ntic master
revolucionamos la comunicación

UNIVERSIDAD COMPLUTENSE DE MADRID

Programación. Python

Dataframes con pandas

Cristóbal Pareja Flores

1. Operaciones básicas

1.a Construcción de tablas

```
import pandas as pd
```

```
# Creación de una tabla o dataframe
# dando los nombres de sus columnas y valores a sus elementos:

tabla_a = pd.DataFrame({"A": [1, 3, 5], "B": [2, 4, 6]})
print(tabla_a)
```

```
   A  B
0  1  2
1  3  4
2  5  6
```

```
# El valor `None` permite forzar un dato missing (`NaN`, *Not A Number* en inglés):

tabla_b = pd.DataFrame({"B": [2, 4], "C": [20, None]})
print(tabla_b)
```

```
   B     C
0  2  20.0
1  4   NaN
```

```
# A veces, necesitaremos crear un dataframe con números aleatorios:

import numpy as np
np.random.seed(0) # damos una semilla concreta para que la ejecución sea reproducible

# Se generan 3x4 números aleatorios N(0, 1):

tabla_c = pd.DataFrame(np.random.randn(3, 4), columns=["A", "B", "C", "D"])
tabla_c
```

	A	B	C	D
0	1.764052	0.400157	0.978738	2.240893
1	1.867558	-0.977278	0.950088	-0.151357
2	-0.103219	0.410599	0.144044	1.454274

1. Operaciones básicas

1.a Construcción de tablas

```
# Creación de un dataframe a partir de una lista de tuplas
# y sin dar los nombres de las columnas:

tuplas = [
    (1, 2, 3),
    (4, 5, 6)
]
tabla_d = pd.DataFrame(tuplas)
tabla_d
```

```
# Ahora podemos renombramos las columnas:

tabla_d.columns = ["Una", "Dos", "Tres"]
tabla_d
```

```
# Los dos pasos anteriores a un tiempo:
# - Creación de la tabla (con una lista de listas esta vez),
# - y nombrabdo las columnas desde el principio:

listas = [
    [1, 2, 3],
    [4, 5, 6]
]

tabla_e = pd.DataFrame(listas, columns=["Una", "Dos", "Tres"])

tabla_e
```

	0	1	2
0	1	2	3
1	4	5	6

	Una	Dos	Tres
0	1	2	3
1	4	5	6

	Una	Dos	Tres
0	1	2	3
1	4	5	6

1.b Modificación de tablas

```
# Añadir (o modificar) una fila:

otra_fila = {"Una": 100, "Dos": 200, "Nueva": 300}
tabla_e.loc[2] = otra_fila
tabla_e
```

	Una	Dos	Tres
0	1	2	3.0
1	4	5	6.0
2	100	200	NaN

```
# Modificar el valor de una celda:

tabla_e.at[1, "Dos"] = 1000
tabla_e
```

	Una	Dos	Tres
0	1	2	3.0
1	4	1000	6.0
2	100	200	NaN

```
# Modificar los valores de una fila:

tabla_e.loc[1] = [-1, -2, -3]
tabla_e
```

	Una	Dos	Tres
0	1	2	3.0
1	-1	-2	-3.0
2	100	200	NaN

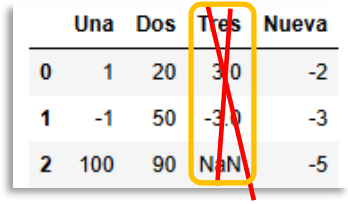
```
# Modificar los valores de una columna:

tabla_e["Dos"] = [20, 50, 90]
tabla_e
```

	Una	Dos	Tres
0	1	20	3.0
1	-1	50	-3.0
2	100	90	NaN

```
# Añadir una columna:

tabla_e["Nueva"] = [-2, -3, -5]
tabla_e
```

	Una	Dos	Tres	Nueva
0	1	20	3.0	-2
1	-1	50	-3.0	-3
2	100	90	NaN	-5

1.b Modificación de tablas



	Una	Dos	Tres	Nueva
0	1	20	30	-2
1	-1	50	-30	-3
2	100	90	NaN	-5

```
# Suprimir una columna:

del tabla_e["Tres"]
tabla_e
```

	Una	Dos	Nueva
0	1	20	-2
1	-1	50	-3
2	100	90	-5

```
# Otra manera de suprimir columnas...

tabla_e.drop(["Una", "Dos"], axis="columns")

# Pero OJO...
```

	Nueva
0	-2
1	-3
2	-5

```
# Cuidado, se ha calculado la eliminación de la columna,
# pero la operación no se ha realizado in place:

tabla_e
```

	Una	Dos	Nueva
0	1	20	-2
1	-1	50	-3
2	100	90	-5

```
tabla_e.drop(["Una", "Dos"], axis="columns", inplace=True)
tabla_e
```

	Nueva
0	-2
1	-3
2	-5

1.c. Operaciones in-place

```
# Una tabla para los ejemplos de este apartado:

tabla = pd.DataFrame({"A": [1, 3, 5, 7, 9, 11], "B": [2, 4, 6, 1, 2, 3]})
tabla
```

	A	B
0	1	2
1	3	4
2	5	6
3	7	1
4	9	2
5	11	3

```
# Ordenamos las filas de un dataframe

tabla.sort_values("B")
tabla

# OJO: no parece haber funcionado
```

	A	B
0	1	2
1	3	4
2	5	6
3	7	1
4	9	2
5	11	3

```
# La operación, en realidad, sí funciona...

tabla.sort_values("B")
```

	A	B
3	7	1
0	1	2
4	9	2
5	11	3
1	3	4
2	5	6

```
# Pero para que la tabla se actualice...

tabla.sort_values("B", inplace=True)
tabla
```

	A	B
3	7	1
0	1	2
4	9	2
5	11	3
1	3	4
2	5	6

```
# Y para que el índice también se reinicie:

tabla.reset_index(inplace=True)

tabla
```

	index	A	B
0	3	7	1
1	0	1	2
2	4	9	2
3	5	11	3
4	1	3	4
5	2	5	6

1.d. Modificación genérica de columnas

```
# Vamos a usar esta pequeña tabla como ejemplo:

viviendas = pd.DataFrame(
    {'Zona': ['B.Pilar', 'Chueca', 'Moncloa', 'B.Pilar', 'Chueca', 'Chueca', 'Moncloa'],
     'Dorms': [3, 2, 4, 4, 3, 1, 2],
     'Precio': [450, 600, 750, 550, 850, 300, 500]})

viviendas
```

	Zona	Dorms	Precio
0	B.Pilar	3	450
1	Chueca	2	600
2	Moncloa	4	750
3	B.Pilar	4	550
4	Chueca	3	850
5	Chueca	1	300
6	Moncloa	2	500

```
# Modificamos la columna "Precio", subiendo un 10%:

viviendas["Precio"] = viviendas["Precio"] * 1.10

viviendas
```

	Zona	Dorms	Precio
0	B.Pilar	3	495.0
1	Chueca	2	660.0
2	Moncloa	4	825.0
3	B.Pilar	4	605.0
4	Chueca	3	935.0
5	Chueca	1	330.0
6	Moncloa	2	550.0

```
# De otro modo, modificamos la columna "Precio", subiendo un 10%:

viviendas["Precio"] = viviendas["Precio"].apply(lambda valor: valor*1.10)

viviendas
```

