semana 05 pandas ejemplos

June 7, 2021

1 Pandas

La biblioteca pandas es un marco de trabajo para el procesamiento y análisis de datos en Python. Para más información sobre la biblioteca pandas, y su documentación oficial, consulte el sitio web del proyecto en http://pandas.pydata.org

2 Importar los modulos

[2]:
```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
from pandas.core.series import Series
```

3 Serie

[3]:
```
# Ejemplo de una serie, las poblaciones de un país
serie = pd.Series( [51,32,45,0.5,6.5,19.5,9.5,17,7,3.5,11.5,
                    28.5,126,212,0.4] ) # Millones de habitantes 0.4 = 400k
print ('Serie = ')
print (serie)

print ('dtype = ', serie.dtype)


lista = (serie.index)
print (lista)
# RangeIndex(start=0, stop=4, step=1)
```

```
Serie =
0      51.0
1      32.0
2      45.0
3       0.5
4       6.5
5      19.5
```

```
6       9.5
7      17.0
8       7.0
9       3.5
10     11.5
11     28.5
12    126.0
13    212.0
14      0.4
dtype: float64
dtype =  float64
RangeIndex(start=0, stop=15, step=1)
```

[3]:
```
# Si bien el uso de matrices o series de datos con índices enteros
# es una representación totalmente funcional de los datos, no es descriptiva.
# Por ejemplo, si los datos representan la población de cuatro paises de LATAM,
# es conveniente y descriptivo utilizar los nombres de las ciudades como␣
 ↪índices
# en lugar de que los números enteros. Con un objeto Serie esto es posible,
# y podemos asignar el atributo índice de un objeto Serie a una lista con␣
 ↪nuevos
# índices para conseguirlo. También podemos establecer el atributo name del␣
 ↪objeto Serie,
# para darle un nombre descriptivo

# Asignamos los nuevos indices
serie.index =['Colombia', 'Perú', 'Argentina', 'Surinam', 'Nicaragua', 'Chile',
              'Honduras', 'Guatemala', 'Paraguay', 'Uruguay', 'Bolivia',
              'Venezuela', 'México', 'Brasil', 'Belize']
# Asignamos el nuevo nombre de la serie
serie.name = 'Poblacion' # ahora la serie se llamará poblacion

print ('serie')
print (serie)

# Tambien podemos hacer esto manual de la siguiente forma
# serieNueva = pd.Series ( [51000000,32000000,45000000,19000000],
#                          name = 'Poblacion',
#                          index = ['Colombia', 'Peru', 'Argentina', 'Chile'])
# print ('serieNueva')
# print (serieNueva)
```

```
serie
Colombia      51.0
Perú          32.0
Argentina     45.0
Surinam        0.5
```

```
Nicaragua      6.5
Chile         19.5
Honduras       9.5
Guatemala     17.0
Paraguay       7.0
Uruguay        3.5
Bolivia       11.5
Venezuela     28.5
México       126.0
Brasil       212.0
Belize         0.4
Name: Poblacion, dtype: float64
```

[4]:
```
# Imprimir solo la población de Colombia
print ('Población de Colombia: ', serie['Colombia']) # Observe el indice

# Imprimir solo la población de Argentina
print ('Población de Argentina: ', serie.Argentina) # Observe el indice

# Imprimir la población de paises en especifico
print ('Poblacion de algunos paises en especifico: ')
print (serie [['Argentina', 'Colombia']] ) # Observe los indices
```

```
Población de Colombia:  51.0
Población de Argentina:  45.0
Poblacion de algunos paises en especifico:
Argentina    45.0
Colombia     51.0
Name: Poblacion, dtype: float64
```

