

# Actividad 8

José Daniel Gaytán Villarreal  
Grupo 3

4 de abril de 2019

## Resumen

Se tomaron dos series de datos independientes y se modificaron de tal manera que se pudieran manejar como una serie conjunta, para después obtener ciertas gráficas de sus valores y realizar las interpretaciones propias.

## 1. Introducción

La principal dificultad del análisis de datos no es, ciertamente, la interpretación de los mismos, sino el saber cómo manejarlos para poder trabajar con ellos. Tal es el caso de esta actividad, en donde contando con dos series de datos, de distinta longitud y con parámetros similares pero con notables diferencias, se trabajó principalmente en su modificación para poder trabajar con ellos de manera conjunta. Se buscó, pues, unificar dos series de datos en una única serie que estuviera de acuerdo a nuestros intereses.

El objetivo que se siguió a lo largo de la actividad fue el de modificar ambas series, con distintas funciones intrínsecas a Python, hasta llegar a un punto en que fuesen lo suficientemente similares para poder unirlas.

En la sección 2, se habla brevemente sobre los métodos a seguir durante el desarrollo de la actividad, así como algunas dificultades con las que se topó durante el desarrollo de la misma; a su vez, se mostrarán los resultados obtenidos en la misma sección, constando en su mayoría de gráficas que ejemplifican la armonía y sencillez con la que se puede trabajar con los datos una vez unidos. Por último, en la sección 3, se emite una breve conclusión referente a la metodología utilizada y sus subsecuentes resultados.

## 2. Desarrollo

### 2.1. Metodología

Trabajar con dos series de datos no es sencillo, menos aún cuando éstas difieren en longitud y criterios de medición. Primero que nada, se tuvo que reducir nuestra primera serie, la cual contenía mediciones de datos cada diez minutos desde inicios del 2009 a finales del 2010, a los datos tomados del año 2009 cada media hora. Lo anterior se logró a partir de condicionales lógicos intrínsecos al lenguaje Python:

```
df1 = df1[((df1["MIN"] == 30.0) | (df1["MIN"] == 0.0)) & (df1["FECHA"] < '2010-01-01 00:00:00')]  
df1 = df1.reset_index(drop=True)  
df1.tail()
```

Dado que la primera serie de datos venía ya con una variable de fecha asignada, su modificación fue relativamente sencilla. La segunda serie, sin embargo, carecía de dicha variable, por lo que fue necesario crear una a partir de los datos presentes.

En la segunda serie, el los datos se tomaban cada media hora, aunque se expresaban como números enteros sin separar horas de minutos (30,100,130,200,230,...). Para reescribirlos de manera adecuada, se aplicó el siguiente condicional:

```
h=[] #Arreglo de horas.  
m=[] #Arreglo de minutos.  
  
for i in range (0, len(df2)):  
    if (len(str(df2['4 Hour_Minute	RTM L'][i]))==4):  
        if (str(df2['4 Hour_Minute	RTM L'][i])[0:2]=='24'): #2400  
            h.append('00')  
            m.append('00')  
        else:  
            h.append(str(df2['4 Hour_Minute	RTM L'][i])[0:2])  
            m.append(str(df2['4 Hour_Minute	RTM L'][i])[2:4])  
    elif (len(str(df2['4 Hour_Minute	RTM L'][i]))==3): #1300  
        h.append(str(df2['4 Hour_Minute	RTM L'][i])[0:1])  
        m.append(str(df2['4 Hour_Minute	RTM L'][i])[1:3])  
    elif (len(str(df2['4 Hour_Minute	RTM L'][i]))==2): #100  
        h.append('00')  
        m.append(str(df2['4 Hour_Minute	RTM L'][i])[0:2])  
  
    t = [] #Arreglo de tiempo  
for i in range (0,len(df2)):  
    t.append( h[i]+':'+m[i])  
  
df2['TIEMPO'] = t
```

El código anterior reescribía los valores de tiempo a un formato de hora del tipo HH:MM:SS; por ejemplo, 100 a 1:00, 1300 a 13:00, 2400 a 00:00, etc.

