computacional4

juanjo$_l$*opez*1997

December 2018

computacional1, Reporte 4 Juan Jose Lopez December 2018

1 Introduction

Los datos utilizados en esta practica son propiedad de *NOAA: National Oceanic and Atmospheric Administration*, y en esta practica los vamos a tabular, para despues aplicarle la transformada de Fourier.

2 Trasformada Rapida de Fourier, intro

La transformada rápida de Fourier (Fast Fourier Transform, FFT) es uno de los algoritmos más importantes en el procesamiento de señales y el análisis de datos. Desarrollado por por J. W. Cooley y John Tukey en 1960, es el algoritmo más utilizado para lograr una Transformada de Fourier en la práctica.

Las características que pueden estar ocultas o invisibles en el dominio del tiempo pueden ser más fáciles de evaluar en el dominio de la frecuencia. Convertir datos del dominio del tiempo en el dominio de la frecuencia también es una técnica exploratoria comúnmente utilizada para revelar patrones repetitivos.

Una de las aplicaciones más prometedoras de una FFT para el monitoreo de condición de equipos es el análisis de la vibración de acelerómetros colocados en maquinaria rotativa. Si la maquinaria giratoria contiene rodamientos de bolas, las bolas dentro de los rodamientos pasarán el acelerómetro a una frecuencia específica (dependiendo del número de bolas y la geometría del rodamiento), que aparecerá como un pico en el espectro de frecuencias. La magnitud del pico se usa a menudo para diagnosticar fallas dentro de los rodamientos, con magnitudes altas que indican una falla inminente.

3 Etapas de desarroyo

Como se me volvio algo usual a lo largo del curso, comence por nombrar las librerias que pense podria ocupar, aunque a lo largo del documento fui agregando

las que fui requiriendo

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
```

datos

Comence por crear un *Dataframe* donde leeria los datos del archivo de *NOAA* que recolecto durante todo el año del 2018 al 2019. Le tuve que hacer algunas modificaciones para que las columnas coincidieran con lo que tenia planeado. Pro ejemplo: a la columna fecha le faltaba la hora para poder cambiar el formato de columna de ese renglon a *datetime64*, algo que me es muy conveniente a la hora de seleccionar informacion con *pandas*. La ultima instruccion que le pido que haga es que lo muestre en pantalla el *Dataframe* para verificar los cabios que realice

```
In [2]: #todo el año
        Ens = pd.read_csv('ensenada.csv', header=None,skiprows= 1 )
        Ens.columns = ['Fecha', 'Dia', 'Hora', 'Prediccion ft', 'Prediccion m', 'High/Low']
        Ens["Fecha"] = Ens["Fecha"].map(str) +" "+ Ens["Hora"]
        Ens["Prediccion m"]= Ens["Prediccion m"] /100
        Ens['Fecha'] = pd.to_datetime(Ens['Fecha'])
        Ens.head()
```

Out[2]:

	Fecha	Dia	Hora	Prediccion ft	Prediccion m	High/Low
0	2018-01-01 01:24:00	Mon	01:24	1.32	0.40	L
1	2018-01-01 07:33:00	Mon	07:33	6.89	2.10	H
2	2018-01-01 14:47:00	Mon	14:47	-1.67	-0.51	L
3	2018-01-01 21:04:00	Mon	21:04	4.16	1.27	H
4	2018-02-01 02:12:00	Tue	02:12	1.34	0.41	L

Aqui le pido que me muestre el formato de las columnas para ver si puedo trabajar con ellas, aunque en este caso, solo me importaba que la columna fecha estubiera en formato *datetime64*

```
In [3]: Ens.dtypes

Out[3]: Fecha            datetime64[ns]
        Dia                      object
        Hora                     object
        Prediccion ft           float64
        Prediccion m            float64
        High/Low                 object
        dtype: object
```

En este renglon creo otro *Dataframe* y le pido que lea otro archivo que contiene la informacion de seis meses de datos. Vuelve a hacerle algunas modificaciones a los datos y como toque final le pido que me lo muestre en pantalla para asegurarme de que hayan surtido efecto.