[5]:
```
# Con una serie de datos representada como un objeto Serie, podemos
# calcular fácilmente sus estadísticas descriptivas utilizando los métodos de␣
 ↪la serie
# count (el número de puntos de datos),
# median (calcular la mediana),
# mean (calcular el valor medio),
# std (calcular la desviación estándar),
# min y max (valores mínimo y máximo), y
# el cuantil (para calcular los cuantiles):

#serie.median(), serie.mean(), serie.std()
print ('media ', serie.mean())
print ('mediana ', serie.median())
print ('desv estándar ', serie.std())
print ('maximo ', serie.max())
print ('minimo ', serie.min())
print ('')
```

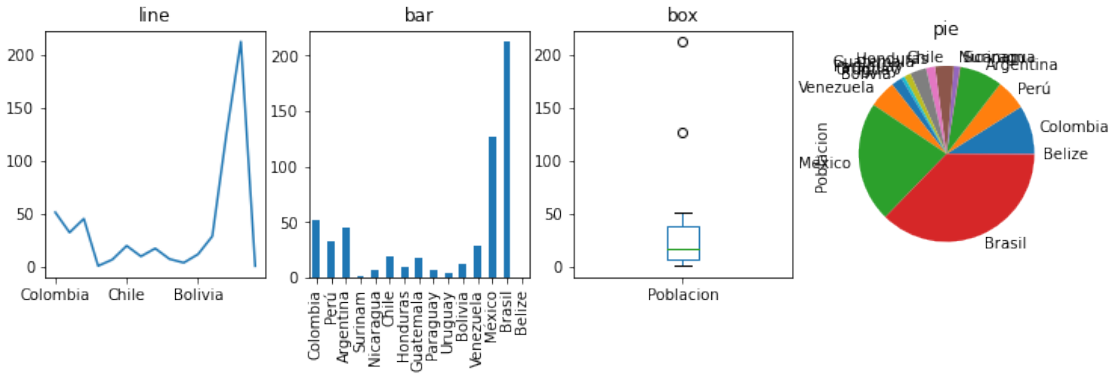
```
print ('serie = ')
print (serie)
print (serie.describe())
```

```
media  37.99333333333333
mediana  17.0
desv estándar  57.70956180573132
maximo  212.0
minimo  0.4

serie =
Colombia      51.0
Perú          32.0
Argentina     45.0
Surinam        0.5
Nicaragua      6.5
Chile         19.5
Honduras       9.5
Guatemala     17.0
Paraguay       7.0
Uruguay        3.5
Bolivia       11.5
Venezuela     28.5
México       126.0
Brasil       212.0
Belize         0.4
Name: Poblacion, dtype: float64
count     15.000000
mean      37.993333
std       57.709562
min        0.400000
25%        6.750000
50%       17.000000
75%       38.500000
max      212.000000
Name: Poblacion, dtype: float64
```

[6]:
```
# la visualización gráfica con matplotlib
figura, ejes = plt.subplots (1, 4, figsize = (12,3))
serie.plot (ax = ejes[0], kind = 'line', title = 'line')
serie.plot (ax = ejes[1], kind = 'bar', title = 'bar')
serie.plot (ax = ejes[2], kind = 'box', title = 'box')
serie.plot (ax = ejes[3], kind = 'pie', title = 'pie')
```

[6]:
```
<AxesSubplot:title={'center':'pie'}, ylabel='Poblacion'>
```



4 Data frame

Para arrays de mayor dimensión (principalmente arrays bidimensionales, o tablas), la estructura de datos estructura de datos correspondiente es el objeto Pandas DataFrame. Se puede ver como una colección de objetos Series con un índice común.

