Reporte de la Actividad 2

Introducción a Python y Jupyter Lab

Luis Fernando Duarte Gonzalí
Universidad de Sonora
Física Computacional

Enero del 2019

1. Introducción

En el siguiente trabajo se muestra una actividad realizada para conocer el entorno de Jupyter Lab y el uso de un lenguaje de programación Python en el cual se muestran los resultados del análisis de datos registrados por el servicio meteorológico nacional en Nogales, Sonora. Los datos registrados datan del 24 al 31 de Enero del presente año.

El análisis de datos se hizo a través de **python**, se realizaron las gráficas las ráfagas, la velocidad y la dirección del viento registrado en esos días, además de la radiación solar y otros parámetros.

2. Estación Meteorológica Automática

Es un conjunto de dispositivos eléctricos y mecánicos que realizan mediciones de las variables meteorológicas de forma automática (sobre todo en forma numérica) (Referencia OMM 182.).

Una Estación Meteorológica Automática, está conformada por un grupo de sensores que registran y transmiten información meteorológica de forma automática de los sitios donde están estratégicamente colocadas. Su función principal es la recopilación y monitoreo de algunas Variables Meteorológicas para generar archivos del promedio de cada 10 minutos de todas las variables, esta información es enviada vía satélite en intervalos de 1 ó 3 horas por estación.

La hora que se utiliza para registrar los datos es el horario TUC ó UTC (Tiempo Universal Coordinado) por esta razón deberá tener en consideración este factor para la correcta interpretación de los datos desplegados en esta página.

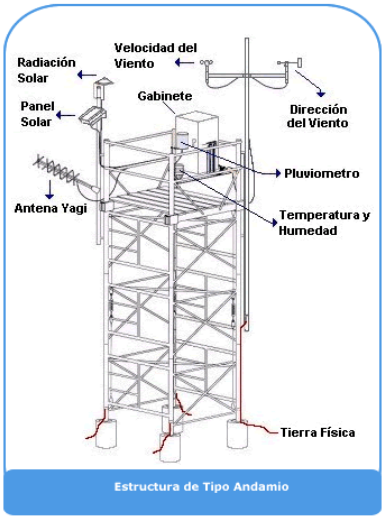
El área representativa de las estaciones es de 5 km de radio aproximadamente, en terreno plano, excepto en terreno montañoso.

2.1. Sensores que integran la estación:

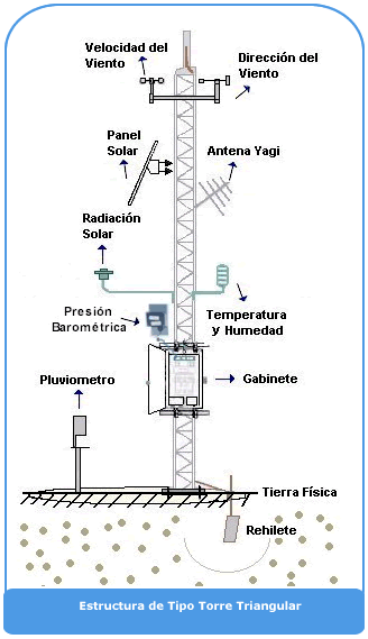
- Velocidad del Viento
- Dirección del Viento
- Presión atmosférica
- Temperatura y Humedad Relativa
- Precipitación
- Radiación Solar

2.2. Tipos de estructuras donde se montan las estaciones:

2.2.1. Tipo Andamio



2.2.2. Tipo Torre Triangular



3. Análisis de Datos

Para comenzar con el análisis fue necesario instalar Python de Anaconda en la versión 3.7. Anaconda incluye Jupyter Lab. Primero creamos una carpeta llamada .ªctividad2"para después, desde Anaconda Prompt, abrir Jupyter Lab escribiendo lo mismo estando en la carpeta de la actividad. Una vez abierta la ventana del navegador seleccionamos la opción de Python 3.

Los datos utilizados se descargaron de la página del Servicio Meteorológico Nacional de las Estaciones Automáticas. Se eligió la región de Nogales con los datos registrados desde el 24 hasta el 31 de Enero.

Para empezar a trabajar en python, primero se importaron las bibliotecas Pandas, Numerical Python y Matplot para generar las gráficas.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Se leyó el archivo de texto, definiéndole una variable, también se saltaron los renglones innecesarios que fueron 4 para después estructurar los datos de mejor manera con *DataFrame*.

