1
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
X_1            1.0339      0.249      4.144      0.003       0.470       1.598
X_3            2.5770      0.820      3.141      0.012       0.721       4.433
X_4           -0.1052      0.738     -0.143      0.890      -1.774       1.564
==============================================================================
Omnibus:                        0.140   Durbin-Watson:                   2.061
Prob(Omnibus):                  0.932   Jarque-Bera (JB):                0.087
Skew:                           0.092   Prob(JB):                        0.957
Kurtosis:                       2.625   Cond. No.                         33.8
```

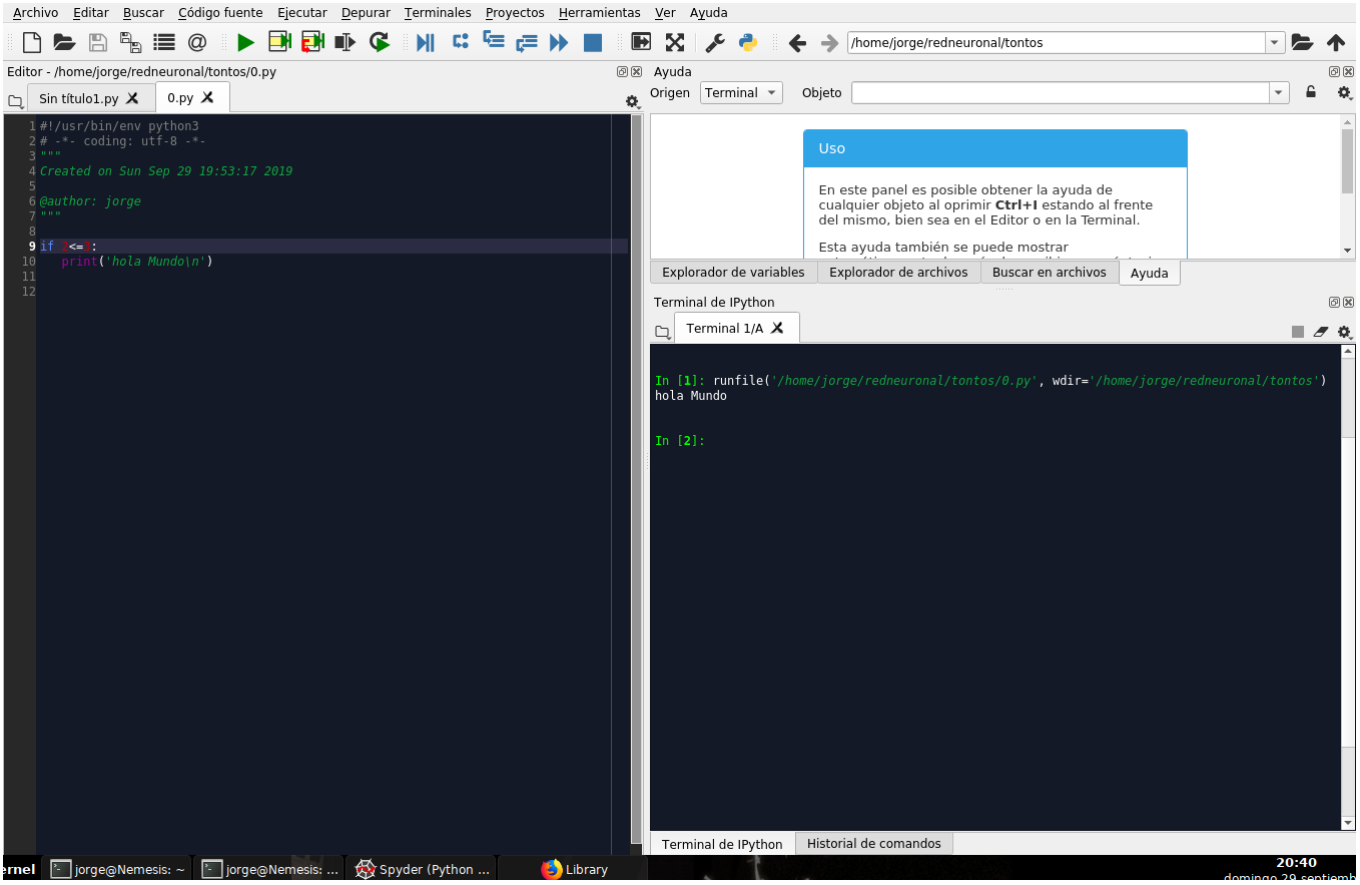
IPython console | History

¿Alguna duda?

Pequeña Introducción a Programación python

Mi primer Programa en Python:

Un programa sencillo de python, imprimir “Hola mundo”.



Notas: **En mi caso el programa lo llame 0.py .**

Se pude correr en la terminal ipython asi**:**

>>> %run 0.py

Estructura de control

1. Sentencias condicionales.

if

if condición: # si condición es True se realiza la acción 1.

accion 1

ejemplo 0:

>>> if 2<=3:

print (‘Hola mundo’)

>>> ‘Hola Mundo!

Sencillo!!!.

¿Que occurre si ponemos? (Consola de python)

>>> if 2<=3:

>>> print(‘hola Mundo)

Marca un error

>>> IndentationError: expected an indented block.

Python necesita simpre esa sangría en estas estructuras

ejemplo 1:

```
>>> if 2==2:
        print(‘soy un analista ...\n’)     # \n dara un salto de linea.
        print(“de datos”)
```

Veamos el sigueinte ejemplo:

```
>> if 2<1:
      print(“...voy adelante”)
 print (“… ¿que paso?”)
```

>>> no hay nada impreso…

Ahora si

>>> if 2<1:

print(“...voy adelante”)

>>>print (“… ¿que paso?”) #!! notar la falta de sangria en esta linea, **esta parte ya no forma parte de sentencia if….**

>>>...¿que paso? .

if … else

Ejemplo 3.

>>> x=3.

>>> if x==2:

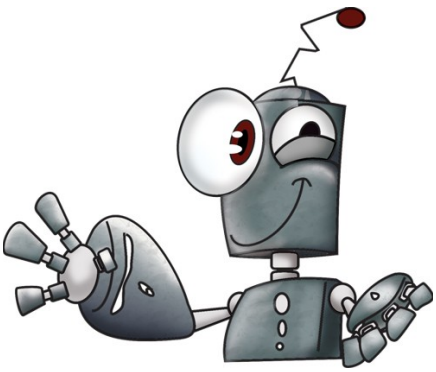
print(“el número %d es igual a 2 ” %x)

else:

print(“el número x es igual a %d ” %x)

!!! Cuidado con los sangrías!!!!

!!!Por hoy, terminamos programación en Python!!



Sigamos con más análisis de datos

Ejemplo muy interesante: Los Datos corresponden a mediciones de 100 niños nacidos con bajo peso (es decir, menos de 1500gr.) en Boston Massachusetts. Para dichos bebés se miden varias variables. La variable que nos interesa es el perímetro cefálico al nacer (medido en cm.). Los datos están en el archivo **low_birth_weight_infants.txt**, la variable **headcirc** es la que contiene los datos del perímetro cefálico.

Asumamos que entra una mamá con su bebé recién nacido al consultorio de niños de bajo peso y quiero predecir su perímetro cefálico con la información de la muestra de los 100 niños.

Hagamos un poco de análisis de datos con lo que sabemos

import pandas as pd
df=pd.read_csv("**low_birth_weight_infants.txt**" ,set="\s+")
df.info()

```
In [7]: import pandas as pd

In [8]: df=pd.read_csv("low_birth_weight_infants.txt", sep="\s+")

In [9]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   headcirc  100 non-null    int64
 1   length    100 non-null    int64
 2   gestage   100 non-null    int64
 3   birthwt   100 non-null    int64
 4   momage    100 non-null    int64
 5   toxemia   100 non-null    int64
dtypes: int64(6)
memory usage: 4.8 KB

In [10]:
```

Definamos Y = perímetro cefálico (en cm.) de un bebé recién nacido con bajo peso. Estamos interesados en la media poblacional $\mathrm{E}(\mathrm{Y})=\mu.$ Sabemos que la media muestral $\bar{Y}$ es el mejor estimador que podamos dar para la media poblacional, veamos su valor:

```
In [10]: df.describe()
Out[10]:
         headcirc      length     gestage      birthwt      momage     toxemia
count  100.000000  100.000000  100.00000   100.000000  100.000000  100.00000
mean    26.450000   36.820000   28.89000  1098.850000   27.730000    0.21000
std      2.532117    3.571435    2.53419   269.993317    5.982896    0.40936
min     21.000000   20.000000   23.00000   560.000000   14.000000    0.00000
25%     25.000000   35.000000   27.00000   880.000000   23.000000    0.00000
50%     27.000000   38.000000   29.00000  1155.000000   28.000000    0.00000
75%     28.000000   39.000000   31.00000  1326.250000   32.000000    0.00000
max     35.000000   43.000000   35.00000  1490.000000   41.000000    1.00000

In [11]:
```

Tenemos así une media muestral igual a 26.45 cm con una desviación muestral de 2.53.

Podemos construir intervalos de confianza del 95% para la media poblacional de Y, usando la muestras.

¿Que necesitamos?, pues las siguientes librerías

```
import numpy as np
import scipy.stats as st
a=df["headcirc"]
st.t.interval(0.95, len(a)-1, loc=np.mean(a), scale=st.sem(a))
```

```
In [8]: import numpy as np

In [9]: import scipy.stats as st

In [10]: a=df["headcirc"]

In [11]: st.t.interval(0.95, len(a)-1, loc=np.mean(a), scale=st.sem(a))
Out[11]: (25.94757306586406, 26.95242693413594)

In [12]: st.t.interval(0.99, len(a)-1, loc=np.mean(a), scale=st.sem(a))
Out[12]: (25.784963427136237, 27.11503657286376)

In [13]:
```

Pero tenemos información adicional para hacer una mejor predicción sobre el perímetro cefálico.

```
In [13]: df.corr()
Out[13]:
          headcirc    length   gestage   birthwt    momage   toxemia
headcirc  1.000000  0.712733  0.780692  0.798837  0.132119  0.132043
length    0.712733  1.000000  0.675231  0.815552  0.217993  0.109024
gestage   0.780692  0.675231  1.000000  0.659938  0.265840  0.411968
birthwt   0.798837  0.815552  0.659938  1.000000  0.154572  0.011803
momage    0.132119  0.217993  0.265840  0.154572  1.000000  0.114119
toxemia   0.132043  0.109024  0.411968  0.011803  0.114119  1.000000