Una vez establecida la variable de tiempo, se prosiguió a construir una variable del tipo *date* que permitiera a Python asignarle un valor temporal a cada medición de datos. La construcción de dicha variable se realizó de la siguiente manera:

```
f = [] #Arreglo de fechas
for i in range (0,len(df2)):
    f.append('2009 ' + str(df2['DIA'][i]) + ' ' + h[i] + ' ' + m[i])

FECHA = []
for i in range(0,len(df2)):
    d=datetime.datetime.strptime(f[i],'%Y %j %H %M')
    F = d.isoformat(' ')
    FECHA.append(F)

df2['F']=FECHA

df2['FECHA'] = pd.to_datetime(df2.apply(lambda x: x['F'], 1), dayfirst=True)
```

Cabe mencionar un error crítico que surgió al construir la variable de tiempo: en la segunda serie de datos, las 00:00 horas contaban como si fueran *del día anterior*, cuando marcan en realidad el inicio del nuevo día. Para solucionar este aparente confusión temporal, bastaron las siguientes líneas de código:

```
d = []
j = -1
for i in range (0,len(df2)):
    if((df2['TIEMPO'][i]=='00:00')):
        d.append(df2['3 Day_RTM_L'][i]+1)
    elif((df2['TIEMPO'][i]!='00:00')):
        d.append(df2['3 Day_RTM_L'][i])
```

La función del código anterior es que si el dato presenta la hora *00:00*, se le suma 1 a la variable que almacena el número de día para esa medición.

Una vez que ambas series de datos contaban con variables de fecha de igual estructura, se eliminaron los datos duplicados de cada serie y se procedió a crear un Dataframe que juntase ambas en una serie conjunta:

```
df1 = pd.DataFrame(df1.drop_duplicates(['FECHA']))
df2 = pd.DataFrame(df2.drop_duplicates(['FECHA']))

df = pd.merge(df1, df2, on=['FECHA'])
```

El Dataframe resultante lucía de la siguiente manera:

	FECHA	1_Avg	2xT_Avg	Tensio_15cm	Tensio_20cm	Tensio_25cm	Tensio_30cm	Tensio_40cm	Tensio_50cm	Tensio_75cm	Tensio_90cm	Tensio_100cm
0	2009-01-01 00:30:00	-05.4	9.079999	14.36	14.70	15.22	15.52	16.11	17.02	17.75	18.62	
1	2009-01-01 01:00:00	8.76	8.34	14.33	14.70	15.24	15.52	16.11	17.02	17.75	18.62	
2	2009-01-01 01:30:00	8.69	7.539001	14.28	14.69	15.24	15.54	16.11	17.02	17.74	18.62	
3	2009-01-01 02:00:00	7.62	6.78	14.24	14.69	15.25	15.55	16.12	17.02	17.74	18.60	
4	2009-01-01 02:30:00	7.61	7.268001	14.19	14.69	15.27	15.55	16.12	17.02	17.75	18.60	

Figura 1: Unión de dos DataFrames de misma variable fecha y diferentes variables de medición.

## 2.2. Resultados

Del análisis anterior se realizaron las siguientes gráficas, las cuales se muestran a continuación:

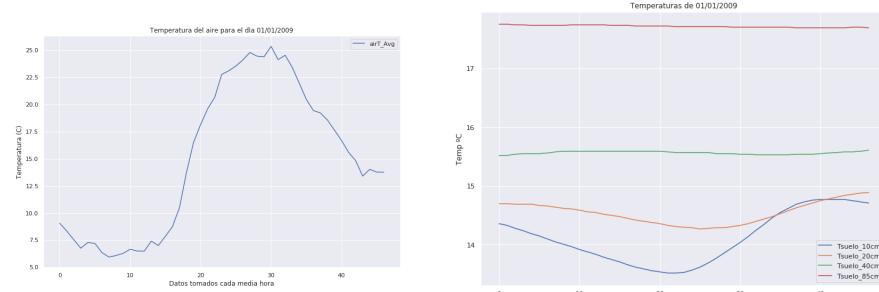


Figura 2: Temperatura del aire y del suelo para el día 1ero de enero de 2009.

