Introducción a Pandas

Martínez Cerda Mario Antonio

29 de enero de 2021

1 Introducción.

La biblioteca Pandas de Python fue desarrollada por Wes McKinney para manipulación y análisis de datos. La biblioteca Pandas fue hecha pública en 2008. La versión más actuali es la 1.2.0 (26 de diciembre de 2020). Durante el desarrollo del siguiente documento, usaremos las funciones principales de Pandas para el análisis de los datos climatológicos vistos en la actividad 1.

2 Actividades a realizar.

2.1 Paso 1.

Abre un nuevo cuaderno de trabajo Jupyter en Google Colab, llamado Actividad 3. Carga las bibliotecas Numpy y Pandas. Enseguida define un nuevo DataFrame df con la ayuda de la función de Pandas pd.read_csv, leyendo el archivo csv con los datos climatológicos de la estación meteorológica que has seleccionado para estas actividades y que has almacenado en tu repositorio en Github.

2.2 Paso 2.

Una vez que hayas leído tu archivo de datos, usando la función df.head(n) y df.tail(), y estudia la estructura de tu archivo, para determinar que renglones trae la información de los nombres de las columnas y en qué número de renglón comienzan los datos. Lo que buscamos es quedarnos con un DataFrame con los nombres de las columnas en el primer renglón y el resto de renglones serán sólo datos diarios de la estación meteorológica.

2.3 Paso 3.

Enseguida se pide explorar el dataframe recién cargado a la memoria. Para esto, se pide hacer una bitácora anotando o agregando un comentario de las acciones desarrolladas.

* ¿Qué dimensiones tiene tu dataframe?: df.shape

* ¿Cómo es el contenido de tu dataframe?: df.info()
* Los datos originales incluyen la cadena de caracteres 'Nulo', indicando que no hubo datos para esa variable, ese día. Tenemos que reemplazar la palabra nulo con la función: df.replace()
* Después habrá que convertir a número flotante o numérico los datos de Precipitación, Evaporación, Temperatura Máxima y Temperatura mínima utilizando la función: df.to_numeric()
* Contrasta ahora la información de tu DataFrame con la función: df.info() Se puede contabilizar el número de datos faltantes en esas variables mediante la función: df.isnull().sum()
* Imprime de nuevo el encabezado y final de tu dataframe, con las funciones df.head() y df.tail()
* Realiza una estadística básica de las variables numéricas de tu dataframe usando la función: df.describe() y haz una interpretación de los resultados, para ver si tienen sentido físico (por ej. valores negativos de precipitación, valores extremos fuera de lo normal, etc)

2.4 Paso 4.

Análisis de la variable Fecha. Pandas maneja las variables tipo Fecha y Tiempo. Ofrece una serie de herramientas para su manipulación. Lo que sigue es convertir el objeto que se ha leído a una variable que Python comprenda.

* Comienza haciendo una copia del dataframe del paso anterior, por si se requiere recuperar el nuevo dataframe, utilizando df:new=df.copy()
* Utiliza la función de Pandas pd.to_datetime() para convertir el objeto Fecha a formato de fecha que comprende Python. Lee el manual de esa función e intenta la conversión.
* Después utiliza la función df.dtypes para verificar que todas las variables son del tipo deseado.
* Enseguida con la ayuda de las funciones de Pandas df['Fecha'].dt.year y df['Fecha'].dt.month, crea dos columnas nuevas adicionales df['Año'] y df['Mes'].
* Utilizando la función print aplicado a las funciones df.head() y df.tail(), verifica que el dataframe tiene la forma deseada.
* Y complementa con la función df.dtypes, para verificar que todas las variables son del tipo adecuado.

3 Resultados.

Durante el desarrollo de la actividad, observamos que Pandas es una diversidad de tablas y análisis de diferentes tipos, que es de gran ayuda para la estadística, especialmente para ver el significado de los datos presentados. De fácil uso y de gran utilidad para trabajos académicos.

