Información climática

César Andrés Pérez Robinson

February 2019

0.1. Introducción

A continuación se presenta una serie de actividades realizadas para aprender a utilizar "jupyter notebook", las actividades estuvieron relacionadas a la creación de tablas y gráficas estadísticas sobre datos climáticos de una ciudad de la República. En el caso particular del presente trabajo, se presentan los datos obtenidos por la página http://smn1.conagua.gob.mx/emas/ de la ciudad de Cancún desde Enero 23 a Enero 29 del 2019.

0.2. Procedimiento

A continuación se presentan los pasos que se utilizaron en "jupyter notebook"para concluir la actividad.
Entrada [1]; Se descarga Pandas, para el manejo de datos y se le asigna las letras *pd* de manera que al usarse solo basta con utilizar *pd*. Mismo procedimiento con numpy y matplolib.

```
# Cargar a la memoria de trabajo las bibliotecas: Pandas (manejo de datos,
# Numpy (numerical python) y la biblioteca de gráficas Matplotlib
# Se asignan nombres cortos.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
#
# Usar "Shift+Enter" para procesar la información de la celda
```

Entrada [2]; Se descargaron los datos de la estación del Servicio Meteorológico Nacional en http://smn1.conagua.gob.mx/emas/ de Cancún.

```
df0 = pd.read_csv('cancun.txt', skiprows=4, sep='\s+')
```

Entrada [3]; Se piden los primeros cinco renglones del texto y se obtiene la Figura 1.

```
df0.head()
```

Figura 1: Primeros cinco renglones de texto

	DD/MM/AAAA	HH:MM	DIRS	DIRR	VELS	VELR	TEMP	HR	PB	PREC	RADSOL
0	22/01/2019	22:00	129.0	119.0	15.51	34.4	24.0	93.0	1016.2	1.0	62.5
1	22/01/2019	23:00	141.0	147.0	18.12	35.0	24.2	86.0	1016.2	0.3	7.0
2	23/01/2019	00:00	141.0	125.0	19.95	39.0	24.8	81.0	1016.4	0.0	0.0
3	23/01/2019	01:00	134.0	156.0	19.29	40.8	25.0	79.0	1016.8	0.0	0.0
4	23/01/2019	02:00	143.0	157.0	22.24	41.9	25.3	76.0	1017.0	0.0	0.0

Entrada [4]; Se le da una estructura de datos.

```
df = pd.DataFrame(df0)
```

Entrada [5]; Se ven los tipos de datos que Pandas ha reconocido al leer.

```
df.dtypes

Out[5]
  DD/MM/AAAA      object
  HH:MM           object
```

```
DIRS          float64
DIRR          float64
VELS          float64
VELR          float64
TEMP          float64
HR            float64
PB            float64
PREC          float64
RADSOL        float64
dtype: object
```

Entrada [6]; A continuación convierte una variable de tiempo, combinando las columnas DD/MM/AAAA con HH:MM para crear la nueva columna FECHA, la cual se presenta en un formato de tiempo. Adicionalmente, se eliminan las dos columnas anteriores.

```
df['FECHA'] = pd.to_datetime(df.apply(lambda x: x['DD/MM/AAAA'] + ' ' +
x['HH:MM'], 1), dayfirst=True)
df = df.drop(['DD/MM/AAAA', 'HH:MM'], 1)
```

Entrada[7]; Utilizando el comando anterior para obtener los primeros cinco reglones del texto, observamos en la Figura 2 que las variables de DD/MM/AAAA y HH:MM se han eliminado y se ha creado la variable FECHA.

```
df.head()
```

Figura 2: Primeros cinco renglones de texto, con variable FECHA

	DIRS	DIRR	VELS	VELR	TEMP	HR	PB	PREC	RADSOL	FECHA
0	129.0	119.0	15.51	34.4	24.0	93.0	1016.2	1.0	62.5	2019-01-22 22:00:00
1	141.0	147.0	18.12	35.0	24.2	86.0	1016.2	0.3	7.0	2019-01-22 23:00:00
2	141.0	125.0	19.95	39.0	24.8	81.0	1016.4	0.0	0.0	2019-01-23 00:00:00
3	134.0	156.0	19.29	40.8	25.0	79.0	1016.8	0.0	0.0	2019-01-23 01:00:00
4	143.0	157.0	22.24	41.9	25.3	76.0	1017.0	0.0	0.0	2019-01-23 02:00:00

