

# Reporte Eval.1

Eduardo Hndz.

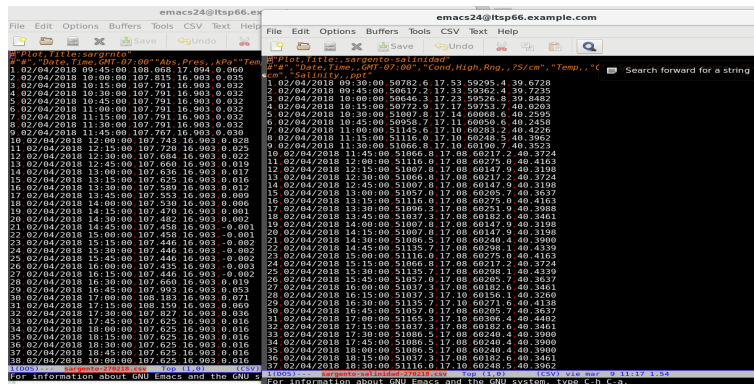
March 9, 2018

## 1 Introducción

En esta evaluación se nos pidió realizar importación, graficación y análisis de datos en *Jupyter Notebook* utilizando las librerías ya conocidas.

## 2 Actividad

1. De entrada tuvimos que descargar los datos con los que íbamos a realizar el análisis, los cuales nos proporcionaba el maestro, tuvimos que modificarlos ya que en un documento eran 2218 y en otro 2216, como ya había trabajado con datos irregulares, recordé que debíamos modificarlos para que fueran del mismo tamaño, también tuve que editarlo para que la fecha quedara en una sola columna y agregar espacios o comas, los datos quedaron como sigue:



```
emac24@itsp66.es: /home/eduardo/Box_Plot
File Edit Options Buffers Tools CSV Text Help
1 02/04/2018 09:45:00 108 068 17 494 0 068
2 02/04/2018 10:00:00 107 815 16 903 0 035
3 02/04/2018 10:15:00 107 791 16 903 0 032
4 02/04/2018 10:30:00 107 791 16 903 0 032
5 02/04/2018 10:45:00 107 791 16 903 0 032
6 02/04/2018 11:00:00 107 791 16 903 0 032
7 02/04/2018 11:15:00 107 791 16 903 0 032
8 02/04/2018 11:30:00 107 791 16 903 0 032
9 02/04/2018 11:45:00 107 791 16 903 0 032
10 02/04/2018 12:00:00 107 743 16 903 0 028
11 02/04/2018 12:15:00 107 720 16 903 0 025
12 02/04/2018 12:30:00 107 684 16 903 0 022
13 02/04/2018 12:45:00 107 660 16 903 0 018
14 02/04/2018 13:00:00 107 636 16 903 0 017
15 02/04/2018 13:15:00 107 625 16 903 0 016
16 02/04/2018 13:30:00 107 589 16 903 0 012
17 02/04/2018 13:45:00 107 553 16 903 0 009
18 02/04/2018 14:00:00 107 530 16 903 0 006
19 02/04/2018 14:15:00 107 508 16 903 0 001
20 02/04/2018 14:30:00 107 482 16 903 0 002
21 02/04/2018 14:45:00 107 458 16 903 0 001
22 02/04/2018 15:00:00 107 458 16 903 0 001
23 02/04/2018 15:15:00 107 446 16 903 0 002
24 02/04/2018 15:30:00 107 440 16 903 0 002
25 02/04/2018 15:45:00 107 446 16 903 0 002
26 02/04/2018 16:00:00 107 435 16 903 0 003
27 02/04/2018 16:15:00 107 466 16 903 0 019
28 02/04/2018 16:30:00 107 660 16 903 0 019
29 02/04/2018 16:45:00 107 993 16 903 0 051
30 02/04/2018 17:00:00 108 183 16 903 0 071
31 02/04/2018 17:15:00 108 159 16 903 0 069
32 02/04/2018 17:30:00 107 827 16 903 0 016
33 02/04/2018 17:45:00 107 825 16 903 0 016
34 02/04/2018 18:00:00 107 825 16 903 0 016
35 02/04/2018 18:15:00 107 825 16 903 0 016
36 02/04/2018 18:30:00 107 825 16 903 0 016
37 02/04/2018 18:45:00 107 825 16 903 0 016
38 02/04/2018 19:00:00 107 825 16 903 0 016

emac24@itsp66.example.com: /home/eduardo/Box_Plot
File Edit Options Buffers Tools CSV Text Help
1 02/04/2018 09:30:00 50762 6 17 53 59295 4 39 6728
2 02/04/2018 09:45:00 50637 2 17 33 59362 4 39 7235
3 02/04/2018 10:00:00 50646 3 17 23 59526 8 39 4482
4 02/04/2018 10:15:00 50772 9 17 17 59753 7 40 9203
5 02/04/2018 10:30:00 51007 8 17 14 60068 6 40 2595
6 02/04/2018 10:45:00 50956 7 17 11 60050 6 40 2458
7 02/04/2018 11:00:00 51145 6 17 10 60283 2 40 3726
8 02/04/2018 11:15:00 51116 6 17 10 60248 5 40 3962
9 02/04/2018 11:30:00 51066 8 17 10 60180 7 40 3525
10 02/04/2018 11:45:00 51046 8 17 08 60271 2 40 3724
11 02/04/2018 12:00:00 51116 0 17 08 60275 9 40 4163
12 02/04/2018 12:15:00 51007 8 17 08 60147 9 40 3188
13 02/04/2018 12:30:00 51066 8 17 08 60217 2 40 3724
14 02/04/2018 12:45:00 51007 8 17 08 60147 9 40 3188
15 02/04/2018 13:00:00 51057 9 17 08 60205 7 40 3637
16 02/04/2018 13:15:00 51116 8 17 08 60275 9 40 4163
17 02/04/2018 13:30:00 51096 3 17 08 60251 9 40 3988
18 02/04/2018 13:45:00 51097 8 17 08 60147 9 40 3188
19 02/04/2018 14:00:00 51097 8 17 08 60147 9 40 3188
20 02/04/2018 14:15:00 51096 5 17 08 60240 4 40 3980
21 02/04/2018 14:30:00 51096 5 17 08 60240 4 40 3980
22 02/04/2018 14:45:00 51135 7 17 08 60298 1 40 4139
23 02/04/2018 15:00:00 51116 0 17 08 60275 0 40 4163
24 02/04/2018 15:15:00 51066 8 17 08 60217 2 40 3724
25 02/04/2018 15:30:00 51135 7 17 08 60298 1 40 4139
26 02/04/2018 15:45:00 51017 0 17 08 60295 7 40 3637
27 02/04/2018 16:00:00 51037 3 17 08 60182 6 40 3461
28 02/04/2018 16:15:00 51017 3 17 10 60194 1 40 3760
29 02/04/2018 16:30:00 51135 7 17 10 60271 6 40 4138
30 02/04/2018 16:45:00 51037 0 17 08 60295 7 40 3637
31 02/04/2018 17:00:00 51165 3 17 10 60306 4 40 4402
32 02/04/2018 17:15:00 51037 0 17 08 60182 6 40 3461
33 02/04/2018 17:30:00 51086 5 17 08 60240 4 40 3980
34 02/04/2018 17:45:00 51086 5 17 08 60240 4 40 3980
35 02/04/2018 18:00:00 51086 5 17 08 60240 4 40 3980
36 02/04/2018 18:15:00 51037 0 17 08 60182 6 40 3461
37 02/04/2018 18:30:00 51116 0 17 10 60248 5 40 3962
```