In [14]:
```

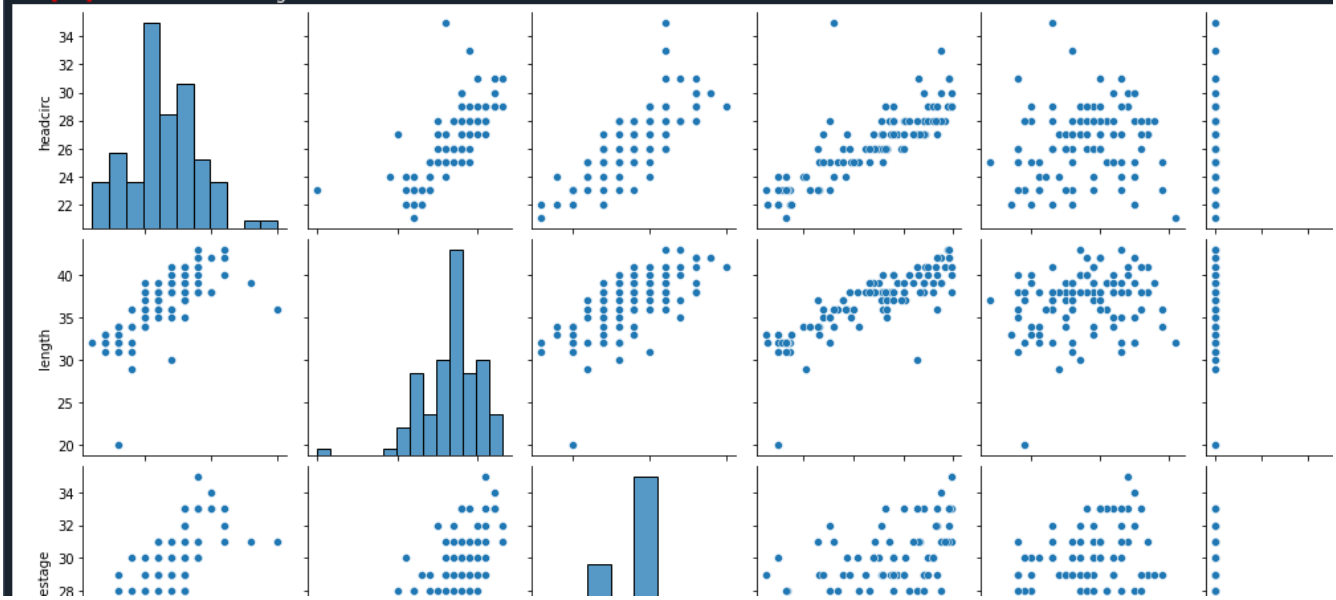
La variable **gestage** (**edad gestacional**, es decir, duración del embarazo medido en semanas).

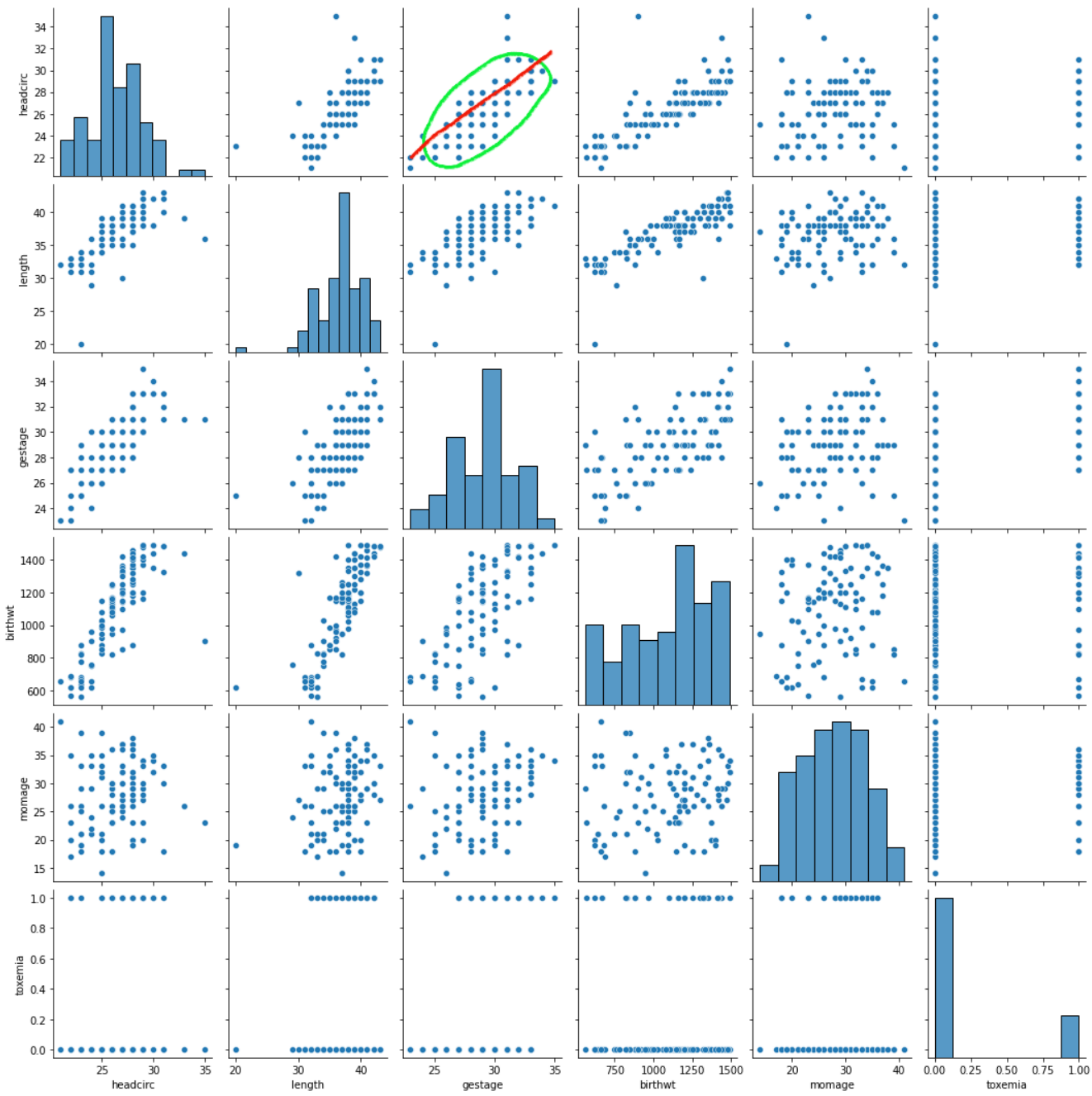
Objetivo: Ver si podemos predecir de una mejor manera el perímetro cefálico de un bebé al nacer si conocemos su edad gestacional.

Veamos como se ve todo esto usando la librería seaborn

import seaborn as sms

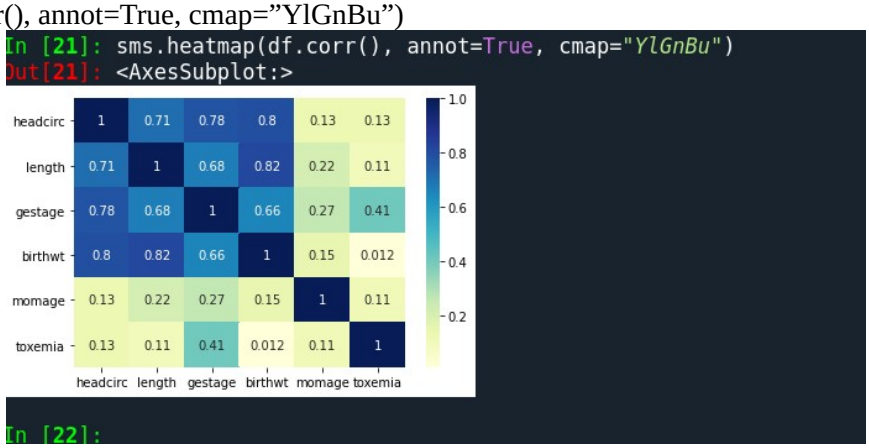
```
In [19]: import seaborn as sms