Entrada[8]; Realizando un análisis exploratorio de datos obtenemos la Figura 3.

```
df.describe()
```

Entrada[8]; Podemos seleccionar un apartado específico de datos, por ejemplo, aquellos datos en los cuales la Temperatura era mayor a $24°C$ y menor a $25°C$, lo que se muestra en la Figura 4.

```
# Selecciona los renglones con Temperatura > 24°C y < 25°C
df_tmp = df[df.TEMP > 24]
df_select = df_tmp[df_tmp.TEMP < 25]
df_select
```

Entrada[9]; Podemos calcular el promedio de todas las columnas utilizando:

```
df.mean()
Out[9]
DIRS       164.604317
DIRR       165.920863
VELS        11.456835
```

Figura 3: Análisis exploratorio de datos

	DIRS	DIRR	VELS	VELR	TEMP	HR	PB	PREC	RADSOL
count	139.000000	139.000000	139.000000	139.000000	139.000000	139.000000	139.000000	139.000000	138.000000
mean	164.604317	165.920863	11.456835	22.905755	22.956835	87.949640	1014.661871	2.166187	120.388406
std	116.263483	115.591426	4.634042	9.801706	3.531392	13.340754	1.717700	18.847883	203.122739
min	1.000000	0.000000	2.070000	7.900000	13.800000	51.000000	1006.300000	0.000000	0.000000
25%	69.000000	77.000000	8.210000	15.150000	21.200000	78.000000	1013.750000	0.000000	0.000000
50%	136.000000	148.000000	10.380000	21.200000	23.200000	92.000000	1014.700000	0.000000	0.000000
75%	299.000000	294.500000	14.050000	28.700000	25.350000	100.000000	1015.700000	0.000000	161.550000
max	358.000000	359.000000	23.800000	66.100000	29.900000	100.000000	1018.500000	221.000000	836.200000

Figura 4: Temperatura $> 24°C$ y $< 25°C$

	DIRS	DIRR	VELS	VELR	TEMP	HR	PB	PREC	RADSOL	FECHA
1	141.0	147.0	18.12	35.0	24.2	86.0	1016.2	0.3	7.0	2019-01-22 23:00:00
2	141.0	125.0	19.95	39.0	24.8	81.0	1016.4	0.0	0.0	2019-01-23 00:00:00
16	160.0	153.0	17.71	36.7	24.5	99.0	1015.4	0.0	90.8	2019-01-23 14:00:00
34	231.0	202.0	6.09	16.6	24.7	89.0	1012.9	0.0	0.0	2019-01-24 08:00:00
48	285.0	295.0	6.38	20.0	24.2	91.0	1013.6	14.2	22.7	2019-01-24 22:00:00
64	8.0	8.0	9.70	18.9	24.2	100.0	1017.6	0.0	238.7	2019-01-25 14:00:00
139	22.0	7.0	13.02	24.6	24.5	60.0	1016.0	0.0	445.0	2019-01-28 17:00:00
161	119.0	122.0	10.04	21.2	24.8	77.0	1016.3	0.0	334.3	2019-01-29 15:00:00

```
VELR         22.905755
TEMP         22.956835
HR           87.949640
PB         1014.661871
PREC          2.166187
RADSOL      120.388406
dtype: float64
```

Entrada[10]; O calculando específicamente una de las columnas, como Temperatura.