Los siguientes datos son imágenes, que se pueden ver con más detenimiento en el archivo hecho en Google Colab, disponible en mi Github.

Actividad a realizar.

Paso 1.-Abre un nuevo cuaderno de trabajo Jupyter en Google Colab, llamado Actividad3. Carga las bibliotecas Numpy y Pandas. Enseguida define un nuevo DataFrame df con la ayuda de la función de Pandas pd.read_csv, leyendo el archivo csv con los datos climatológicos de la estación meteorológica que has seleccionado para estas actividades y que has almacenado en tu repositorio en Github.

```
[ ] #Primeramente lo que hacemos es importar las bibliotecas correspondientes: Pandas y numpy. Después lo que se hará es definir la URL que viene de nuestro Github.
    #Después definimos nuestras columnas iguales o muy parecidas a las que vienen en el archivo de texto que queremos leer.
    #Definimos ahora el DataFrame con las carácterísticas necesarias para que lea nuestro csv. En este caso, como hablamos en un idioma español
    #Es necesario colocar el encoding correspondiente de pandas. Después ingresamos las columnas que quiere leer.
    #Con header decimos que no habrá encabezados.
    #Gracias a las opciones skip (row o footer) nos encargamos de saltarnos los renglones que no nos importan en el archivo, para que lea solo los datos correspondie
    #Como estamos trabajando con python, aunque suene obvio, al final ingresamos el engine con el que trabajaremos.
    #Al correr el programa no debería saltar ningún problema.
    import numpy as np
    import pandas as pd
    url = 'https://raw.githubusercontent.com/MarioMtzC99/FisicaComputacional1/main/Actividad1/DatosClimatolog%C3%ADaPitiquito.txt'
    mis_columnas = ['Fecha', 'Precip', 'Evap', 'Tmax', 'Tmin']
    df_carac = pd.read_csv(url, names=mis_columnas, encoding='cp1251', sep='\s+', header=None, skiprows=19, skipfooter=1, engine='python')
```

Paso 2.-Una vez que hayas leído tu archivo de datos, usando la función df.head(n) y df.tail(), y estudia la estructura de tu archivo, para determinar que renglones trae la información de los nombres de las columnas y en qué número de renglón comienzan los datos. Lo que buscamos es quedarnos con un DataFrame con los nombres de las columnas en el primer renglón y el resto de renglones serán sólo datos diarios de la estación meteorológica.

```
[ ] #Podemos observar como el archivo por lo menos comienza bien y termina bien.
    print(df_carac.head())
    print(df_carac.tail())


    #Podemos observar que está justo como nos interesa, ya que los nombres están en el primer renglón.
    #Además nos interesa quitar ese "Nulo", debido a que queremos que Python lo vea como un NaN y no como caracteres.
```

```
        Fecha Precip  Evap  Tmax  Tmin
0  01/07/1952      0  Nulo    42    22
1  02/07/1952      0  Nulo  42.5    25
2  03/07/1952      0  Nulo    43  25.5
3  04/07/1952      0  Nulo    44  27.5
4  05/07/1952      0  Nulo  43.5  25.5
            Fecha Precip  Evap Tmax  Tmin
21580  27/12/2015      0  Nulo   16    -4
21581  28/12/2015      0  Nulo   15    -4
21582  29/12/2015      0  Nulo   15  -2.5
21583  30/12/2015      0  Nulo   16    -3
21584  31/12/2015      0  Nulo   14    -1
```

Paso 3.-Enseguida se pide explorar el dataframe recién cargado a la memoria. Para esto, se pide hacer una bitácora anotando o agregando un comentario de las acciones desarrolladas.

- Imprime el encabezado y final del dataframe: print(dif.head(10)),print(df.tail(10))

```
[ ]  print(df_carac.head(10))
     print(df_carac.tail(10))
```

```
         Fecha Precip  Evap  Tmax  Tmin
0  01/07/1952      0  Nulo    42    22
1  02/07/1952      0  Nulo  42.5    25
2  03/07/1952      0  Nulo    43  25.5
3  04/07/1952      0  Nulo    44  27.5
4  05/07/1952      0  Nulo  43.5  25.5
5  06/07/1952      0  Nulo    44    26
6  07/07/1952      0  Nulo    42    24
7  08/07/1952      0  Nulo    41  26.5
8  09/07/1952      0  Nulo    40    24
9  10/07/1952   38.8  Nulo  40.5    25
             Fecha Precip  Evap  Tmax  Tmin
21575  22/12/2015      0  Nulo    20     1
21576  23/12/2015      0  Nulo    21     4
21577  24/12/2015      0  Nulo    21   4.5
21578  25/12/2015      0  Nulo  19.5     2
21579  26/12/2015      0  Nulo  12.5     3
21580  27/12/2015      0  Nulo    16    -4
21581  28/12/2015      0  Nulo    15    -4
21582  29/12/2015      0  Nulo    15  -2.5
21583  30/12/2015      0  Nulo    16    -3
21584  31/12/2015      0  Nulo    14    -1
```

- ¿Qué dimensiones tiene el dataframe?

```
]  df_carac.shape
   #Podemos observar que tiene 21585 filas y 5 columnas)
```

```
(21585, 5)
```

- ¿Cómo es el contenido de tu dataframe?

```
]  df_carac.info()
   #Tiene 21585 entradas con 5 columnas, la memoria usada es de 842.3+ KB.
   #Se observa que los datos los tiene como objetos y no como números.
   #En cada columna especifica que todos los datos son no nulos.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21585 entries, 0 to 21584
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Fecha   21585 non-null  object
 1   Precip  21585 non-null  object
 2   Evap    21585 non-null  object
 3   Tmax    21585 non-null  object
 4   Tmin    21585 non-null  object
dtypes: object(5)
memory usage: 843.3+ KB
```

- Los datos originales incluyen la cadena de caracteres 'Nulo', indicando que no hubo datos para esa variable, ese día. Tenemos que reemplazar la palabra nulo con la función: df.replace()

```
df_work = df_carac.copy()

str_Nulo = 'Nulo'
df_trabajo = df_work.replace(to_replace=str_Nulo, value='', regex=True)
df_trabajo.head()
```

	Fecha	Precip	Evap	Tmax	Tmin
0	01/07/1952	0		42	22
1	02/07/1952	0		42.5	25
2	03/07/1952	0		43	25.5
3	04/07/1952	0		44	27.5
4	05/07/1952	0		43.5	25.5

- Después habrá que convertir a número flotante o numérico los datos de Precipitación, Evaporación, Temperatura Máxima y Temperatura mínima utilizando la función: df.to_numeric()

```
#Definimos la variable que  tendrá las columnas que queremos cambiar.
columnas_list = ['Precip', 'Evap', 'Tmax', 'Tmin']
#Agregamos un Loop para que los datos de las columnas sean numeros flotantes.
for columnas in columnas_list:
  df_trabajo[columnas] = pd.to_numeric(df_trabajo[columnas], errors='coerce')
```

- Contrasta ahora la información de tu DataFrame con la función: df.info()

```
#pedimos la información el dataframe modificado.
df_trabajo.info()

#Observaremos que ahora las columnas que definimos son del tipo float64.(Números reales)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21585 entries, 0 to 21584
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Fecha   21585 non-null  object
 1   Precip  21536 non-null  float64
 2   Evap    14380 non-null  float64
 3   Tmax    21528 non-null  float64
 4   Tmin    21529 non-null  float64
dtypes: float64(4), object(1)
memory usage: 843.3+ KB
```

- Se puede contabilizar el número de datos faltantes en esas variables mediante la función: df.isnull().sum()

```
df_trabajo.isnull().sum()
#Vemos cuantos datos nulos o vacios hay en cada columna.
```

```
Fecha        0
Precip      49
Evap      7205
Tmax        57
Tmin        56
dtype: int64
```

- Imprime de nuevo el encabezado y final de tu dataframe, con las funciones df.head() y df.tail()

```
] #Ahora observamos como queda nuestra tabla actualizada.
  print(df_trabajo.head())
  print(df_trabajo.tail())
```

```
        Fecha  Precip  Evap  Tmax  Tmin
0  01/07/1952     0.0   NaN  42.0  22.0
1  02/07/1952     0.0   NaN  42.5  25.0
2  03/07/1952     0.0   NaN  43.0  25.5
3  04/07/1952     0.0   NaN  44.0  27.5
4  05/07/1952     0.0   NaN  43.5  25.5
            Fecha  Precip  Evap  Tmax  Tmin
21580  27/12/2015     0.0   NaN  16.0  -4.0
21581  28/12/2015     0.0   NaN  15.0  -4.0
21582  29/12/2015     0.0   NaN  15.0  -2.5
21583  30/12/2015     0.0   NaN  16.0  -3.0
21584  31/12/2015     0.0   NaN  14.0  -1.0
```

- Realiza una estadística básica de las variables numéricas de tu dataframe usando la función: df.describe() y haz una interpretación de los resultados, para ver si tienen sentido físico (por ej. valores negativos de precipitación, valores extremos fuera de lo normal, etc)

```
] #Usamos cuatro cifras significativas.
  df_trabajo.describe().round(4)
```

	Precip	Evap	Tmax	Tmin
count	21536.0000	14380.0000	21528.0000	21529.0000
mean	0.6742	7.1673	31.3675	12.3381
std	3.7723	3.7375	8.1094	7.8970
min	0.0000	0.0000	4.0000	-11.0000
25%	0.0000	4.0000	25.0000	6.0000
50%	0.0000	6.9000	32.0000	11.0000
75%	0.0000	9.9250	38.5000	19.0000
max	97.0000	18.0000	48.0000	30.5000

Podemos ver en un análisis a simple vista que las precipitaciones se encuentran dentro de lo normal esperado en el desierto. Sin tener datos que no tengan una interpretación extraña.
En cuanto a evaporación, podemos ver por los datos que la curva se presenta un poco más alto hacia la derecha en el tercer cuartil, lo que nos indica que las precipitaciones se presentan más a menudo durante los últimos meses del año. Al igual, como se presentan las precipitaciones se presentan las humedades.

La temperatura media máxima presente en el municipio de Pitiquito es de 31 grados centígrados y una mínima de 12.33 grados centígrados.

Paso 4.-Análisis de la variable Fecha. Pandas maneja las variables tipo Fecha y Tiempo. Ofrece una serie de herramientas para su manipulación. Lo que sigue es convertir el objeto que se ha leído a una variable que Python comprenda.

- Comienza haciendo una copia del dataframe del paso anterior, por si se requiere recuperar el nuevo dataframe, utilizando df:new=df.copy()

```
[ ] #Realizamos una copia nueva para no perder el dataframe.
    df_paso4 = df_trabajo.copy()
    #Mostramos la columna de fecha.
    df_paso4['Fecha'].head()
```

```
0    01/07/1952
1    02/07/1952
2    03/07/1952
3    04/07/1952
4    05/07/1952
Name: Fecha, dtype: object
```

- Utiliza la función de Pandas pd.to_datetime(), para convertir el objeto Fecha a formato de fecha que comprende Python. Lee el manual de esa función e intenta la conversión.

```
#Lo que hacemos es establecer una copia del cambio de la fecha con ayuda de
#pd.to_datetime.
#Definimos que queremos que la nueva copia cambie la Fecha, considerando el día como lo primero que debe aparecer.
df_paso4['Fecha']=pd.to_datetime(df_paso4['Fecha'], dayfirst=True).copy()
#Imprimimos para ver como se observa.
print(df_trabajo.head())
```

```
        Fecha  Precip  Evap  Tmax  Tmin
0  01/07/1952     0.0   NaN  42.0  22.0
1  02/07/1952     0.0   NaN  42.5  25.0
2  03/07/1952     0.0   NaN  43.0  25.5
3  04/07/1952     0.0   NaN  44.0  27.5
4  05/07/1952     0.0   NaN  43.5  25.5
```

- Después utiliza la función df.dtypes para verificar que todas las variables son del tipo deseado.

```
#Con dtypes, vemos que la fecha es del tipo datetime, y las demás columnas del tipo real, como lo queremos.
df_paso4.dtypes
```

```
Fecha     datetime64[ns]
Precip           float64
Evap             float64
Tmax             float64
Tmin             float64
dtype: object
```

- Enseguida con la ayuda de las funciones de Pandas df['Fecha'].dt.year y df['Fecha'].dt.month, crea dos columnas nuevas adicionales df['Año'] y df['Mes'].

```
#Definimos Año y Mes para separar la fechas.
df_paso4['Año'] = df_paso4['Fecha'].dt.year
df_paso4['Mes'] = df_paso4['Fecha'].dt.month
```

- Utilizando la función print aplicado a las funciones df.head() y df.tail(), verifica que el dataframe tiene la forma deseada.

```
print(df_paso4.head())
print(df_paso4.tail())
```

```
       Fecha  Precip  Evap  Tmax  Tmin   Año  Mes
0 1952-07-01     0.0   NaN  42.0  22.0  1952    7
1 1952-07-02     0.0   NaN  42.5  25.0  1952    7
2 1952-07-03     0.0   NaN  43.0  25.5  1952    7
3 1952-07-04     0.0   NaN  44.0  27.5  1952    7
4 1952-07-05     0.0   NaN  43.5  25.5  1952    7
           Fecha  Precip  Evap  Tmax  Tmin   Año  Mes
21580 2015-12-27     0.0   NaN  16.0  -4.0  2015   12
21581 2015-12-28     0.0   NaN  15.0  -4.0  2015   12
21582 2015-12-29     0.0   NaN  15.0  -2.5  2015   12
21583 2015-12-30     0.0   NaN  16.0  -3.0  2015   12
21584 2015-12-31     0.0   NaN  14.0  -1.0  2015   12
```

- Y complementa con la función df.dtypes, para verificar que todas las variables son del tipo adecuado.

```
] #Verificamos que son del tipo que deseamos.
  df_paso4.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21585 entries, 0 to 21584
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype         
---  ------  --------------  -----         
 0   Fecha   21585 non-null  datetime64[ns]
 1   Precip  21536 non-null  float64       
 2   Evap    14380 non-null  float64       
 3   Tmax    21528 non-null  float64       
 4   Tmin    21529 non-null  float64       
 5   Año     21585 non-null  int64         
 6   Mes     21585 non-null  int64         
dtypes: datetime64[ns](1), float64(4), int64(2)
memory usage: 1.2 MB
```

4 Conclusión.

Como actividad introductoria a la biblioteca Pandas fue bastante sencilla y clara. Cada una de las funciones que se presenta en la actividad tienen un fácil uso y una gran variedad de opciones para analizar los datos. Lo que más se me dificulta en la realización de estas actividades es determinar las funciones más especificas que ayudan con el comportamiento de los datos, pero es solo cuestión de estudiar y poner en práctica las guías. El grado de complejidad me pareció intermedio, debido a que las ayudas que nos proporciona el maestro sobre la actividad es bastante clara. La gestión de tiempo es clave en este tipo de trabajos por que suelen ser olvidadas algunas de las funciones que se nos enseñan.

5 Referencias.

1.-https://pandas.pydata.org/docs/getting_started/intro_tutorials/index.html

2.-https://www.learndatasci.com/tutorials/python-pandas-tutorial-complete-introduction-for-beginners/

3.-http://computacional1.pbworks.com/w/page/142921350/Actividad%203%20(2021-1)