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

```
# Dar estructura de datos (DataFrame)
df = pd.DataFrame(df0)
```

A modo de ejemplo se visualizaron los primeros 5 datos obtenidos de la siguiente manera:

```
df.head()
```

	DIRS	DIRR	VELS	VELR	TEMP	HR	PB	PREC	RADSOL	FECHA
0	37.0	352.0	4.06	12.9	10.5	29.0	877.0	0.0	620.7	2019-01-24 17:00:00
1	96.0	123.0	5.48	12.5	13.1	27.0	876.7	0.0	729.2	2019-01-24 18:00:00
2	66.0	338.0	6.00	15.8	15.7	26.0	876.1	0.0	771.2	2019-01-24 19:00:00
3	24.0	58.0	8.79	23.4	17.1	24.0	875.0	0.0	737.0	2019-01-24 20:00:00
4	6.0	354.0	8.89	19.2	17.7	23.0	874.3	0.0	630.7	2019-01-24 21:00:00

4. Resultados

Al momento de obtener los resultados, lo que nos interesaba era la variación en el tiempo de la velocidad, ráfagas y su dirección. También se graficó el comportamiento de la radiación solar como función del tiempo. Se realizó una diferencia entre el mínimo y máximo de la temperatura diaria con los datos que arroja el análisis exploratorio de datos.

4.1. Rapidez del viento, ráfagas y dirección

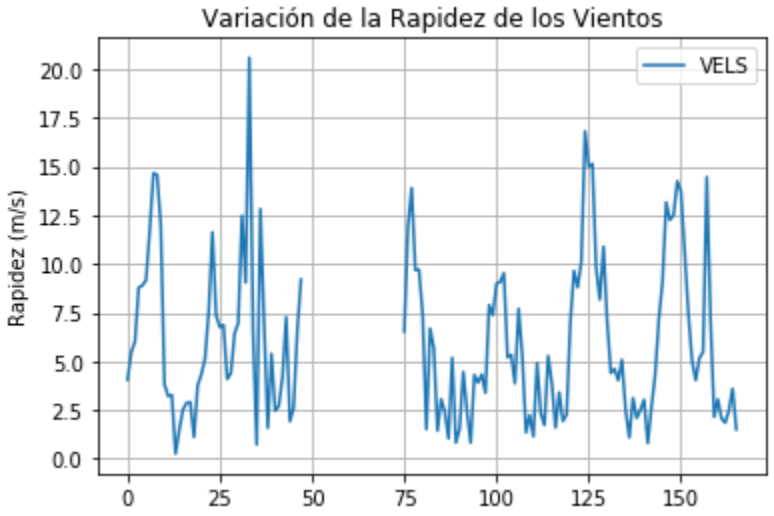
Para obtener la rapidez de los vientos y de las ráfagas se usó el siguiente código:

```
# Gráfica de la rapidez de los vientos (m/s)
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()
```

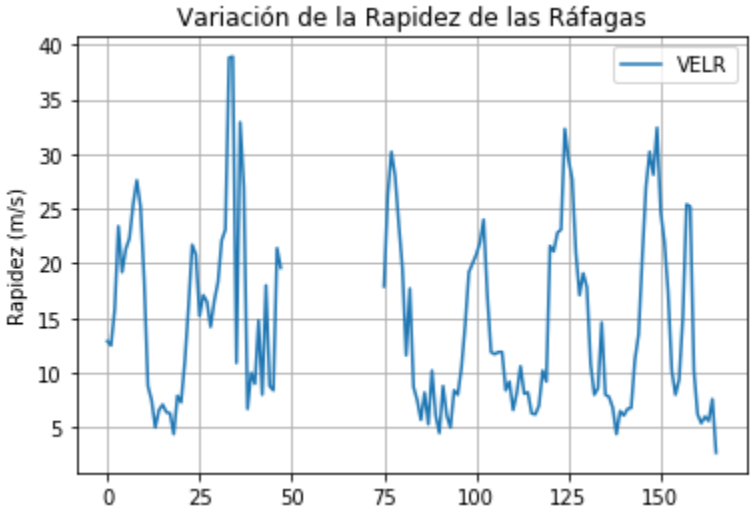
Sólo se cambió "VELS"por "VELR"para poder graficar la rapidez de las ráfagas:

```
# Gráfica de la rapidez de las ráfagas (m/s)
plt.figure(); df.VELR.plot(); plt.legend(loc='best')
plt.title("Variación de la Rapidez de las Ráfagas")
plt.ylabel("Rapidez (m/s)")
plt.grid(True)
plt.show()
```

4.1.1. Gráfica de la Rapidez del Viento



4.1.2. Gráfica de la Rapidez de las Ráfagas



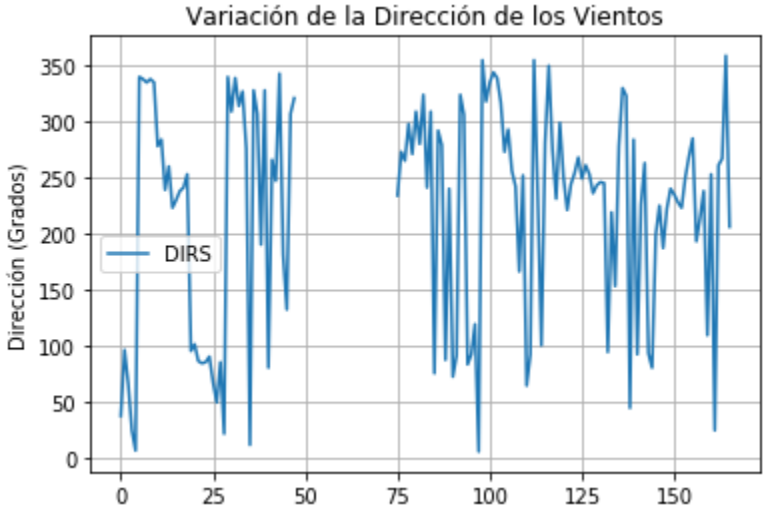
Como podemos observar, las horas del día con más viento van desde las 22:00 hasta 03:00 horas.

4.1.3. Gráfica de la dirección del viento

Se utilizó el mismo formato del código para generar la gráfica de la variación de la dirección del viento con respecto al viento.

```
# Gráfica de la dirección de los vientos (Grados)
plt.figure(); df.DIRS.plot(); plt.legend(loc='best')
plt.title("Variación de la Dirección de los Vientos")
plt.ylabel("Dirección (Grados)")
plt.grid(True)
plt.show()
```

La gráfica que se generó fue la siguiente



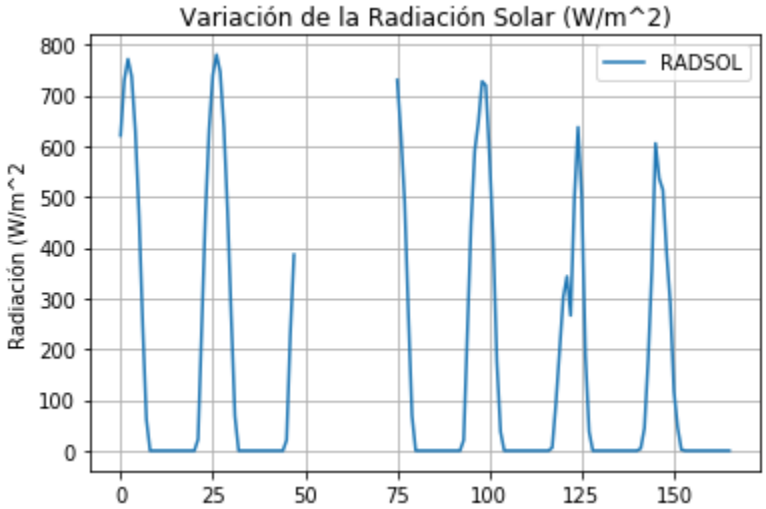
Como podemos observar, los vientos dominantes van en la dirección de entre 200 y 300 grados que es en dirección Oeste o Sur-Oeste.

4.2. Radiación Solar

El código que se utilizó para generar la gráfica de la radiación solar fue el siguiente:

```
# Gráfica de la radiación solar como función del tiempo
plt.figure(); df.RADSOL.plot(); plt.legend(loc='best')
plt.title("Variación de la Radiación Solar (W/m^2)")
plt.ylabel("Radiación (W/m^2")
plt.grid(True)
plt.show()
```

4.2.1. Gráfica de la Radiación Solar como Función del Tiempo.



La gráfica presenta los valores de la radiación solar en diferentes horas del día, como se puede observar hay un cierto patrón que se forma, aproximadamente las horas con mayor cantidad de radiación son al medio día y por obvias razones se encuentra en 0 en horas de la noche.

4.3. Temperatura y Humedad

4.3.1. Diferencia entre temperatura máxima y mínima

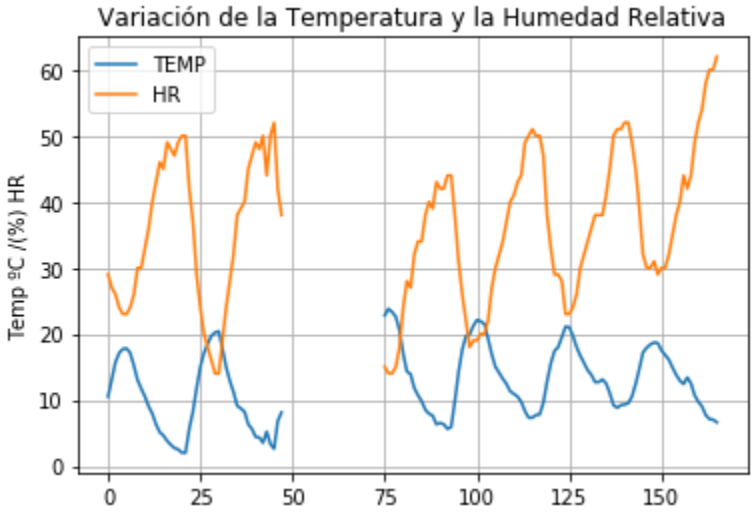
Para calcular la diferencia se utilizó la función *.min* y *.max* para la columna de las temperaturas y se imprimió la diferencia entre las dos cantidades:

```
# Con los datos que nos arrojó al realizar el análisis exploratorio de datos,
# podemos observar las temperaturas máxima y mínima:

Tmax = df.TEMP.max()
Tmin = df.TEMP.min()
print('La diferencia entre la temperatura máxima y mínima es', Tmax - Tmin, '°C')
```

```
La diferencia entre la temperatura máxima y mínima es 21.8 °C
```

4.3.2. Gráfica de la Humedad Relativa y Temperatura



Como se puede observar en la gráfica, hay una relación entre los dos parámetros. Podemos ver que cuando la temperatura sube, la humedad relativa baja y es lo mismo en sentido contrario.

4.4. Análisis Exploratorio de Datos

Para generar el análisis exploratorio sólo fue necesario utilizar la función *describe* sobre nuestro *DataFrame* que nos arrojó una tabla con los siguientes 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	139.0	139.000000
mean	220.482014	255.805755	6.019137	14.748201	12.376978	35.971223	873.312950	0.0	164.618705
std	97.760603	89.283694	4.118416	8.226701	5.505548	11.485304	1.974322	0.0	251.052767
min	5.000000	20.000000	0.260000	2.700000	2.000000	14.000000	869.900000	0.0	0.000000
25%	125.500000	224.500000	2.625000	8.000000	8.000000	27.500000	871.900000	0.0	0.000000
50%	245.000000	275.000000	5.150000	11.900000	12.400000	37.000000	873.100000	0.0	0.000000
75%	292.500000	329.000000	8.795000	21.050000	17.100000	45.000000	874.900000	0.0	280.900000
max	359.000000	360.000000	20.590000	38.900000	23.800000	62.000000	877.400000	0.0	780.000000

5. Jupyter Lab y Biblioteca Pandas de Python

5.1. Jupyter

Como fue mencionado con anterioridad, Jupyter Notebook es un ambiente de programación que permite editar código, widgets, texto, ecuaciones, imagenes, video, los cuales se pueden convertir a diferentes formatos y compartir facilmente.

5.1.1. Características

Jupyter Notebook contiene tres componentes principales:

- Aplicación web: su aplicación interactiva, donde se puede editar y crear código y documentos "notebook". Esta edita y corre código, permite crear widgets de JavaScript e incluye ecuaciones matemática.
- Kernels: son procesos separados que corren el código del usuario, en cierto lenguaje, en la aplicación web y muestran la salida directamente en la aplicación. Cuenta con kernels que soportan: Phyton, R, Julia, Ruby, Haskell, Scala, node.js, Go.
- Documentos "notebook": documentos que contienen una representación de lo que se ve en la aplicación web, todas las entradas y salidas, imagenes, ecuaciones, etc. Cada notebook tiene su propio kernel. Los notebooks consisten en una sucesión de celdas y tienen la extensión ".ipynb", pero pueden exportarse en diferentes formatos como PDF o LaTeX.

5.1.2. Ventajas

- Se puede usar en prácticamente cualquier lugar, es un software libre disponible en diferentes sistemas operativos y fácil de instalar.
- Soporta una amplia variedad de lenguajes de programación, entre los que destaca Phyton, R, Ruby, entre otros.
- Es interactivo, te brinda inmediatamente el "feedback.º salida del código.
- Es open source, permite a los usuarios revisarlo y proponer cambios o extensiones, incluso es posible personalizarlo para un uso específico.
- Produce gráficas, tablas entre otras capacidades matemáticas, es muy utilizado en la ciencia.

5.1.3. Desventajas

- Aunque es útil que sea interactivo, es un poco difícil acostumbrarse a esto.
- Es más complicado instalarlo y acomodarlo todo para un primer uso, que otros entornos de programación.
- No es fácil configurarlo para poder compartirlo y editar en otra computadora, es necesario realizar ciertos pasos que no podrían resultar triviales para cualquier usuario.
- Es difícil poder trabajar de manera colaborativa con este entorno, por lo mismo que no es sencillo .ªbrir.el código en otra computadora, como se revisó en el punto anterior.

5.2. Phyton: Pandas

Pandas es un paquete de Phyton que facilita el trabajar con datos.

Provee estructuras de datos de manera rápida y flexible y promete ser el analista y manipulador de datos, en open source, más poderoso, disponible en cualquier lenguaje.

Trabaja muy bien en el manejo de varios tipos de datos, ya sean tabulares, datos observacionales o estadísticos, entre otros.

5.2.1. Ventajas

- Reporta cuando se tienen datos perdidos.
- Permite el cambio de tamaño de los datos, se pueden intertar o eliminar columnas del data frame.
- Buenas herramientas IO que permiten cargar datos desde archivos planos, bases de datos, archivos excel y guardarlos en un formato HDf5 ultra rápido.
- Pandas es rápido.
- Se usa en la producción de aplicaciones financieras.
- Permite un manejo de bases de datos bastante extensas.
- Es fácil de usar y fácil de aprender.

5.2.2. Desventajas

- Cuenta con mucha competencia, como es excel, con el que es bastante comparado, ya que también permite el uso de bases de datos, de una manera un poco más sencilla para un usuario no experimentado en el uso de código.

6. Bibliografía

- Servicio Meteorológico Nacional. (2019). Consultado: 31 de Enero del 2019, de Conagua. Sitio web: http://smn1.conagua.gob.mx/emas/
- Python Tutorial (2019). Consultado: 31 de Enero del 2019, de tutorialspoint. Sitio web: https://www.tutorialspoint.com/python/index.htm
- Pros and Cons of Python Jupyter vs Normal Python. (2017). Consultado: 1 de Febrero del 2019, de Quora. Sitio web: https://www.quora.com/What-are-the-pros-and-cons-of-using-Python-Jupyter-versus-a-normal-Python-development-environment
- Biblioteca Pandas. (2019). Consultado: 1 de Febrero del 2019, de Pandas Pydata. Sitio web: https://pandas.pydata.org/pandas-docs/stable/index.html

7. Apéndice

1. ¿Cuál es tu primera impresión de Jupyter Notebook?

   Es muy práctico porque se trabaja todo en una sola carpeta y el código cada cierto tiempo se guarda de manera automática. El manejo de archivos dentro de la misma interfaz es muy cómodo y se pueden abrir desde el mismo JupyterLab.

2. ¿Se te dificultó leer código en Python?

   No, de hecho el código que maneja Python es demasiado intuitivo así como las funciones muestran de manera clara su objetivo, es muy práctico y sencillo de aprender. Además no es necesario compilar el código, sólo correrlo.

3. ¿En base a tu experiencia de programación en Fortran, que te parece el entorno de trabajar en Python?

   Es mucho más cómo y sencillo, empezando con que no es necesario declarar variables como cierto número entero o de otro tipo. Es más fácil trabajar en el entorno de python que el de Fortran.

4. A diferencia de Fortran, ahora se producen las gráficas utilizando la biblioteca Matplotlib. ¿Cómo fue tu experiencia?.

   Primero, es diferente estéticamente a gnuplot de fortran, es muy agradable a la vista y son más fáciles de producir.

5. En general, ¿qué te pareció el entorno de trabajo en Python?

   Me pareció muy cómodo, más que nada, en definitiva preferiría seguir programando en Python y dejar otros lenguajes de lado por el momento. Me parece intuitivo, fácil de manejar y de aprender.

6. ¿Qué opinas de la actividad? ¿Estuvo compleja? ¿Mucho material nuevo? ¿Que le faltó o que le sobró? ¿Qué modificarías para mejorar?

   Sólo fue un poco laboriosa al momento de escribir el reporte en LaTeX, pero sólo es por falta de práctica. En general, la actividad es sencilla, sí hay material nuevo porque no conocía bien el entorno de python y menos el de JupyterLab.

7. ¿Comentarios adicionales que desees compartir?

   Ya había entrado a programar en python desde el semestre pasado para la materia de .ªnálisis Numérico"donde se me pidió calcular numéricamente las constantes elásticas y otras cosas en el problema de oscilador armónico. Me gustaría seguir aprendiendo python en otras áreas para encontrar más aplicaciones, en especial la inteligencia artificial junto con el análisis de datos.