```
In [27]: #seis meses
Ens4 = pd.read_csv('ens.csv', header=None,skiprows= 8, dtype= object)
Ens4.columns = ['0', '1', '2', '3', '4', '5','6','7','8','9','10','11']
Ens4['0']= Ens4['0']+'-'+Ens4['1']+Ens4['2']+'-'+Ens4['3']+Ens4['4']+' '+Ens4['5']+Ens4['6']+':'+Ens4['7']+Ens4['8']\
           +':'+ Ens4['9']+Ens4['10']
Ens4['0'] = pd.to_datetime(Ens4['0'])
Ens4['11'] = Ens4['11'].astype(float)
Ens4.head()
```

Out[27]:

	0	1	2	3	4	5	6	7	8	9	10	11
0	2018-04-21 09:00:00	0	4	2	1	0	9	0	0	0	0	1.373
1	2018-04-21 10:00:00	0	4	2	1	1	0	0	0	0	0	1.178
2	2018-04-21 11:00:00	0	4	2	1	1	1	0	0	0	0	0.900
3	2018-04-21 12:00:00	0	4	2	1	1	2	0	0	0	0	0.586
4	2018-04-21 13:00:00	0	4	2	1	1	3	0	0	0	0	0.289

Creo un nuevo *Dataframe* y utilizando el formato *datetime64* de la columna 0, selecciono la informacion de un dia que le voy a asignar a este.

```
In [5]: #un dia
        #13 de nov
        Ens4Dia = Ens4[(Ens4['0'] < '21/11/2018 23:00') &  (Ens4['0'] > '21/11/2018 00:00')]

In [6]: print(Ens4Dia)

                              0  1  2  3  4  5  6  7  8  9 10     11
5128 2018-11-21 01:00:00  1  1  2  1  0  1  0  0  0  0  0.871
5129 2018-11-21 02:00:00  1  1  2  1  0  2  0  0  0  0  1.110
5130 2018-11-21 03:00:00  1  1  2  1  0  3  0  0  0  0  1.232
5131 2018-11-21 04:00:00  1  1  2  1  0  4  0  0  0  0  1.209
5132 2018-11-21 05:00:00  1  1  2  1  0  5  0  0  0  0  1.053
5133 2018-11-21 06:00:00  1  1  2  1  0  6  0  0  0  0  0.815
5134 2018-11-21 07:00:00  1  1  2  1  0  7  0  0  0  0  0.567
5135 2018-11-21 08:00:00  1  1  2  1  0  8  0  0  0  0  0.388
5136 2018-11-21 09:00:00  1  1  2  1  0  9  0  0  0  0  0.338
5137 2018-11-21 10:00:00  1  1  2  1  1  0  0  0  0  0  0.445
5138 2018-11-21 11:00:00  1  1  2  1  1  1  0  0  0  0  0.694
5139 2018-11-21 12:00:00  1  1  2  1  1  2  0  0  0  0  1.032
5140 2018-11-21 13:00:00  1  1  2  1  1  3  0  0  0  0  1.377
5141 2018-11-21 14:00:00  1  1  2  1  1  4  0  0  0  0  1.642
5142 2018-11-21 15:00:00  1  1  2  1  1  5  0  0  0  0  1.755
5143 2018-11-21 16:00:00  1  1  2  1  1  6  0  0  0  0  1.681
5144 2018-11-21 17:00:00  1  1  2  1  1  7  0  0  0  0  1.428
5145 2018-11-21 18:00:00  1  1  2  1  1  8  0  0  0  0  1.049
5146 2018-11-21 19:00:00  1  1  2  1  1  9  0  0  0  0  0.625
5147 2018-11-21 20:00:00  1  1  2  1  2  0  0  0  0  0  0.249
5148 2018-11-21 21:00:00  1  1  2  1  2  1  0  0  0  0  0.002
5149 2018-11-21 22:00:00  1  1  2  1  2  2  0  0  0  0 -0.069
```

```
In [8]: Ens4.dtypes

Out[8]: 0     datetime64[ns]
        1             object
        2             object
        3             object
        4             object
        5             object
        6             object
        7             object
        8             object
        9             object
        10            object
        11           float64
        dtype: object
```

Uso *print(dataframe)* para ver con mayor extension mi *dataframe*

```
In [9]: print(Ens)

                  Fecha  Dia   Hora  Prediccion ft  Prediccion m High/Low
0   2018-01-01 01:24:00  Mon  01:24           1.32          0.40        L
1   2018-01-01 07:33:00  Mon  07:33           6.89          2.10        H
2   2018-01-01 14:47:00  Mon  14:47          -1.67         -0.51        L
3   2018-01-01 21:04:00  Mon  21:04           4.16          1.27        H
4   2018-02-01 02:12:00  Tue  02:12           1.34          0.41        L
5   2018-02-01 08:18:00  Tue  08:18           6.93          2.11        H
6   2018-02-01 15:33:00  Tue  15:33          -1.74         -0.53        L
7   2018-02-01 21:52:00  Tue  21:52           4.22          1.29        H
8   2018-03-01 03:01:00  Wed  03:01           1.40          0.43        L
9   2018-03-01 09:05:00  Wed  09:05           6.73          2.05        H
10  2018-03-01 16:19:00  Wed  16:19          -1.58         -0.48        L
11  2018-03-01 22:41:00  Wed  22:41           4.23          1.29        H
12  2018-04-01 03:53:00  Thu  03:53           1.52          0.46        L
13  2018-04-01 09:53:00  Thu  09:53           6.30          1.92        H
14  2018-04-01 17:06:00  Thu  17:06          -1.23         -0.37        L
15  2018-04-01 23:32:00  Thu  23:32           4.22          1.29        H
16  2018-05-01 04:50:00  Fri  04:50           1.68          0.51        L
17  2018-05-01 10:44:00  Fri  10:44           5.67          1.73        H
18  2018-05-01 17:54:00  Fri  17:54          -0.74         -0.23        L
19  2018-06-01 00:27:00  Sat  00:27           4.24          1.29        H
20  2018-06-01 05:56:00  Sat  05:56           1.85          0.56        L
21  2018-06-01 11:40:00  Sat  11:40           4.91          1.50        H
22  2018-06-01 18:44:00  Sat  18:44          -0.18         -0.05        L
23  2018-07-01 01:26:00  Sun  01:26           4.29          1.31        H
24  2018-07-01 07:16:00  Sun  07:16           1.92          0.59        L
25  2018-07-01 12:46:00  Sun  12:46           4.13          1.26        H
26  2018-07-01 19:38:00  Sun  19:38           0.39          0.12        L
27  2018-08-01 02:28:00  Mon  02:28           4.42          1.35        H
28  2018-08-01 08:49:00  Mon  08:49           1.77          0.54        L
29  2018-08-01 14:08:00  Mon  14:08           3.48          1.06        H
...                 ...  ...    ...            ...           ...      ...
```

Final mente para conclui esta seccion, creo un *Dataframe* donde almaceno la informacion de un dia y le pido al programa que me lo muestre para verificar que salga bien.

```
In [10]: #datos de un dia
         Ensday= Ens[(Ens['Fecha'] <= '2018-12-31 23:01:00') &  (Ens['Fecha'] >= '2018-12-30 22:09:00')]
         Ensday.head()
```

Out[10]:

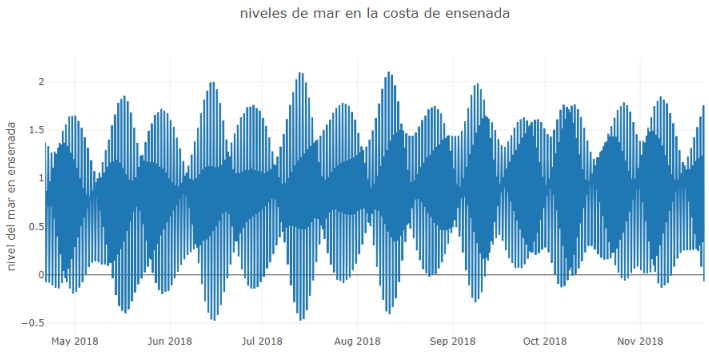
	Fecha	Dia	Hora	Prediccion ft	Prediccion m	High/Low
1406	2018-12-30 22:09:00	Sun	22:09	1.05	0.32	L
1407	2018-12-31 04:47:00	Mon	04:47	5.28	1.61	H
1408	2018-12-31 11:51:00	Mon	11:51	0.51	0.16	L
1409	2018-12-31 17:37:00	Mon	17:37	3.39	1.03	H
1410	2018-12-31 23:01:00	Mon	23:01	1.30	0.40	L

```
In [11]: print(Ensday)

                    Fecha  Dia   Hora  Prediccion ft  Prediccion m High/Low
1406 2018-12-30 22:09:00  Sun  22:09           1.05          0.32        L
1407 2018-12-31 04:47:00  Mon  04:47           5.28          1.61        H
1408 2018-12-31 11:51:00  Mon  11:51           0.51          0.16        L
1409 2018-12-31 17:37:00  Mon  17:37           3.39          1.03        H
1410 2018-12-31 23:01:00  Mon  23:01           1.30          0.40        L
```

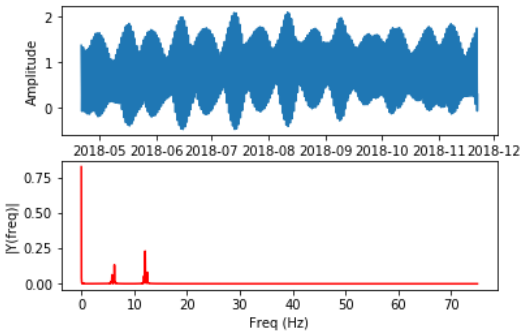
graficas

En esta seccion se muestra el codigo utilizado para realizar las graficas seguido por la salida que arroja dicho renglon al ejecutarlo.

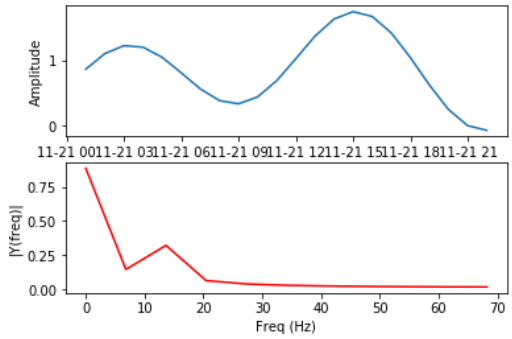
```
In [14]: # año offline grafica: nuevos  datos
import plotly
import plotly.graph_objs as go

import plotly.offline as offline
import plotly.graph_objs as go

plotly.offline.init_notebook_mode(connected=True)
trace1 = go.Scatter(
    x=Ens4['0'],
    y=Ens4['11'],
    name='ensenada',
)

data = [trace1]
layout = go.Layout(
    title=' niveles de mar en la costa de ensenada',
    yaxis=dict(
        title='nivel del mar en ensenada'
    ),
    yaxis2=dict(
        title='nivel del mar en el canal',
        titlefont=dict(
            color='rgb(148, 103, 189)'
        ),
        tickfont=dict(
            color='rgb(148, 103, 189)'
        ),
        overlaying='y',
        side='right'
    )
)

fig = go.Figure(data=data, layout=layout)
plot_url = offline.iplot(fig, filename='multiple-axes-double')
```



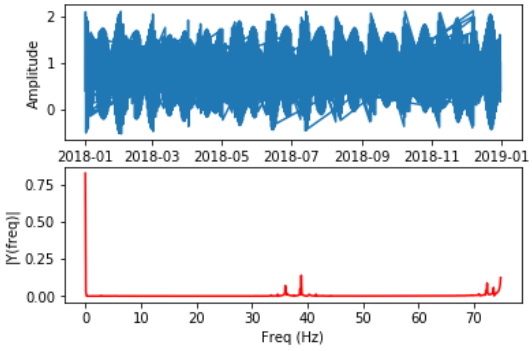

```
In [15]: #6 meses
import matplotlib.pyplot as plt
import plotly.plotly as py
import numpy as np
import plotly
import plotly.graph_objs as go
import plotly.offline as offline
plotly.offline.init_notebook_mode(connected=True)


# Learn about API authentication here: https://plot.ly/python/getting-started
# Find your api_key here: https://plot.ly/settings/api

Fs = 150.0;  # sampling rate
Ts = 1.0/Fs; # sampling interval
t = np.array(Ens4['0'])# time vector


ff = 5;   # frequency of the signal
y = Ens4['11']
n = len(y) # length of the signal
k = np.arange(n)
T = n/Fs
frq = k/T # two sides frequency range
frq = frq[range(n//2)] # one side frequency range

Y = np.fft.fft(y)/n # fft computing and normalization
Y = Y[range(n//2)]

fig, ax = plt.subplots(2, 1)
ax[0].plot(t,y)
ax[0].set_xlabel('Time')
ax[0].set_ylabel('Amplitude')
ax[1].plot(frq,abs(Y),'r') # plotting the spectrum
ax[1].set_xlabel('Freq (Hz)')
ax[1].set_ylabel('|Y(freq)|')

plot_url = offline.iplot(fig, filename='mpl-basic-fft')
```



In [16]:
```
# 1 dia
import matplotlib.pyplot as plt
import plotly.plotly as py
import numpy as np
import plotly
import plotly.graph_objs as go
import plotly.offline as offline
plotly.offline.init_notebook_mode(connected=True)


# Learn about API authentication here: https://plot.ly/python/getting-started
# Find your api_key here: https://plot.ly/settings/api

Fs = 150.0;  # sampling rate
Ts = 1.0/Fs; # sampling interval
t = np.array(Ens4Dia['0'])# time vector


ff = 5;   # frequency of the signal
y = Ens4Dia['11']
n = len(y) # length of the signal
k = np.arange(n)
T = n/Fs
frq = k/T # two sides frequency range
frq = frq[range(n//2)] # one side frequency range

Y = np.fft.fft(y)/n # fft computing and normalization
Y = Y[range(n//2)]

fig, ax = plt.subplots(2, 1)
ax[0].plot(t,y)
ax[0].set_xlabel('Time')
ax[0].set_ylabel('Amplitude')
ax[1].plot(frq,abs(Y),'r') # plotting the spectrum
ax[1].set_xlabel('Freq (Hz)')
ax[1].set_ylabel('|Y(freq)|')

plot_url = offline.iplot(fig, filename='mpl-basic-fft')
```



In [21]:
```
#del 2018 al 2019
import matplotlib.pyplot as plt
import plotly.plotly as py
import numpy as np
import plotly
import plotly.graph_objs as go
import plotly.offline as offline
plotly.offline.init_notebook_mode(connected=True)

# Learn about API authentication here: https://plot.ly/python/getting-started
# Find your api_key here: https://plot.ly/settings/api

Fs = 150.0;  # sampling rate
Ts = 1.0/Fs; # sampling interval
t = np.array(Ens['Fecha'])# time vector


ff = 5;   # frequency of the signal
y = Ens['Prediccion m']
n = len(y) # length of the signal
k = np.arange(n)
T = n/Fs
frq = k/T # two sides frequency range
frq = frq[range(n//2)] # one side frequency range

Y = np.fft.fft(y)/n # fft computing and normalization
Y = Y[range(n//2)]

fig, ax = plt.subplots(2, 1)
ax[0].plot(t,y)
ax[0].set_xlabel('Time')
ax[0].set_ylabel('Amplitude')
ax[1].plot(frq,abs(Y),'r') # plotting the spectrum
ax[1].set_xlabel('Freq (Hz)')
ax[1].set_ylabel('|Y(freq)|')

plot_url = offline.iplot(fig, filename='mpl-basic-fft')
```



4 conclusión

Encontramos que como esperamos el nivel del mar alrededor de un año sigue un comportamiento oscilatorio. Tras aplicar la serie de Fourier: tomando como correspondiente el valor X el tiempo y el Y el nivel en el que se encontraba el mar en dicha fecha, trasformamos el Eje X en frecuencia y se encontró que en seis meses se encontraron varios picos alrededor de los diez Hertz con un valor menos a 0.2m. En el dia 03 de noviembre se encontró un pico alrededor de los 15Hertz con un valor superior a los 0.25 metros y para el año 2018 se encontraron varios picos oscilando los 40 Hertz y los 70 Hertz, con un valor menor a los 0.2m

Referencias

https://ericstrong.org/fast-fourier-transforms-in-python/

https://jakevdp.github.io/blog/2013/08/28/understanding-the-fft/

https://en.wikipedia.org/wiki/Cooley