In [20]: sms.pairplot(df)
Out[20]: <seaborn.axisgrid.PairGrid at 0x7fbbae872f40>
```





Recordado la función heatmap de seaborn

sms.heamap(df.corr(), annot=True, cmap="YlGnBu")



Construyendo el modelo de regresión lineal:

```
In [31]: import statsmodels.formula.api as sm

In [32]: resultado=sm.ols(formula="headcirc ~ gestage", data=df).fit()

In [33]: resultado.summary()
Out[33]:
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                headcirc   R-squared:                       0.609
Model:                            OLS   Adj. R-squared:                  0.605
Method:                 Least Squares   F-statistic:                     152.9
Date:                Wed, 06 Oct 2021   Prob (F-statistic):           1.00e-21
Time:                        17:07:03   Log-Likelihood:                -187.28
No. Observations:                 100   AIC:                             378.6
Df Residuals:                      98   BIC:                             383.8
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      3.9143      1.829      2.140      0.035       0.284       7.544
gestage        0.7801      0.063     12.367      0.000       0.655       0.905
==============================================================================
Omnibus:                       23.475   Durbin-Watson:                   2.037
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               53.083
Skew:                           0.849   Prob(JB):                     2.97e-12
Kurtosis:                       6.140   Cond. No.                         334.
==============================================================================
```

Podemos asumir que la constate es cero en el modelo:

```
In [34]: resultado=sm.ols(formula="headcirc ~ gestage -1", data=df).fit()