	Zona	Dorms	Precio
0	B.Pilar	3	544.5
1	Chueca	2	726.0
2	Moncloa	4	907.5
3	B.Pilar	4	665.5
4	Chueca	3	1028.5
5	Chueca	1	363.0
6	Moncloa	2	605.0

```
# Observa que, sin la asignación, parece que la tabla se actualiza...

viviendas["Precio"].apply(lambda valor: 0)
```

```
0    0
1    0
2    0
3    0
4    0
5    0
6    0
Name: Precio, dtype: int64
```

```
# Pero no, la operación no es in place y por eso hace falta realizar la asignación:

viviendas
```

	Zona	Dorms	Precio
0	B.Pilar	3	544.5
1	Chueca	2	726.0
2	Moncloa	4	907.5
3	B.Pilar	4	665.5
4	Chueca	3	1028.5
5	Chueca	1	363.0
6	Moncloa	2	605.0

1.d. Modificación genérica de columnas

```
# Modificación condicional:

aumento_de_precio = lambda precio : precio + 100 if precio >= 500 else precio + 50

viviendas["Precio"] = viviendas["Precio"].apply(aumento_de_precio)

viviendas
```

	Zona	Dorms	Precio
0	B.Pilar	3	644.5
1	Chueca	2	826.0
2	Moncloa	4	1007.5
3	B.Pilar	4	765.5
4	Chueca	3	1128.5
5	Chueca	1	413.0
6	Moncloa	2	705.0

```
# Modificación condicional:
# Los inmuebles de Chueca bajan su precio en 100 euros exactamente:

viviendas["Precio"] = viviendas.loc[viviendas["Zona"] == "Chueca", ["Precio"]] = \
   viviendas["Precio"] - 100

viviendas
```

	Zona	Dorms	Precio
0	B.Pilar	3	544.5
1	Chueca	2	726.0
2	Moncloa	4	907.5
3	B.Pilar	4	665.5
4	Chueca	3	1028.5
5	Chueca	1	313.0
6	Moncloa	2	605.0

```
# Cálculo de una columna nueva con una fórmula:

viviendas["P.p.d."] = viviendas["Precio"] / viviendas["Dorms"]

viviendas
```

	Zona	Dorms	Precio	P.p.d.
0	B.Pilar	3	544.5	181.500000
1	Chueca	2	726.0	363.000000
2	Moncloa	4	907.5	226.875000
3	B.Pilar	4	665.5	166.375000
4	Chueca	3	1028.5	342.833333
5	Chueca	1	313.0	313.000000
6	Moncloa	2	605.0	302.500000

```
# Modificación de una columna:

viviendas["P.p.d."] = round(viviendas["P.p.d."], 2)

viviendas
```

	Zona	Dorms	Precio	P.p.d.
0	B.Pilar	3	544.5	181.50
1	Chueca	2	726.0	363.00
2	Moncloa	4	907.5	226.88
3	B.Pilar	4	665.5	166.38
4	Chueca	3	1028.5	342.83
5	Chueca	1	313.0	313.00
6	Moncloa	2	605.0	302.50

1.e. Agrupamiento de datos

```
viviendas = pd.DataFrame(
    {'Zona': ['B.Pilar', 'Chueca', 'Moncloa', 'B.Pilar', 'Chueca', 'Chueca', 'Moncloa'],
     'Dorms': [3, 2, 4, 4, 3, 1, 2],
     'Precio': [450, 600, 750, 550, 850, 300, 500]})

viviendas
```

	Zona	Dorms	Precio
0	B.Pilar	3	450
1	Chueca	2	600
2	Moncloa	4	750
3	B.Pilar	4	550
4	Chueca	3	850
5	Chueca	1	300
6	Moncloa	2	500

```
viviendas.groupby("Zona").mean()
```

Zona	Dorms	Precio
B.Pilar	3.5	500.000000
Chueca	2.0	583.333333
Moncloa	3.0	625.000000

```
viviendas.groupby("Zona").count()
```

Zona	Dorms	Precio
B.Pilar	2	2
Chueca	3	3
Moncloa	2	2

```
# Una manera de evitar la duplicidad de columnas anterior:

viviendas.groupby("Zona").size()
```

```
Zona
B.Pilar    2
Chueca     3
Moncloa    2
dtype: int64
```

```
# Otra manera de evitar la duplicidad de columnas:

props = viviendas[["Zona", "Dorms"]]
props.groupby("Zona").count()
```

Zona	Dorms
B.Pilar	2
Chueca	3
Moncloa	2

```
viviendas.groupby(["Zona", "Dorms"]).mean()
```

		Precio
Zona	Dorms	
B.Pilar	3	450.0
	4	550.0
Chueca	1	300.0
	2	600.0
	3	850.0
Moncloa	2	500.0
	4	750.0

```
viviendas.groupby(["Zona", "Dorms"]).agg(["mean", "count"])
```

		Precio	
		mean	count
Zona	Dorms		
B.Pilar	3	450.0	1
	4	550.0	1
Chueca	1	300.0	1
	2	600.0	1
	3	850.0	1
Moncloa	2	500.0	1
	4	750.0	1

1.f. Estadísticos básicos

```
viviendas = pd.DataFrame(
    {'Zona': ['B.Pilar', 'Chueca', 'Moncloa', 'B.Pilar', 'Chueca', 'Chueca', 'Moncloa'],
     'Dorms': [3, 2, 4, 4, 3, 1, 2],
     'Precio': [450, 600, 750, 550, 850, 300, 500]})

viviendas
```

	Zona	Dorms	Precio
0	B.Pilar	3	450
1	Chueca	2	600
2	Moncloa	4	750
3	B.Pilar	4	550
4	Chueca	3	850
5	Chueca	1	300
6	Moncloa	2	500

```
# Estadísticos básicos con datos cuantitativos:

viviendas.describe()

# Obsérvese que la columna "Zona" no se está incluyendo
```

	Dorms	Precio
count	7.000000	7.000000
mean	2.714286	571.428571
std	1.112697	184.519969
min	1.000000	300.000000
25%	2.000000	475.000000
50%	3.000000	550.000000
75%	3.500000	675.000000
max	4.000000	850.000000

```
# Si deseamos limitarnos a una columna:

viviendas["Dorms"].describe()
```

```
count    7.000000
mean     2.714286
std      1.112697
min      1.000000
25%      2.000000
50%      3.000000
75%      3.500000
max      4.000000
Name: Dorms, dtype: float64
```

```
# Podríamos haber obtenido
# estos estadísticos por separado:

print(viviendas["Dorms"].count())
print(viviendas["Dorms"].mean())
print(viviendas["Dorms"].min())
print(viviendas["Dorms"].quantile(q=0.25))
print(viviendas["Dorms"].quantile(q=0.50))
print(viviendas["Dorms"].quantile(q=0.75))
print(viviendas["Dorms"].max())
```

```
7
2.7142857142857144
1
2.0
3.0
3.5
4
```

```
# Con datos cualitativos o categóricos:

viviendas["Zona"].describe()

# Obtenemos: cuantas filas tenemos en total,
# cuantas zonas distintas,
# la más frecuente (la moda)
# y su frecuencia
```

```
count          7
unique         3
top       Chueca
freq           3
Name: Zona, dtype: object
```

```
# Estadística completa, datos cuantotativos y cualitativos:

viviendas.describe(include="all")

# Obvérvense los valores NaN
```