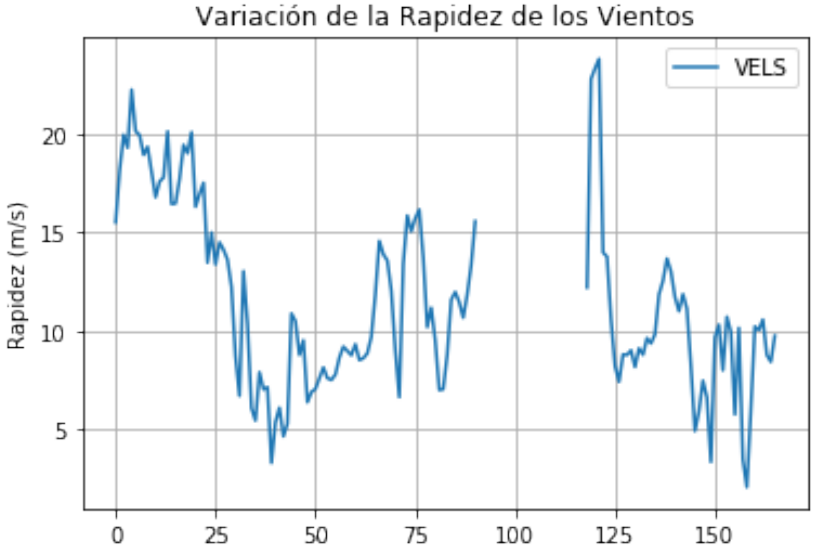
```
df.TEMP.mean()
Out[13]:
22.9568345323741
```

Entrada[10]; Graficando la rapidez de los vientos en $\frac{m}{s}$, obtenemos la Figura 5.

```
plt.figure(); df.VELS.plot(); plt.legend(loc='best')
plt.title("Variación de la Rapidez de los Vientos")
plt.ylabel("Rapidez (m/s)")
plt.grid(True)
plt.show()
```

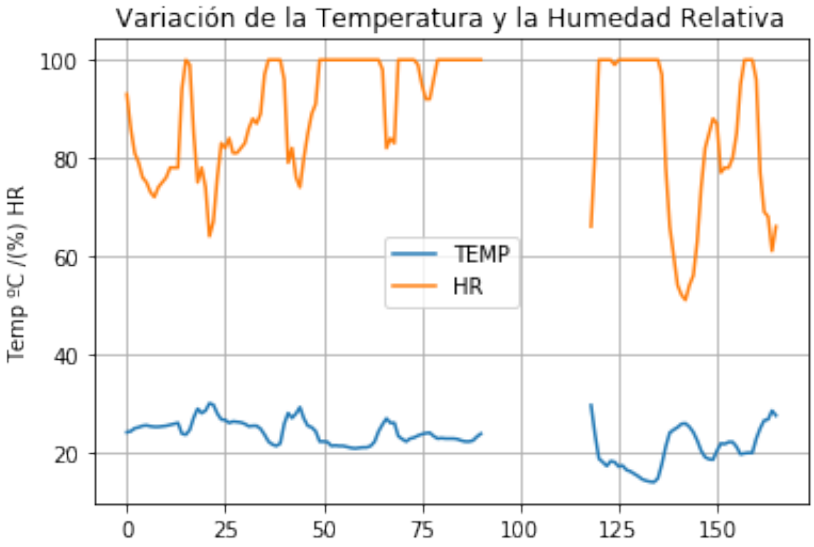
Entrada[11]; Graficando la Temperatura y la Humedad Relativa se obtiene la Figura 6.

Figura 5: Variación de la rapidez de los vientos




```
df1 = df[['TEMP','HR']]
plt.figure(); df1.plot(); plt.legend(loc='best')
plt.title("Variación de la Temperatura y la Humedad Relativa")
plt.ylabel("Temp °C /(%) HR")
plt.grid(True)
plt.show()
```

Figura 6: Temperatura y Humedad Relativa

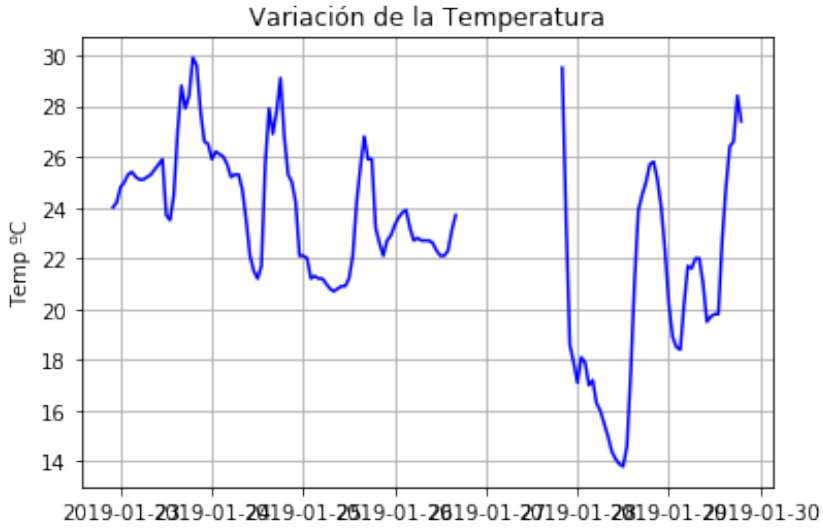


Entrada[12]; Graficando la variación de Temperatura en función del tiempo en la Figura 7.

```
plt.plot_date(x=df.FECHA, y=df.TEMP, fmt="b-")
plt.title("Variación de la Temperatura")
plt.ylabel("Temp °C")
plt.grid(True)
plt.show()
```