In [35]: resultado.summary()
Out[35]:
<class 'statsmodels.iolib.summary.Summary'>
"""
                                 OLS Regression Results
=======================================================================================
Dep. Variable:                headcirc   R-squared (uncentered):                   0.996
Model:                            OLS   Adj. R-squared (uncentered):              0.996
Method:                 Least Squares   F-statistic:                          2.684e+04
Date:                Wed, 06 Oct 2021   Prob (F-statistic):                    2.43e-122
Time:                        17:11:53   Log-Likelihood:                         -189.57
No. Observations:                 100   AIC:                                      381.1
Df Residuals:                      99   BIC:                                      383.7
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
gestage        0.9145      0.006    163.819      0.000       0.903       0.926
==============================================================================
Omnibus:                       14.055   Durbin-Watson:                   2.044
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               23.631
Skew:                           0.581   Prob(JB):                     7.39e-06
Kurtosis:                       5.079   Cond. No.                         1.00
==============================================================================
```

Nota: Si solo queremos ver los valor de los coeficientes:

resultado.params

```
In [38]: resultado.params
Out[38]:
gestage    0.914517
dtype: float64

In [39]: resultado=sm.ols(formula="headcirc ~ gestage", data=df).fit()

In [40]: resultado.params
Out[40]:
Intercept    3.914264
gestage      0.780053
dtype: float64
```

Para ver las predicciones:

resultado.predict()

```
In [46]: resultado.predict()
Out[46]:
array([26.53580585, 28.09591217, 29.6560185 , 28.09591217, 27.31585901,
       23.4155932 , 24.97569952, 26.53580585, 25.75575269, 26.53580585,
       24.19564636, 27.31585901, 26.53580585, 26.53580585, 26.53580585,
       26.53580585, 26.53580585, 29.6560185 , 29.6560185 , 26.53580585,
       25.75575269, 27.31585901, 24.97569952, 29.6560185 , 28.87596533,
       25.75575269, 26.53580585, 25.75575269, 26.53580585, 27.31585901,
       28.09591217, 27.31585901, 28.09591217, 26.53580585, 24.97569952,
       24.97569952, 24.97569952, 28.87596533, 28.09591217, 25.75575269,
       27.31585901, 26.53580585, 25.75575269, 28.09591217, 24.97569952,
       23.4155932 , 27.31585901, 25.75575269, 25.75575269, 23.4155932 ,
       21.85548687, 24.97569952, 25.75575269, 24.97569952, 24.97569952,
       24.19564636, 23.4155932 , 21.85548687, 24.19564636, 22.63554004,
       26.53580585, 26.53580585, 24.97569952, 27.31585901, 27.31585901,
       28.87596533, 29.6560185 , 24.97569952, 28.09591217, 24.19564636,
       24.97569952, 24.97569952, 31.21612482, 25.75575269, 27.31585901,
       28.09591217, 27.31585901, 24.97569952, 23.4155932 , 23.4155932 ,
       24.19564636, 26.53580585, 26.53580585, 30.43607166, 27.31585901,
       26.53580585, 29.6560185 , 27.31585901, 26.53580585, 22.63554004,
       29.6560185 , 23.4155932 , 28.87596533, 28.09591217, 28.09591217,
       28.09591217, 26.53580585, 28.87596533, 29.6560185 , 25.75575269])
```

¿Como calculamos los errores?

Queremos ahora hacer predicciones para 25, 29 y 33 semanas de gestación.

¿Como hacemos esto?, !Fácil!

resultado.predict(pd.DataFrame({“gestage”:[25,29,33]}))

```
In [97]: resultado.predict(pd.DataFrame({"gestage":[25,29,33]}))
Out[97]:
0    23.415593
1    26.535806
2    29.656018
dtype: float64

In [98]:
```

Nota: Otra forma de hacerlo

```
In [99]: Nuevos_Valores = dict(gestage=[25,29,33])

In [100]: type(Nuevos_Valores)
Out[100]: dict

In [101]: resultado.predict(pd.DataFrame(Nuevos_Valores))
Out[101]:
0    23.415593
1    26.535806
2    29.656018
dtype: float64

In [102]:
```

¿Alguna duda?