	Zona	Dorms	Precio
count	7	7.000000	7.000000
unique	3	NaN	NaN
top	Chueca	NaN	NaN
freq	3	NaN	NaN
mean	NaN	2.714286	571.428571
std	NaN	1.112697	184.519969
min	NaN	1.000000	300.000000
25%	NaN	2.000000	475.000000
50%	NaN	3.000000	550.000000
75%	NaN	3.500000	675.000000
max	NaN	4.000000	850.000000

1.g. Combinación de tablas, vertical

```python
# Combinación vertical de tablas:

tabla1 = pd.DataFrame({"A": [1, 2, 3], "B": [4, 5, 6]})
print(tabla1, "\n")

tabla2 = pd.DataFrame({"A": [7, 8, 9], "B": [10, 11, 12]})
print(tabla2, "\n")

tabla3 = pd.concat([tabla1, tabla2], ignore_index=True)
print(tabla3)
```

```
   A  B
0  1  4
1  2  5
2  3  6

   A   B
0  7  10
1  8  11
2  9  12

   A   B
0  1   4
1  2   5
2  3   6
3  7  10
4  8  11
5  9  12
```

```python
# Otra situación:

tabla1 = pd.DataFrame({"A": [1, 2, 3], "B": [4, 5, 6]})
print(tabla1, "\n")

tabla2 = pd.DataFrame({"B": [7, 8, 9], "C": [10, 11, 12]})
print(tabla2, "\n")

tabla3 = pd.concat([tabla1, tabla2], ignore_index=True)
print(tabla3)


# Observa que, al no coincidir el nombre de *todas* las columnas,
# la operación completa los valores desconocidos,
# sin tener en cuenta si hay valores coincidentes.
```

```
   A  B
0  1  4
1  2  5
2  3  6

   B   C
0  7  10
1  8  11
2  9  12

     A  B     C
0  1.0  4   NaN
1  2.0  5   NaN
2  3.0  6   NaN
3  NaN  7  10.0
4  NaN  8  11.0
5  NaN  9  12.0
```

1.g. Combinación de tablas, join interno

```python
# Merge:

tabla1 = pd.DataFrame({"A": [1, 2, 3], "B": [4, 5, 6]})
print(tabla1, "\n")

tabla2 = pd.DataFrame({"B": [4, 5, 6], "C": [7, 8, 9]})
print(tabla2, "\n")

tabla3 = pd.merge(tabla1, tabla2, on="B")

print(tabla3)
```

```
   A  B
0  1  4
1  2  5
2  3  6

   B  C
0  4  7
1  5  8
2  6  9

   A  B  C
0  1  4  7
1  2  5  8
2  3  6  9
```

```python
# Es el join interno de SQL.
# Observa el funcionamiento cuando algún valor no coincide:

tabla1 = pd.DataFrame({"A": [1, 2, 3], "B": [4, 5, 6]})
print(tabla1, "\n")

tabla2 = pd.DataFrame({"B": [4, 5, 60], "C": [7, 8, 9]})
print(tabla2, "\n")

tabla3 = pd.merge(tabla1, tabla2, on="B")

print(tabla3)
```

```
   A  B
0  1  4
1  2  5
2  3  6

    B  C
0   4  7
1   5  8
2  60  9

   A  B  C
0  1  4  7
1  2  5  8
```

1.g. Combinación de tablas, join externos

```
tabla1 = pd.DataFrame({"A": [1, 2, 3], "B": [4, 5, 6]})
print(tabla1, "\n")

tabla2 = pd.DataFrame({"B": [4, 5, 60], "C": [7, 8, 9]})
print(tabla2, "\n")
```

```
   A  B
0  1  4
1  2  5
2  3  6

    B  C
0   4  7
1   5  8
2  60  9
```

```
tabla4 = tabla1.merge(tabla2, how="outer")
print(tabla4)

tabla5 = tabla1.merge(tabla2, how="left")
print(tabla5)

tabla6 = tabla1.merge(tabla2, how="right")
## Aplicación: *realstate*
print(tabla6)
```

```
     A   B    C
0  1.0   4  7.0
1  2.0   5  8.0
2  3.0   6  NaN
3  NaN  60  9.0
   A  B    C
0  1  4  7.0
1  2  5  8.0
2  3  6  NaN
     A   B  C
0  1.0   4  7
1  2.0   5  8
2  NaN  60  9
```

2. Ejemplo. Realestate

```
import pandas as pd
realestate = pandas.read_csv('realestate.csv')
realestate
```

	street	city	zip	state	beds	baths	sq__ft	type	sale_date	price	latitude	longitude
0	3526 HIGH ST	SACRAMENTO	95838	CA	2	1	836	Residential	Wed May 21 00:00:00 EDT 2008	59222	38.631913	-121.434879
1	51 OMAHA CT	SACRAMENTO	95823	CA	3	1	1167	Residential	Wed May 21 00:00:00 EDT 2008	68212	38.478902	-121.431028
2	2796 BRANCH ST	SACRAMENTO	95815	CA	2	1	796	Residential	Wed May 21 00:00:00 EDT 2008	68880	38.618305	-121.443839
3	2805 JANETTE WAY	SACRAMENTO	95815	CA	2	1	852	Residential	Wed May 21 00:00:00 EDT 2008	69307	38.616835	-121.439146
4	6001 MCMAHON DR	SACRAMENTO	95824	CA	2	1	797	Residential	Wed May 21 00:00:00 EDT 2008	81900	38.519470	-121.435768
5	5828 PEPPERMILL CT	SACRAMENTO	95841	CA	3	1	1122	Condo	Wed May 21 00:00:00 EDT 2008	89921	38.662595	-121.327813
6	6048 OGDEN NASH WAY	SACRAMENTO	95842	CA	3	2	1104	Residential	Wed May 21 00:00:00 EDT 2008	90895	38.681659	-121.351705
7	2561 19TH AVE	SACRAMENTO	95820	CA	3	1	1177	Residential	Wed May 21 00:00:00	91002	38.535092	-121.481367

2. Realestate

```
import pandas as pd
realestate = pandas.read_csv('realestate.cs
realestate
```

	street	city	zip	state	beds	baths	sc	gitude
0	3526 HIGH ST	SACRAMENTO	95838	CA	2	1		34879
1	51 OMAHA CT	SACRAMENTO	95823	CA	3	1		31028
2	2796 BRANCH ST	SACRAMENTO	95815	CA	2	1		43839
3	2805 JANETTE WAY	SACRAMENTO	95815	CA	2	1		39146
4	6001 MCMAHON DR	SACRAMENTO	95824	CA	2	1		35768
5	5828 PEPPERMILL CT	SACRAMENTO	95841	CA	3	1		27813
6	6048 OGDEN NASH WAY	SACRAMENTO	95842	CA	3	2		51705
7	2561 19TH AVE	SACRAMENTO	95820	CA	3	1	1177 Residential	Wed May 21 00:00:00	91002	38.535092	-121.481367

```
type(realestate)
```

```
pandas.core.frame.DataFrame
```

```
realestate.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 985 entries, 0 to 984
Data columns (total 12 columns):
street        985 non-null object
city          985 non-null object
zip           985 non-null int64
state         985 non-null object
beds          985 non-null int64
baths         985 non-null int64
sq__ft        985 non-null int64
type          985 non-null object
sale_date     985 non-null object
price         985 non-null int64
latitude      985 non-null float64
longitude     985 non-null float64
dtypes: float64(2), int64(5), object(5)
memory usage: 92.4+ KB
```