Entrada[13]; Creando una gráfica que nos muestra la rapidez de los vientos y de las ráfagas en función del tiempo en la Figura 8. La hora donde se presentó la mayor velocidad del viento fue a las

Figura 7: Temperatura y Humedad Relativa



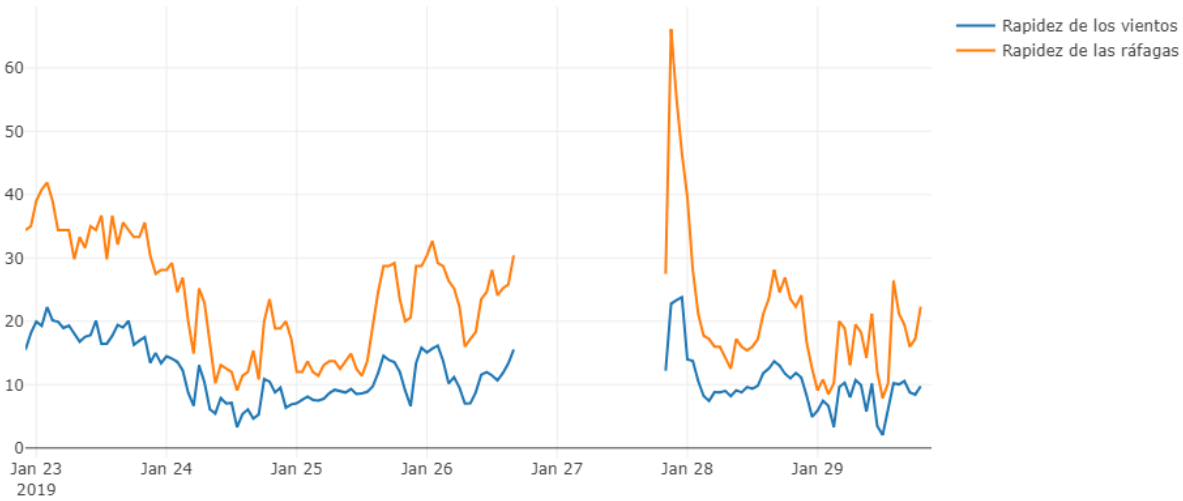
11:00 PM del dia 27, sin embargo la mayor rapidez de las ráfagas fue obtenida a las 9:00 del mismo día.

```
trace1 = go.Scatter(
    x=df.FECHA,
    y=df.VELS,
name='Rapidez de los vientos',

)
trace2 = go.Scatter(
    x=df.FECHA,
    y=df.VELR,
name = 'Rapidez de las ráfagas ',
)

data = [trace1, trace2]

fig = dict(data=data)
py.iplot(fig, filename='simple-connectgaps')
```

Entrada[14]; Para graficar la dirección de los vientos junto con la velocidad de los mismos en función a la fecha en la Figura 9, primero se creó una nueva variable VELS2, la cual representa la velocidad de los vientos multiplicada por 10, con la finalidad de que ambos trazos se ubicaran dentro de los mismos rangos numéricos.
La velocidad de los vientos más alta fue la registrada a las 11:00 el 27 de Enero, cuya dirección se orientaba a 13° del Norte