Existen numerosas formas de inicializar un DataFrame. Para ejemplos sencillos, la forma más fácil es pasar una lista anidada de Python o un diccionario al constructor del objeto DataFrame del objeto DataFrame. Por ejemplo, consideremos una extensión del conjunto de datos que utilizamos en la sección anterior sección anterior, en la que, además de la población de cada país, incluimos una columna que especifica del idioma. Podemos crear el correspondiente objeto DataFrame de la siguiente manera:

[4]:
```
# population = [51,32,45,0.5,6.5,19.5,9.5,17,7,3.5,11.5,28.5,126,212, 0.4]
# indices = ['Colombia', 'Perú', 'Argentina', 'Chile', 'Nicaragua', 'Chile',
#               'Honduras', 'Guatemala', 'Paraguay', 'Uruguay', 'Bolivia',
#               'Venezuela', 'México', 'Brasil', 'Belize']
matriz = [[51,'Español'],[32,'Español'],[45,'Español'],[0.5,'Neerlandés'],
          [6.5,'Español'],[19.5,'Español'],[9.5,'Español'],[17, 'Español'],
          [7,'Español'],[3.5,'Español'], [11.5,'Español'],[28.5,'Español'],
          [126,'Español'],[212,'Portugués'], [0.4,'Inglés']]
dataFrame = pd.DataFrame (matriz)
print ('dataFrame')
print (dataFrame)
```

```
dataFrame
        0           1
0    51.0     Español
1    32.0     Español
2    45.0     Español
3     0.5  Neerlandés
4     6.5     Español
5    19.5     Español
6     9.5     Español
```

```
7    17.0     Español
8     7.0     Español
9     3.5     Español
10   11.5     Español
11   28.5     Español
12  126.0     Español
13  212.0   Portugués
14    0.4      Inglés
```

[5]:
```
# Podemos utilizar la indexación por etiquetas para las filas
# asignando una secuencia de etiquetas al atributo y, además, podemos asignar
# al atributo de las columnas una secuencia de etiquetas para las columnas:

# Cambiamos los indices enteros por los nombres de los paises
dataFrame.index = ['Colombia', 'Perú', 'Argentina', 'Surinam', 'Nicaragua',␣
 ↪'Chile',
             'Honduras', 'Guatemala', 'Paraguay', 'Uruguay', 'Bolivia',
             'Venezuela', 'México', 'Brasil', 'Belize']
print ('dataFrame con los indices correspondientes a cada pais')
print (dataFrame)
```

```
dataFrame con los indices correspondientes a cada pais
               0           1
Colombia    51.0     Español
Perú        32.0     Español
Argentina   45.0     Español
Surinam      0.5  Neerlandés
Nicaragua    6.5     Español
Chile       19.5     Español
Honduras     9.5     Español
Guatemala   17.0     Español
Paraguay     7.0     Español
Uruguay      3.5     Español
Bolivia     11.5     Español
Venezuela   28.5     Español
México     126.0     Español
Brasil     212.0   Portugués
Belize       0.4      Inglés
```

[6]:
```
# Ahora asignamos atributos a las columnas
dataFrame.columns = ['Población', 'Idioma']
print ('daFrame completo: ')
print (dataFrame)
```

```
daFrame completo:
           Población      Idioma
Colombia        51.0     Español
Perú            32.0     Español
```

```
Argentina       45.0     Español
Surinam          0.5  Neerlandés
Nicaragua        6.5     Español
Chile           19.5     Español
Honduras         9.5     Español
Guatemala       17.0     Español
Paraguay         7.0     Español
Uruguay          3.5     Español
Bolivia         11.5     Español
Venezuela       28.5     Español
México         126.0     Español
Brasil         212.0   Portugués
Belize           0.4      Inglés
```

5 dataFrame a partir de un diccionario

Creamos un dataFrame a partir de un diccionario de la siguiente forma:

[7]:
```
dataFrame = pd.DataFrame ( {'Población': [51,32,45,0.5,6.5,19.5,9.5,17,7,3.5,11.
↪5,
                            28.5,126,212, 0.4],
                           'Idioma':['Español',␣
↪'Español','Español','Neerlandés',
                                     'Español','Español','Español','Español',
                                     'Español','Español','Español','Español',
                                     'Español','Portugués','Inglés']
                           },
                           index = ['Colombia', 'Perú', 'Argentina', 'Surinam',␣
↪'Nicaragua', 'Chile',
                                    'Honduras', 'Guatemala', 'Paraguay',␣
↪'Uruguay', 'Bolivia',
                                    'Venezuela', 'México', 'Brasil', 'Belize']
                          )
print ('DataFrame a partir de una estructura tipo dict')
print (dataFrame)
```

```
DataFrame a partir de una estructura tipo dict
           Población      Idioma
Colombia        51.0     Español
Perú            32.0     Español
Argentina       45.0     Español
Surinam          0.5  Neerlandés
Nicaragua        6.5     Español
Chile           19.5     Español
Honduras         9.5     Español
Guatemala       17.0     Español
Paraguay         7.0     Español
```

```
Uruguay          3.5     Español
Bolivia         11.5     Español
Venezuela       28.5     Español
México         126.0     Español
Brasil         212.0   Portugués
Belize           0.4      Inglés
```

[8]:
```
# obtener solo la población
# Se puede acceder a cada columna de un marco de datos utilizando el nombre de la columna
# como atributo (o, alternativamente, indexando con la etiqueta de la columna,
# por ejemplo, dataFrame ['Población']):
print ('Poblaciones :')
# poblaciones = dataFrame.Población
poblaciones = dataFrame ['Población']
print (poblaciones)
```

```
Poblaciones :
Colombia      51.0
Perú          32.0
Argentina     45.0
Surinam        0.5
Nicaragua      6.5
Chile         19.5
Honduras       9.5
Guatemala     17.0
Paraguay       7.0
Uruguay        3.5
Bolivia       11.5
Venezuela     28.5
México       126.0
Brasil       212.0
Belize         0.4
Name: Población, dtype: float64
```

Las filas de una instancia de DataFrame se puede acceder mediante el atributo loc indexer. La indexación de este atributo también da como resultado un objeto Serie, que corresponde a una fila del DataFrame original:

[12]:
```
dataFrame.loc['México']
```

[12]:
```
Población        126
Idioma       Español
Name: México, dtype: object
```

Si se pasa una lista de etiquetas de filas al indexador loc, se obtiene un nuevo DataFrame que es un subconjunto del DataFrame original, que contiene sólo las filas seleccionadas

[15]:
```
dataFrame.loc[['México', 'Colombia']]
```

[15]:
```
           Población   Idioma
México         126.0  Español
Colombia        51.0  Español
```

El indexador loc también puede utilizarse para seleccionar simultáneamente filas y columnas pasando primero una etiqueta de fila (o una lista de ellas) y segundo una etiqueta de columna (o una lista de las mismas). El resultado es un DataFrame, una Serie, o un valor de elemento, dependiendo del número de columnas y filas que se seleccionen

[17]:
```
dataFrame.loc[['México', 'Colombia'],'Población']
```

[17]:
```
México      126.0
Colombia     51.0
Name: Población, dtype: float64
```

[19]:
```
dataFrame.loc[['México', 'Colombia'],'Idioma']
```

[19]:
```
México      Español
Colombia    Español
Name: Idioma, dtype: object
```

Podemos calcular las estadísticas descriptivas utilizando los mismos métodos que ya utilizamos para los objetos Serie. Al invocar esos métodos (media, std, mediana, min, max, etc.) para un DataFrame, el cálculo se realiza para cada columna con tipos de datos numéricos:

[20]:
```
dataFrame.mean()
```

[20]:
```
Población    37.993333
dtype: float64
```

En este caso, sólo una de las dos columnas tiene un tipo de dato numérico (la denominada Población). Utilizando el método DataFrame info y el atributo dtypes, podemos obtener un resumen del contenido de un DataFrame y los tipos de datos de cada columna:

[21]:
```
dataFrame.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 15 entries, Colombia to Belize
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Población  15 non-null     float64
 1   Idioma     15 non-null     object
dtypes: float64(1), object(1)
memory usage: 1000.0+ bytes
```

[ ]: