"El saber de mis hijos hará mi grandeza"

Universidad de Sonora
Departamento de Física
Licenciatura en Física
Física Computacional-1
2016-2

Graficando con Python/Pandas

Danira Rios Quijada
Profesor: Carlos Lizárraga Celaya

16 de septiembre de 2016

Resumen

En el siguiente informe de trabajo, se describen los pasos que seguimos en la actividad número 3 del curso de Física computacional, la cual consistió en empezar a trabajar con el lenguaje de programación Python y con su biblioteca Pandas.

1. Introducción

Python es un lenguaje de programacion interpretado y dinámico, esta diseñado bajo una "filosofía"que enfatiza la legibilidad del código y su sintaxis permite al programdor expresar conceptos en el menor número de lineas de código posible, menor por ejemplo que en C++ o Java.
Python soporta multiples paradigmas de programación, incluyendo; orientación de objetos, programación imperativa y, en menor medida, programación funcional.
Posee una licencia de código abierto, denominada Python Software Foundation License,que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1.
Python posee una librería llamada Pandas, destinada al análisis de datos, que proporciona estructuras de datos flexibles y que permiten trabajar con ellos de forma muy eficiente.En esta actividad usamos la librería Pandas en Python para empezar con el análisis de los datos de actividades anteriores.[3]

2. Preparando de nuevo los datos

Cuando empezamos a trabajar con los datos en Python, tuvimos algunos problemas con el archivo de datos atmosféricos descargados en actividades anteriores, dado que el número de datos recaudados cada día, a veces era mayor o menor, y esta discrepancia en numero de columnas hacía que el archivo no se pudiera cargar, por lo tanto optamos por filtrar de nuevo el archivo con el script que habiamos utilizado en actividades anteriores:

```
# Script para filtrar renglones de un archivo que
contengan las cadenas de caracteres dados

#!/bin/bash

egrep -v  'PRES|hPa'  datos2015.txt | egrep '72440|CAPE|CINS' > CAPE.csv
```

Como se puede ver en el script, eliminamos todas las columnas y sólo nos quedamos con la columna de la fecha,estación,CAPE y CINS, ya que eran las únicas de interes para el análisis de datos, el archivo al que se mandó el contenido de estas columnas se nombro CAPE.csv (archivo separado por comas).
Una vez que ya contabamos con este archivo, usamos el filtro grep, desde la terminal, para extraer solamente la data de las 12Z (medidiano de Greenwich):

```
grep 12Z CAPE.csv > cape.csv
```

Una vez que ya extrajimos sólo la información de este horario, ya no era necesario tener en el archivo la hora de la medición, por lo tanto, desde emacs eliminamos esa parte del registro utilizando query-replace. Una vez que el archivo estuvo listo, empezamos a trabajar con Python.

3. Empezando a utilizar Python

Primeramente desde una terminal abrimos Python notebook, abriendose una ventana del navegador, desde la cual trabajaríamos con nuestros datos.
Creamos un nuevo archivo Python, desde el cual trabajariamos y en una línea de inserción, escribimos lo siguiente:

```
import pandas as pd
import numpy as np
import matplotlib as plt
df=pd_read.csv (ruta de mi archivo)
```

En donde podemos ver que importamos la librería Pandas, asi como Numpy y Matplotlib, también cargamos nuestro archivo previamente trabajado y lo denominamos (df.)
Ahora probamos una de las funciones de Pandas:

```
df.head(20)
```

Con lo cual podríamos ver las primeras 20 filas del archivo, empezando a contar desde cero, el resultado fue:

	Fecha	estacion	CAPE	CINS
0	01 Jan 2015	72440	0	0
1	02 Jan 2015	72440	0	0
2	03 Jan 2015	72440	0	0
3	04 Jan 2015	72440	0	0
4	05 Jan 2015	72440	0	0
5	06 Jan 2015	72440	0	0
6	07 Jan 2015	72440	0	0
7	08 Jan 2015	72440	0	0
8	09 Jan 2015	72440	0	0
9	10 Jan 2015	72440	0	0
10	11 Jan 2015	72440	0	0
11	12 Jan 2015	72440	0	0
12	13 Jan 2015	72440	0	0
13	14 Jan 2015	72440	0	0
14	15 Jan 2015	72440	0	0
15	16 Jan 2015	72440	0	0
16	17 Jan 2015	72440	0	0
17	18 Jan 2015	72440	0	0
18	19 Jan 2015	72440	0	0
19	20 Jan 2015	72440	0	0

Después utilizamos la función "describe",

```
df.describe()
```

con la cual pudimos ver varios datos estadísticos de la información que teniamos en nuestro archivo:

	estacion	CAPE	CINS
count	363	363.000000	363.000000
mean	72440	233.539780	-34.298678
std	0	531.053259	70.156343
min	72440	0.000000	-373.210000
25%	72440	0.000000	-32.645000
50%	72440	0.000000	0.000000
75%	72440	112.020000	0.000000
max	72440	3374.470000	0.000000

Con el siguiente paso tuvimos algunos problemas, ya que los encabezados de las columnas tenían un formato u', es decir, que su formato permitía incluir acentos o caractéres alternativos, sin embargo no lo sabiamos y por lo tanto, a la hora de querer graficar e incluir el nombre de cierta columna en el plot, Python no la identificaba. Entonces recurrimos a:

```
In [40]: print df.columns

         Index([u'Fecha', u' estacion', u' CAPE', u' CINS'], dtype='object')
```

con lo que nos dimos cuenta que a la hora de hacer los plots teniamos que usar el formato de:

```
[u'------]
```

en el nombre de las columnas.
Aun que no fue nuestro caso, a veces se cuenta con filas donde existen datos nulos, por lo tanto hay que eliminar esas filas, ya que no se puede trabajar con el resto de la información si estan estas interviniendo, en dado caso se aplica lo siguiente:

```
df_complete=df.dropna()
```

En donde nombramos una nueva base, donde literalmente se desechan las filas con datos na, o nulos.
Dado que trabajamos con datos de un año, en mi caso con datos de Tucson, AZ, solo para saber si teníamos datos de todo el año, usamos la función (len(df)), con la cual me di cuenta que contaba con 363 filas, osea que ese año faltaron 2 días de datos.
Ahora, para que los plots aperecieran directamente en la ventana del navegador, utilizamos:

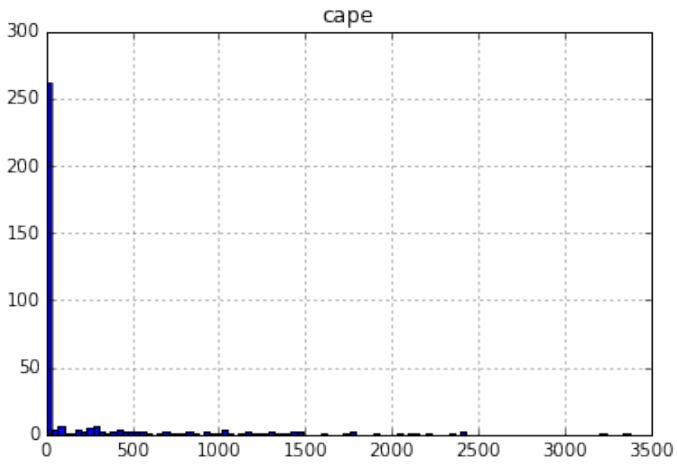
```
%mathplotlib inline
```

3.1. Comenzamos con los gráficos.

- Para hacer un histograma de la columna (cape) (con cien particiones), escribimos lo siguinete:

  ```
  df.hist(u'cape',bins=100)
  ```

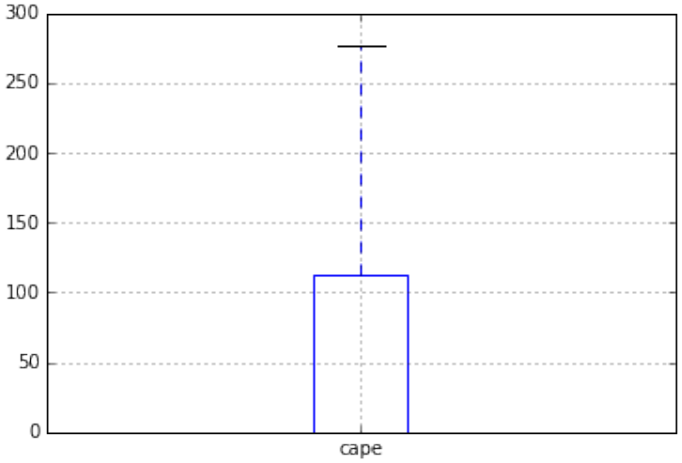
  y obtenemos:



- Para hacer una gráfica de caja, en la cual se representan gráficamente los cuantiles estadísticos de la información, escribimos lo siguiente:

  ```
  df.boxplot(column=u'cape')
  ```

  y obtenemos:



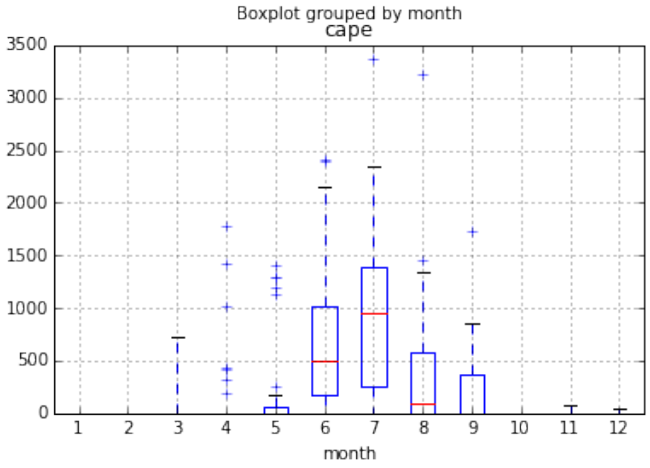
- El objetivo ahora era crear un box-plot, de cada mes por separado, para esto, incluimos una nueva columna en el archivo cargado, la cual nos diría de que mes se trata, para esto escribimos:

  ```
  df['month']=pd.DatetimeIndex(df[u'fecha']).month
  ```

  Lo que esta haciendo esta línea de código, es creando una columna llamada (month),tomando de la columna (fecha), con formato de (fecha/hora), la parte del mes, por ejemplo de (january) tomaría un 1, de (august) tomaría un 8, y asi creamos una columna exclusivamente con el mes. Para crear el boxplot con esta infarmación, escribimos lo siguiente:

  ```
  df.boxplot(column=u'cape',by=u'month')
  ```

  con lo cual obtuvimos:



4. Bibliografía

Referencias

[1] Carlos Lizárraga Celaya, *Actividad dos, curso de computacional 1*, (2016, 25 de agosto).Recuperado (2016, 25 de agosto), Desde: `http://computacional1.pbworks.com/w/page/110289883/Actividad20220282016-229`

[2] University of Wyoming; Department of Atmospheric Science *Datos, sondeo atmosférico*, (2015), recuperado (2016, 25 de agosto). Desde: `http://weather.uwyo.edu/upperair/sounding.html`

[3] Wikipedia de free encyclopedia, *Python (programming language)*, (2016, 16 de septiembre). Recuperado (2016,16 de septiembre), Desde: `https://en.wikipedia.org/wiki/Python_(programming_language)`