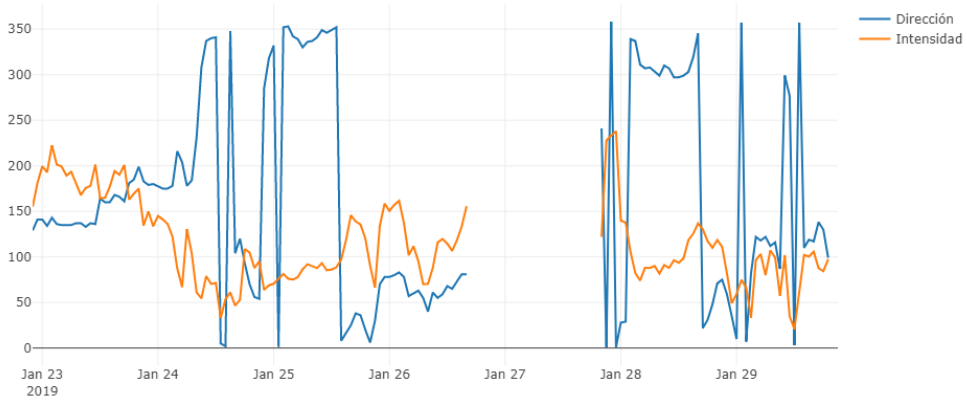
```
df['VELS2']=10*(df.VELS)

trace1 = go.Scatter(
    x=df.FECHA,
    y=df.DIRS,
    name='Dirección',
)
trace2 = go.Scatter(
    x=df.FECHA,
    y=df.VELS2,
```

Figura 8: Rapidez de Vientos y Ráfagas




```
    name = 'Intensidad ',
)
data = [trace1, trace2]
fig = dict(data=data)
py.iplot(fig, filename='simple-connectgaps')
```

Figura 9: Dirección de los vientos

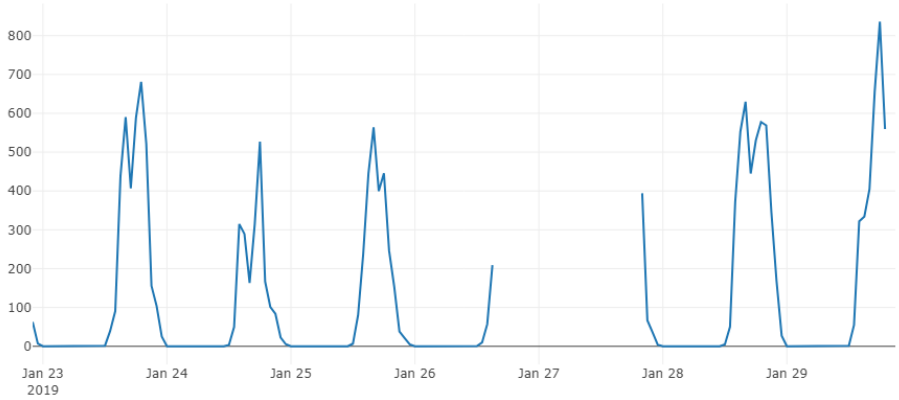


Entrada[15]; Se analiza la radiación solar registrada en la Figura 10. Obviamente la radiación solar es nula en los horarios de la noche, por lo que la gráfica muestra momentos donde la radiación solar es igual a 0.

```
trace = go.Scatter(
x=df.FECHA,
y=df.RADSOL,
name='Radiación Solar',
```

```
)
data = [trace]
fig = dict(data=data)
py.iplot(fig, filename='simple-connectgaps')
```

Figura 10: Radiación Solar



Entrada[16]; El lapso presentado en los datos obtenidos es de 16,09°C, lo cual fue obtenido a partir de restar la temperatura mínima a la temperatura máxima con la entrada:

```
df.TEMP.max()-df.TEMP.min()
```

0.3. Comentarios

Las impresiones al utilizar jupyter notebook son positivas, como lo es en la mayoría de las cosas, al inicio se debe de ajustarse a la sintaxis y a las restricciones que presenta, sin embargo, una vez entendiendo el procedimiento y sabiendo donde buscar información, es fácil de usar, la creación de gráficas interactivas es muy eficiente y sin duda utilizaré el programa de nuevo para la realización de distintos trabajos.
Me pareció similar a Fortran, al declarar y llamar una variable, el hecho de que en Phyton se tiene que importar el programa que deseas utilizar y asignar una manera de cómo será llamado y utilizado en el futuro.
No me pareció que la actividad fuera difícil, es normal en cualquier situación el tener una curva de aprendizaje y sentir que no se está avanzando en nada, pero el saber donde y como buscar información en internet para poder completar la actividad es la manera en que se logra el objetivo.