2. Realestate

```
import pandas as pd
realestate = pandas.read_csv('realest
realestate
```

```
realestate['price']
```

```
0       59222
1       68212
2       68880
3       69307
4       81900
5       89921
6       90895
7       91002
     ...
980    232425
981    234000
982    235000
983    235301
984    235738
Name: price, dtype: int64
```

	street	city	zip	state	bed					te	price	latitude	longitude
0	3526 HIGH ST	SACRAMENTO	95838	CA						00 08	59222	38.631913	-121.434879
1	51 OMAHA CT	SACRAMENTO	95823	CA						00 08	68212	38.478902	-121.431028
2	2796 BRANCH ST	SACRAMENTO	95815	CA						00 08	68880	38.618305	-121.443839
3	2805 JANETTE WAY	SACRAMENTO	95815	CA						00 08	69307	38.616835	-121.439146
4	6001 MCMAHON DR	SACRAMENTO	95824	CA						00 EDT 2008	81900	38.519470	-121.435768
5	5828 PEPPERMILL CT	SACRAMENTO	95841	CA	3	1	1122	Condo	Wed May 21 00:00:00 EDT 2008		89921	38.662595	-121.327813
6	6048 OGDEN NASH WAY	SACRAMENTO	95842	CA	3	2	1104	Residential	Wed May 21 00:00:00 EDT 2008		90895	38.681659	-121.351705
7	2561 19TH AVE	SACRAMENTO	95820	CA	3	1	1177	Residential	Wed May 21 00:00:00		91002	38.535092	-121.481367

2. Realestate

```
import pandas as pd
realestate = pandas.read_csv('re
realestate
```

	street	city	zip
0	3526 HIGH ST	SACRAMENTO	95838
1	51 OMAHA CT	SACRAMENTO	95823
2	2796 BRANCH ST	SACRAMENTO	95815
3	2805 JANETTE WAY	SACRAMENTO	95815
4	6001 MCMAHON DR	SACRAMENTO	95824
5	5828 PEPPERMILL CT	SACRAMENTO	95841
6	6048 OGDEN NASH WAY	SACRAMENTO	95842
7	2561 19TH AVE	SACRAMENTO	95820

```
realestate['price']
```

```
realestate[['price','zip']]
```

	price	zip
0	59222	95838
1	68212	95823
2	68880	95815
3	69307	95815
...	...	...
981	234000	95823
982	235000	95610
983	235301	95758
984	235738	95762

985 rows × 2 columns

te	price	latitude	longitude
)0)8	59222	38.631913	-121.434879
)0)8	68212	38.478902	-121.431028
)0)8	68880	38.618305	-121.443839
)0)8	69307	38.616835	-121.439146
64)0 T 2008	81900	38.519470	-121.435768
:00:00 T 2008	89921	38.662595	-121.327813
:00:00 T 2008	90895	38.681659	-121.351705
:00:00	91002	38.535092	-121.481367

	street	city	zip	state	beds	baths	sq__ft	type	sale_date	price	latitude	longitude
0	3526 HIGH ST	SACRAMENTO	95838	CA	2	1	836	Residential	Wed May 21 00:00:00 EDT 2008	59222	38.631913	-121.434879
1	51 OMAHA CT	SACRAME								68212	38.478902	-121.431028
2	2796 BRANCH ST	SACRAME								68880	38.618305	-121.443839
3	2805 JANETTE WAY	SACRAME								69307	38.616835	-121.439146
4	6001 MCMAHON DR	SACRAME								81900	38.519470	-121.435768
5	5828 PEPPERMILL CT	SACRAME								89921	38.662595	-121.327813
6	6048 OGDEN NASH WAY	SACRAME								90895	38.681659	-121.351705
7	2561 19TH AVE	SACRAM								91002	38.535092	-121.481367

```
selcols = realestate.columns[2:6]
selcols
```

```
Index(['zip', 'state', 'beds', 'baths'], dtype='object')
```

```
realestate[selcols]
```

	zip	state	beds	baths
0	95838	CA	2	1
1	95823	CA	3	1
2	95815	CA	2	1
3	95815	CA	2	1
...	...	...	...	...
981	95823	CA	3	2
982	95610	CA	3	2
983	95758	CA	4	2
984	95762	CA	3	2

985 rows × 4 columns

	street	city	zip	state	beds	baths	sq__ft	type	sale_date	price	latitude	longitude
0	3526 HIGH ST	SACRAMENTO	95838	CA	2	1	836	Residential	Wed May 21 00:00:00 EDT 2008	59222	38.631913	-121.434879
1	51 OMAHA CT	SACRAME								68212	38.478902	-121.431028
2	2796 BRANCH ST	SACRAME								68880	38.618305	-121.443839
3	2805 JANETTE WAY	SACRAME										-121.439146
4	6001 MCMAHON DR	SACRAME										-121.435768
5	5828 PEPPERMILL CT	SACRAME										-121.327813
6	6048 OGDEN NASH WAY	SACRAME										-121.351705
7	2561 19TH AVE	SACRAM										-121.481367

```
selcols = realestate.columns[2:6]
selcols
```

```
Index(['zip', 'state', 'beds', 'baths'], dtype='object')
```

```
realestate[selco
```

	zip	state
0	95838	CA
1	95823	CA
2	95815	CA
3	95815	CA
...	...	...
981	95823	CA
982	95610	CA
983	95758	CA
984	95762	CA

985 rows × 4 columns

```
mixed = list(selcols)+['price']
mixed
```

```
['zip', 'state', 'beds', 'baths', 'price']
```

```
realestate[mixed]
```

	zip	state	beds	baths	price
0	95838	CA	2	1	59222
1	95823	CA	3	1	68212
2	95815	CA	2	1	68880
3	95815	CA	2	1	69307
4	95824	CA	2	1	81900

	street	city	zip	state	beds	baths	sq__ft	type	sale_date	price	latitude	longitude
0	3526 HIGH ST	SACRAMENTO	95838	CA	2	1	836	Residential	Wed May 21 00:00:00 EDT 2008	59222	38.631913	-121.434879
1	51 OMAHA CT	SACRAMENTO	95823	CA	3	1	1167	Residential	Wed May 21 00:00:00 EDT 2008	68212	38.478902	-121.431028
2	2796 BRANCH ST	SACRAMENTO	95815	CA	2	1	796	Residential	Wed May 21 00:00:00 EDT 2008	68880	38.618305	-121.443839
3	2805 JANETTE WAY	SACRAMENTO	95815	CA	2	1	852	Residential	Wed May 21 00:00:00 EDT 2008	69307	38.616835	-121.439146
4	6001 MCMAHON DR	SACRAMENTO	95824	CA	2						70	-121.435768
5	5828 PEPPERMILL CT	SACRAMENTO	95841	CA	3						95	-121.327813
6	6048 OGDEN NASH WAY	SACRAMENTO	95842	CA	3						59	-121.351705
7	2561 19TH AVE	SACRAMENTO	95820	CA	3						02	-121.481367

```
realestate.iloc[3]
```

```
street                    2805 JANETTE WAY
city                                SACRAMENTO
zip                                      95815
state                                       CA
beds                                         2
baths                                        1
sq__ft                                     852
type                               Residential
sale_date    Wed May 21 00:00:00 EDT 2008
price                                    69307
latitude                               38.6168
longitude                             -121.439
Name: 3, dtype: object
```