2. Una vez editados los datos se procedió a ingresar a *Jupyter Notebook* para importar las bibliotecas con las que íbamos a trabajar y los datos. aquí utilizamos dos nuevas bibliotecas (usadas en la actividad5), una de ellas nos servirá para realizar **Box Plot** y la otra para convertir a Variables

temporales.

En este punto también se tuvo que importar/leer los datos con los que íbamos a trabajar y darles formato *Data Frame*

```
Branch: master | Computacional-1 / Evaluacion1 / Evaluación1.ipynb | Find File | Co
darkedy7 Add files via upload | 1 contributor | 2402 lines (2402 sloc) | 502 KB | Raw | Blame | History |

In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime
import seaborn as sns

In [2]: df0=pd.read_csv("sargento-270218.csv", header=None,skiprows=2)
df0.columns=['#','Date','APRE','Temp','WL']
df1=pd.read_csv("sargento-salinidad-270218.csv", header=None,skiprows=2)
df1.columns=['#','Date','CondHighRng','Temp','SC','PPM']

In [3]: df0=pd.DataFrame(df0)

In [4]: df0.head()
Out[4]:
```

#	Date	APRE	Temp	WL	
0	1	02/04/2018 09:45:00	108.068	17.094	0.060
1	2	02/04/2018 10:00:00	107.815	16.903	0.035
2	3	02/04/2018 10:15:00	107.791	16.903	0.032
3	4	02/04/2018 10:30:00	107.791	16.903	0.032
4	5	02/04/2018 10:45:00	107.791	16.903	0.032

```

In [5]: df0.dtypes
Out[5]:
```