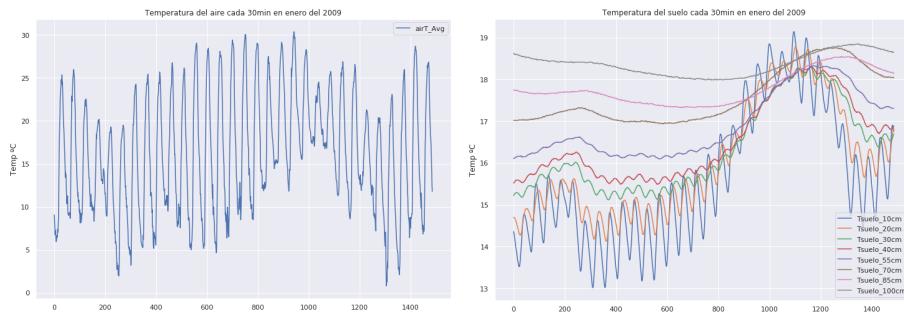


Figura 3: Temperatura del aire y del suelo para el mes de enero de 2009.

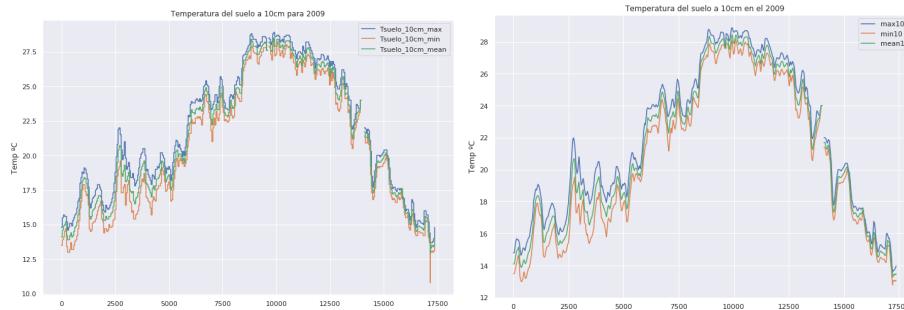


Figura 4: Gráficas de las temperaturas medidas 10cm bajo suelo, antes y después de aplicar el promedio móvil.

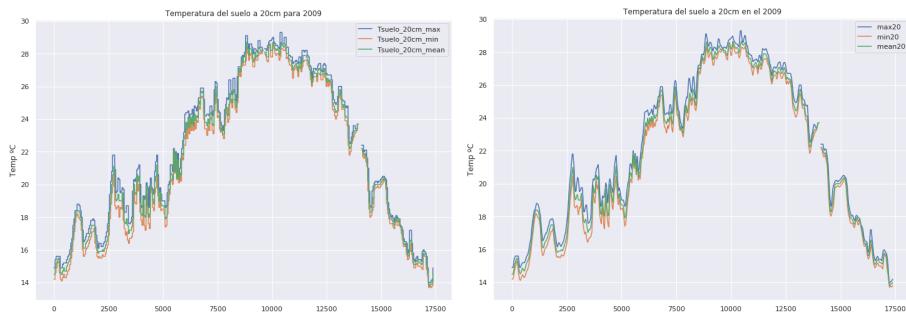


Figura 5: Gráficas de las temperaturas medidas 20cm bajo suelo, antes y después de aplicar el promedio móvil.

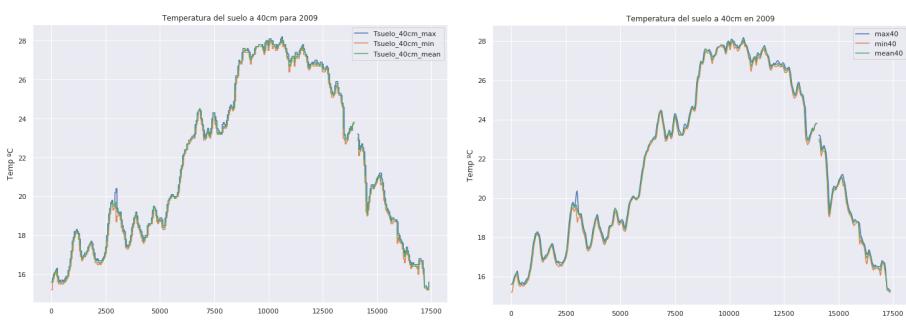


Figura 6: Gráficas de las temperaturas medidas 40cm bajo suelo, antes y después de aplicar el promedio móvil.

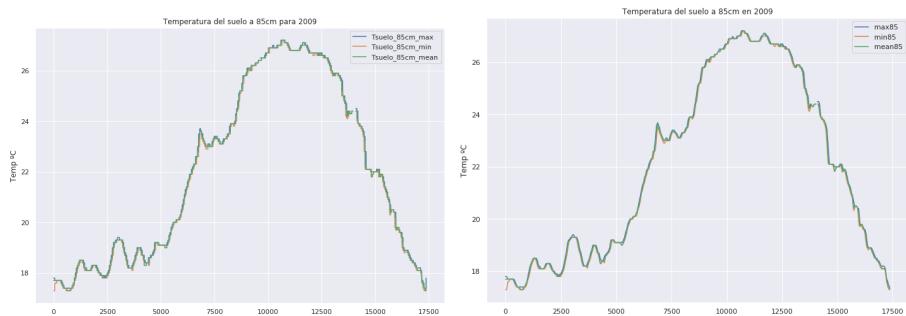


Figura 7: Gráficas de las temperaturas medidas 85cm bajo suelo, antes y después de aplicar el promedio móvil.

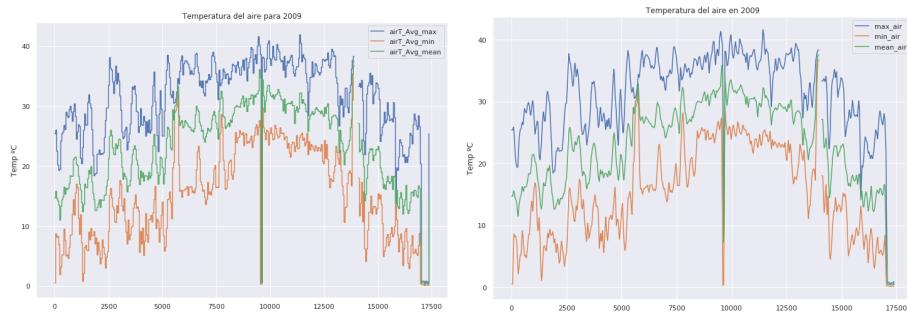


Figura 8: Temperatura del aire para el año 2009, antes y después de aplicar el promedio móvil.

### **3. Conclusión**

Interpretando los datos, vemos un patrón recurrente en las temperaturas del suelo: a mayor profundidad, menor será la diferencia entre la temperatura máxima, mínima y promedio, lo que indica que después de cierta profundidad la temperatura se estabiliza. En cuanto al aire, es curioso observar su naturaleza aparentemente ondulatoria, notándose como a partir de cierta hora del día ésta sube a un punto máximo y vuelve a decaer.

Por otro lado, merece la pena recalcar la utilidad de obtener un *promedio móvil* (rolling mean) a una serie de datos, ya que, como observamos, brinda a las gráficas mayor suavidad, lo que facilita su interpretación, a costa de una pequeña perdida de datos.

La actividad fue, en general, un complejo ejercicio para aprender a manejar series de datos, cuyo resultado se puede apreciar en las gráficas conjuntas que permiten una mejor interpretación de los mismos.