	street	city	zip	state	beds	baths	sq__ft	type	sale_date	price	latitude	longitude
0	3526 HIGH ST	SACRAMENTO	95838	CA	2	1	836	Residential	Wed May 21 00:00:00 EDT 2008	59222	38.631913	-121.434879
1	51 OMAHA CT	SACRAMENTO	95823	CA	3	1	1167	Residential	Wed May 21 00:00:00 EDT 2008	68212	38.478902	-121.431028
2	2796 BRANCH ST	SACRAMENTO	95815	CA	2	1	796	Residential	Wed May 21 00:00:00 EDT 2008	68880	38.618305	-121.443839
3	2805 JANETTE WAY	SACRAMENTO	95815	CA	2	1	852	Residential	Wed May 21 00:00:00 EDT 2008	69307	38.616835	-121.439146
4	6001 MCMAHON DR	SACRAMENTO	95824	CA	2							-121.435768
5	5828 PEPPERMILL CT	SACRAMENTO	95841	CA	3							-121.327813
6	6048 OGDEN NASH WAY	SACRAMENTO	95842	CA	3							-121.351705
7	2561 19TH AVE	SACRAMENTO	95820	CA	3							-121.481367

```
realestate.iloc[3]
```

```
street                2805 JANETTE WAY
city                        SACRAMENTO
zip                              95815
state                               CA
beds                                 2
baths                                1
longitude                      -121.439
Name: 3, dtype: object
```

```
realestate.iloc[[3,6]]
```

	street	city	zip	state	beds	baths	sq__ft	type	sale_date	price	latitude	longitude
3	2805 JANETTE WAY	SACRAMENTO	95815	CA	2	1	852	Residential	Wed May 21 00:00:00 EDT 2008	69307	38.616835	-121.439146
6	6048 OGDEN NASH WAY	SACRAMENTO	95842	CA	3	2	1104	Residential	Wed May 21 00:00:00 EDT 2008	90895	38.681659	-121.351705

	street	city	zip	state	beds	baths	sq__ft	type	sale_date	price	latitude	longitude
0	3526 HIGH ST	SACRAMENTO	95838	CA	2	1	836	Residential	Wed May 21 00:00:00	59222	38.631913	-121.434879
1	51 OMAHA CT	SACRAMENTO	95823	CA						68212	38.478902	-121.431028
2	2796 BRANCH ST	SACRAMENTO	95815	CA						68880	38.618305	-121.443839
3	2805 JANETTE WAY	SACRAMENTO	95815	CA						69307	38.616835	-121.439146
4	6001 MCMAHON DR	SACRAMENTO	95824	CA						81900	38.519470	-121.435768
5	5828 PEPPERMILL CT	SACRAMENTO	95841	CA						89921	38.662595	-121.327813
6	6048 OGDEN NASH WAY	SACRAMENTO	95842	CA						90895	38.681659	-121.351705
7	2561 19TH AVE	SACRAMENTO	95820	CA						91002	38.535092	-121.481367

```
realestate.iloc[3]['city']
```

```
'SACRAMENTO'
```

```
realestate.iloc[3][1]
```

```
'SACRAMENTO'
```

```
realestate[['price','zip']][3:8]
```

	price	zip
3	69307	95815
4	81900	95824
5	89921	95841
6	90895	95842
7	91002	95820

2. Realestate

```
realestate.sort_values(by='price', ascending=False)
```

	street	city	zip	state	beds	baths	sq__ft	type	sale_date	price	latitude	longitude
864	9401 BARREL RACER CT	WILTON	95693	CA	4	3	4400	Residential	Fri May 16 00:00:00 EDT 2008	884790	38.415298	-121.194858
863	2982 ABERDEEN LN	EL DORADO HILLS	95762	CA	4	3	0	Residential	Fri May 16 00:00:00 EDT 2008	879000	38.706692	-121.058869
...	...	...	...	...	...	...	...	...	...	...	...	...

```
realestate.describe()
```

	zip	beds	baths	sq__ft	price	latitude	longitude
count	985.000000	985.000000	985.000000	985.000000	985.000000	985.000000	985.000000
mean	95750.697462	2.911675	1.776650	1314.916751	234144.263959	38.607732	-121.355982
std	85.176072	1.307932	0.895371	853.048243	138365.839085	0.145433	0.138278
min	95603.000000	0.000000	0.000000	0.000000	1551.000000	38.241514	-121.551704
25%	95660.000000	2.000000	1.000000	952.000000	145000.000000	38.482717	-121.446127
50%	95762.000000	3.000000	2.000000	1304.000000	213750.000000	38.626582	-121.376220
75%	95828.000000	4.000000	2.000000	1718.000000	300000.000000	38.695589	-121.295778
max	95864.000000	8.000000	5.000000	5822.000000	884790.000000	39.020808	-120.597599

2. Realestate

```
cond = realestate['baths'] == 2
cond
```

```
0       False
1       False
2       False
3       False
4       False
5       False
6        True
7       False
8        True
```

```
realestate[cond]
```

	street	city	zip	state	beds	baths	sq__ft	type	sale_date	price	latitude	longitude
6	6048 OGDEN NASH WAY	SACRAMENTO	95842	CA	3	2	1104	Residential	Wed May 21 00:00:00 EDT 2008	90895	38.681659	-121.351705
8	11150 TRINITY RIVER DR Unit 114	RANCHO CORDOVA	95670	CA	2	2	941	Condo	Wed May 21 00:00:00 EDT 2008	94905	38.621188	-121.270555
9	7325 10TH ST	RIO LINDA	95673	CA	3	2	1146	Residential	Wed May 21 00:00:00 EDT 2008	98937	38.700909	-121.442979
10	645 MORRISON AVE	SACRAMENTO	95838	CA	3	2	909	Residential	Wed May 21 00:00:00 EDT 2008	100309	38.637663	-121.451520
11	4085 FAWN CIR	SACRAMENTO	95823	CA	3	2	1289	Residential	Wed May 21 00:00:00 EDT 2008	106250	38.470746	-121.458918

2. Realestate

```
two_baths = realestate['baths'] == 2
three_beds = realestate['beds'] == 3
realestate[two_baths & three_beds]
```

	street	city	zip	state	beds	baths	sq__ft	type	sale_date	price	latitude	longitude
6	6048 OGDEN NASH WAY	SACRAMENTO	95842	CA	3	2	1104	Residential	Wed May 21 00:00:00 EDT 2008	90895	38.681659	-121.351705
9	7325 10TH ST	RIO LINDA	95673	CA	3	2	1146	Residential	Wed May 21 00:00:00 EDT 2008	98937	38.700909	-121.442979
10	645 MORRISON AVE	SACRAMENTO	95838	CA	3	2	909	Residential	Wed May 21 00:00:00 EDT 2008	100309	38.637663	-121.451520
11	4085 FAWN CIR	SACRAMENTO	95823	CA	3	2	1289	Residential	Wed May 21 00:00:00 EDT 2008	106250	38.470746	-121.458918
19	113 LEEWILL AVE	RIO LINDA	95673	CA	3	2	1356	Residential	Wed May 21 00:00:00 EDT 2008	121630	38.689999	-121.463220

2. Realestate

```
two_baths = realestate['baths'] == 2
three_beds = realestate['beds'] == 3
realestate[two_baths & three_beds]
```

	street	city	zip	state	beds	baths	sq_	de
6	6048 OGDEN NASH WAY	SACRAMENTO	95842	CA	3	2	11	05
9	7325 10TH ST	RIO LINDA	95673	CA	3	2	11	79
10	645 MORRISON AVE	SACRAMENTO	95838	CA	3	2	9	20
11	4085 FAWN CIR	SACRAMENTO	95823	CA	3	2	12	18
19	113 LEEWILL AVE	RIO LINDA	95673	CA	3	2	13	20

```
realestate.groupby(['zip','baths']).size()
```

```
zip    baths
95603  2          2
       3          3
95608  1          4
       2         15
       3          1
95610  2          5
       3          1
       4          1
95614  2          1
95619  2          1
95621  1          7
       2         20
       3          1
```

2. Realestate

```
two_baths = realestate['baths'] == 2
three_beds = realestate['beds'] == 3
realestate[two_baths & three_beds]
```

```
realestate.groupby(['zip','baths']).size().unstack()
```

baths	0	1	2	3	4	5
zip						
95603	NaN	NaN	2.0	3.0	NaN	NaN
95608	NaN	4.0	15.0	1.0	NaN	NaN
95610	NaN	NaN	5.0	1.0	1.0	NaN
95614	NaN	NaN	1.0	NaN	NaN	NaN
95619	NaN	NaN	1.0	NaN	NaN	NaN
95621	NaN	7.0	20.0	1.0	NaN	NaN
95623	NaN	1.0	1.0	NaN	NaN	NaN
95624	3.0	1.0	22.0	8.0	NaN	NaN

```
realestate.groupby(['zip','baths']).size()
```

```
zip    baths
95603  2        2
       3        3
95608  1        4
       2       15
       3        1
95610  2        5
       3        1
       4        1
95614  2        1
95619  2        1
95621  1        7
       2       20
       3        1
```

2. Realestate

```
realestate.groupby(['zip','baths']).size().unsta
```

baths	0	1	2	3	4	5
zip						
95603	NaN	NaN	2.0	3.0	NaN	NaN
95608	NaN	4.0	15.0	1.0	NaN	NaN
95610	NaN	NaN	5.0	1.0	1.0	NaN
95614	NaN	NaN	1.0	NaN	NaN	NaN
95619	NaN	NaN	1.0	NaN	NaN	NaN
95621	NaN	7.0	20.0	1.0	NaN	NaN
95623	NaN	1.0	1.0	NaN	NaN	NaN
95624	3.0	1.0	22.0	8.0	NaN	NaN

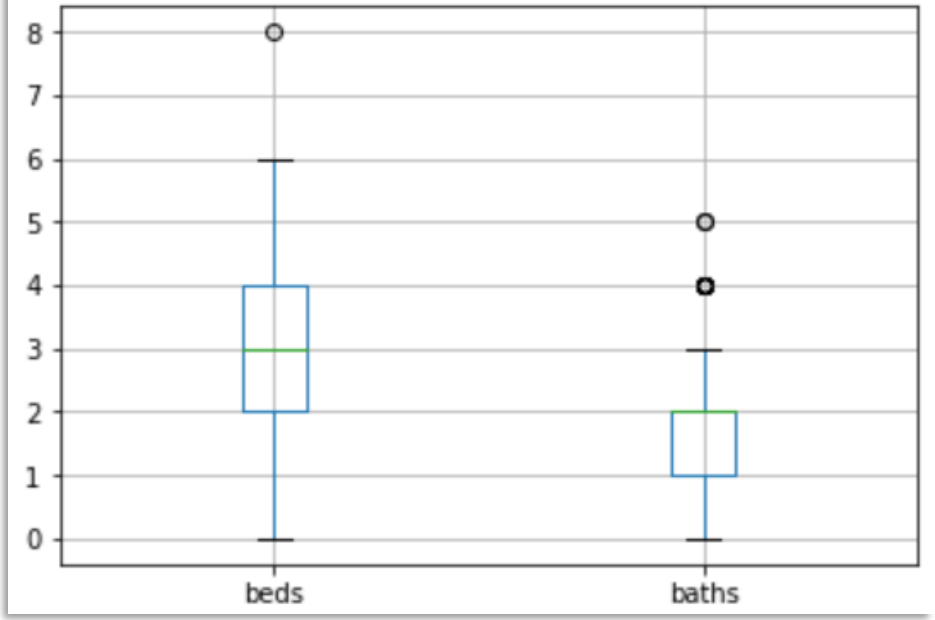
```
tabla = realestate.groupby(['zip','baths']).size().unstack()
tabla = tabla.fillna(0)
tabla[tabla.columns] = tabla[tabla.columns].astype(int)
tabla
```

baths	0	1	2	3	4	5
zip						
95603	0	0	2	3	0	0
95608	0	4	15	1	0	0
95610	0	0	5	1	1	0
95614	0	0	1	0	0	0
95619	0	0	1	0	0	0
...	...	...	...	...	...	...
95838	1	11	24	1	0	0
95841	0	4	3	0	0	0
95842	0	7	14	1	0	0
95843	0	1	24	8	0	0
95864	0	3	1	1	0	0

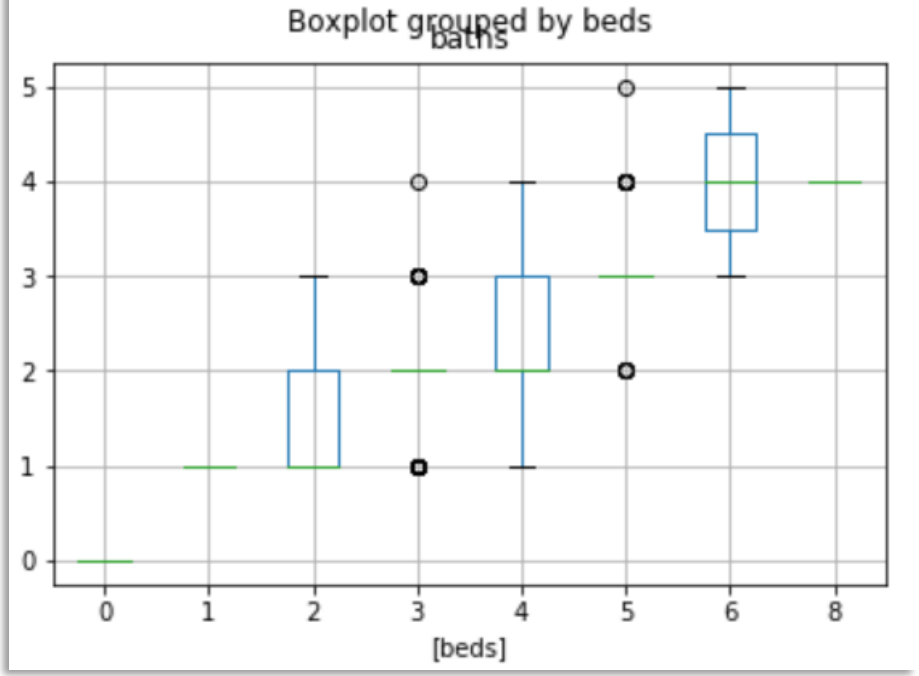
68 rows × 6 columns

Realestate. Visualización

```
realestate[['beds','baths']].boxplot()
```




```
realestate[['beds','baths']].boxplot(by="beds")
```

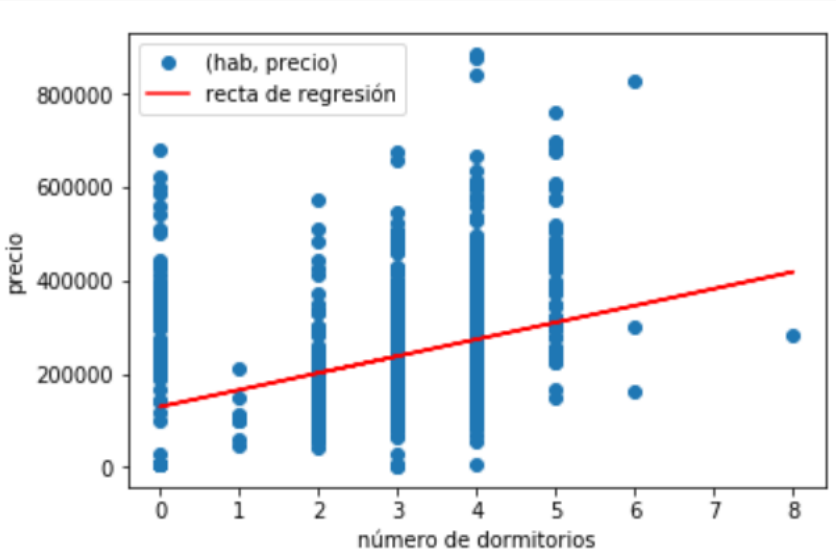


Realestate. Matplotlib

```
tabla_habs_precios = realestate[['beds','price']]
tabla_habs_precios
```

	beds	price
0	2	59222
1	3	68212
2	2	68880
3	2	69307
4	2	81900

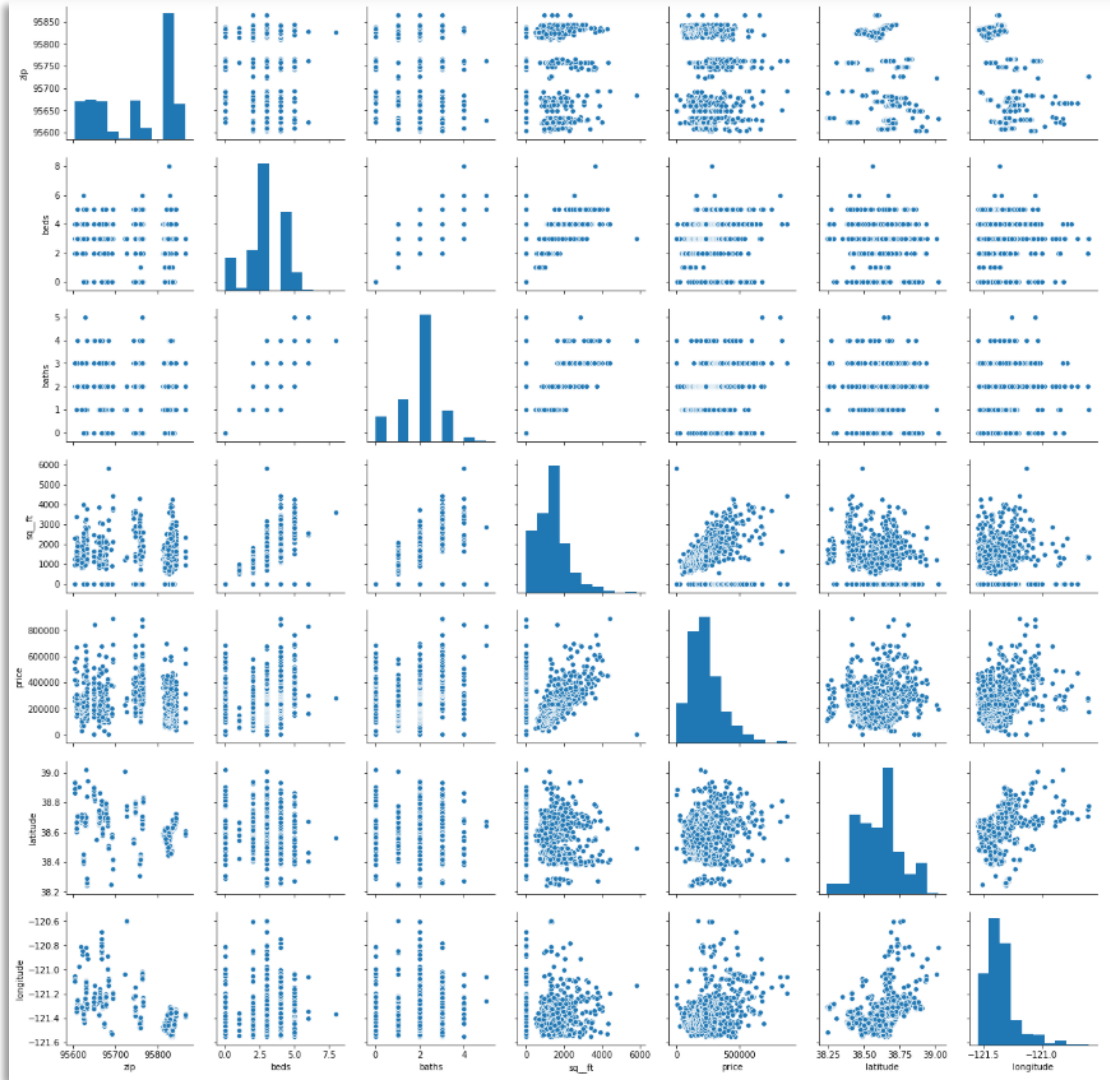
```
from scipy import stats
import numpy as np
import matplotlib.pyplot as plt

xs = tabla_habs_precios['beds']
ys = tabla_habs_precios['price']
slope, intercept, r_value, p_value, std_err = \
    stats.linregress(xs, ys)
recta_regres = lambda x: intercept + slope*x

plt.plot(xs, ys, 'o', label='(hab, precio)')
plt.plot(xs, recta_regres(xs), 'r', \
         label='recta de regresión')
plt.legend(loc = 'upper left')
plt.xlabel('número de dormitorios')
plt.ylabel('precio')
plt.show()
```



Realestate. seaborn

```
import seaborn

seaborn.pairplot(realestate)
```

https://seaborn.pydata.org/



Realestate. seaborn

Asuntos técnicos

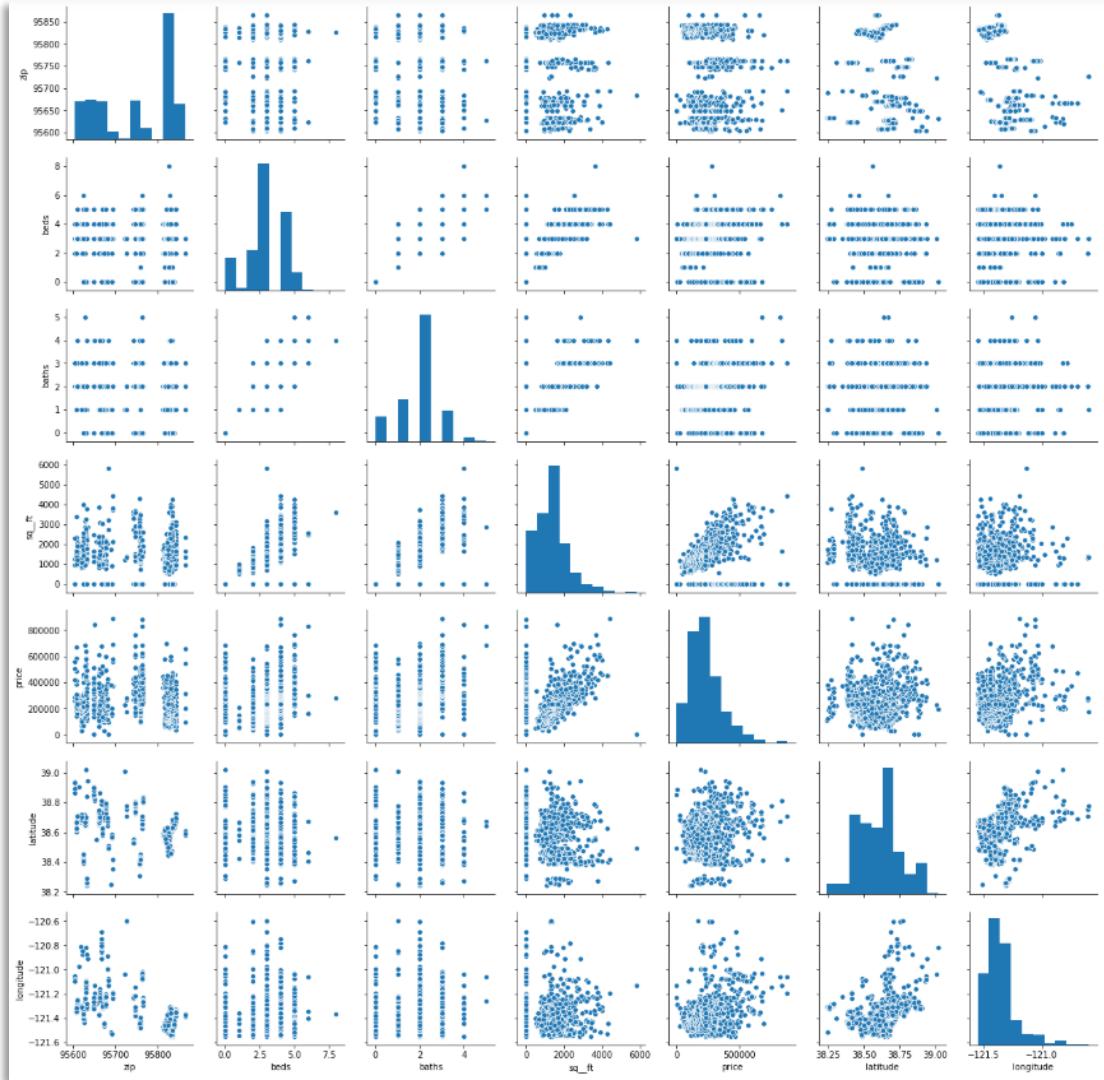
```
import seaborn
```

```
import warnings
warnings.filterwarnings('ignore')
```

```
%matplotlib inline
```

```
seaborn.pairplot(realestate)
```

https://seaborn.pydata.org/



3. Series

```
notas = [3.5, 5.0, 4.1, 4.5, 6.8, 8.0, 6.0, 8.9, 7.5, 9.0, 10.0, 8.5, 9.0, 8.5, 9.5]

serie_de_notas = pd.Series(notas)

print(serie_de_notas)
```

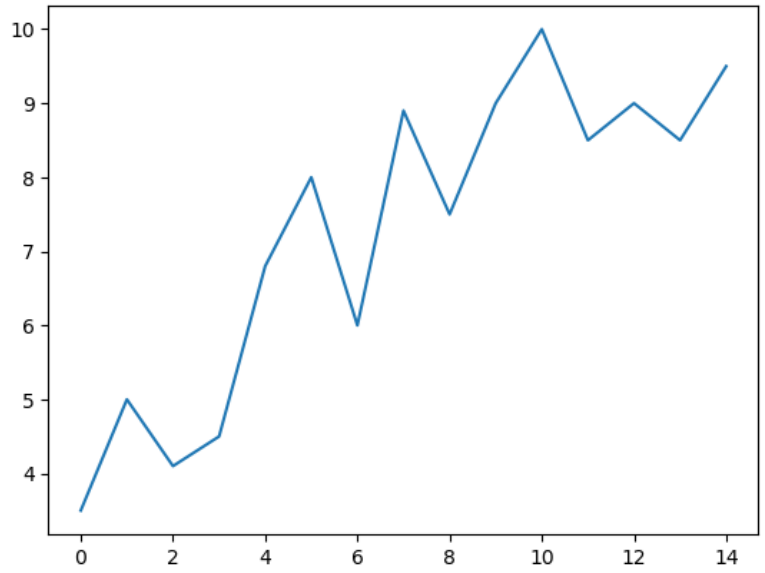
```
0      3.5
1      5.0
2      4.1
3      4.5
4      6.8
5      8.0
6      6.0
7      8.9
8      7.5
9      9.0
10    10.0
11     8.5
12     9.0
13     8.5
14     9.5
dtype: float64
```

```
print("La primera nota: ", serie_de_notas[0])
num_notas =  serie_de_notas.count()
print("Cuantas notas tengo: ", num_notas)
print("La última nota: ", serie_de_notas[num_notas-1])

serie_de_notas.describe()
```

```
La primera nota:  3.5
Cuantas notas tengo:  15
La última nota:  9.5

count    15.000000
mean      7.253333
std       2.131353
min       3.500000
25%       5.500000
50%       8.000000
75%       8.950000
max      10.000000
dtype: float64
```

```
serie_de_notas.plot()
```

```
<Axes: >
```



3. Series

```
serie_de_notas.to_csv("./data/notas.csv", sep = ';')
```

A	B
	0
0	3.5
1	5.0
2	4.1
3	4.5
4	6.8
5	8.0
6	6.0
7	8.9
8	7.5
9	9.0
10	10.0
11	8.5
12	9.0
13	8.5
14	9.5

```
# Lectura de la tabla de datos, seleccionando sólo la segunda columna:

notas_cargadas = pd.Series(pd.read_csv("./data/notas.csv", sep = ';', header=None, skiprows=1)[1])
notas_cargadas
```

```
0      3.5
1      5.0
2      4.1
3      4.5
4      6.8
5      8.0
6      6.0
7      8.9
8      7.5
9      9.0
10    10.0
11     8.5
12     9.0
13     8.5
14     9.5
Name: 1, dtype: float64
```

ntic master
revolucionamos la comunicación

UNIVERSIDAD COMPLUTENSE DE MADRID

Programación. Python

Dataframes con pandas

Cristóbal Pareja Flores