#	
Date	object
APRE	float64
Temp	float64
WL	float64
dtype:	object

3. Como eran dos archivos de datos distintos, se les tuvo que dar formato a ambos.

```
In [7]: df1=pd.DataFrame(df1)

In [8]: df1.dtypes
Out[8]:
```

#	
Date	object
CondHighRng	float64
Temp	float64
SC	float64
PPM	float64
dtype:	object

```

In [9]: df1.head()
Out[9]:
```

#	Date	CondHighRng	Temp	SC	PPM
0	1	02/04/2018 09:30:00	50782.6	17.53	59295.4
1	2	02/04/2018 09:45:00	50617.2	17.33	59362.4
2	3	02/04/2018 10:00:00	50646.3	17.23	59526.8
3	4	02/04/2018 10:15:00	50772.9	17.17	59753.7
4	5	02/04/2018 10:30:00	51007.8	17.14	60068.6

```

In [10]: df1.describe()
Out[10]:
```

	#	CondHighRng	Temp	SC	PPM
count	2216.00000	2216.00000	2216.00000	2216.00000	2216.00000
mean	1108.50000	49436.482536	16.903294	58526.413854	39.091827
std	639.84842	1147.813884	0.669905	922.453266	0.697321
min	1.00000	46569.900000	14.100000	55305.400000	36.669100
25%	554.75000	48410.500000	16.530000	57655.400000	38.433300
50%	1108.50000	49294.000000	16.850000	58557.450000	39.114250
75%	1662.25000	50068.200000	17.390000	59244.200000	39.634000
max	2216.00000	52095.300000	18.100000	60306.400000	40.440200

4. Una vez que teníamos los datos procesados, debíamos realizar 3 gráficas en **Box Plot** pero para eso debíamos transformar el *Dataframe* de la fecha en variable temporal, ya que era de tipo OBJETO

```
In [11]: # Convertir la cadena de caracteres 'DateTime' en variable temporal 'NDateTime'  
df0['Ndt'] = pd.to_datetime(df0['Date'], format='%m/%d/%Y %H:%M:%S')  
df0['month'] = df0['Ndt'].dt.month
```

```
In [12]: df1['Ndt'] = pd.to_datetime(df1['Date'], format='%m/%d/%Y %H:%M:%S')  
df1['month'] = df1['Ndt'].dt.month
```

```
In [13]: ax = seb.boxplot(x="month", y="WL", data=df0, color='r')  
mpl.show()
```

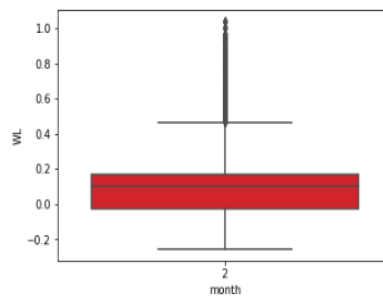


Figure 1: Nivel del Agua en Febrero

```
In [14]: ax = seb.boxplot(x="month", y="PPM", data=df1, color='y')  
mpl.show()
```

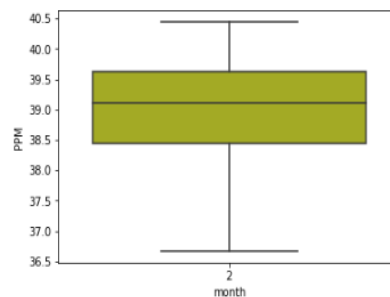


Figure 2: Salinidad en Febrero

Las Figuras (1,2,3) muestran los estadísticos de cada medición.

```
In [15]: ax = seb.boxplot(x="month", y="Temp", data=df1, color='b')
mpl.show()
```

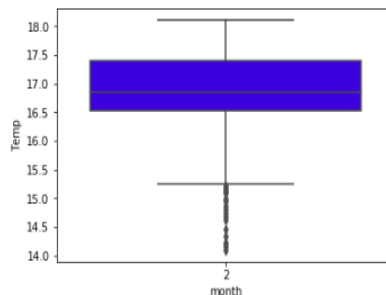


Figure 3: Temperatura-agua/Febrero

5. Posteriormente debía realizar 3 gráficas de correlación y aquí fue donde mi mente crasheó ya que debía unir dos dataframes para poder realizar la primer gráfica, ya que eran datos diferentes.  
para esto acudí a **San google** y tuve que buscar cómo se hacía eso y pude unir los datos con el comando **concat** el cual nos permite concatenar datos de diferentes archivos.

Aquí podemos observar que la correlación es muy poca ya que es de 0.21

```
In [18]: seb.set(style="darkgrid", color_codes=True)
df5=pd.concat([df0, df1], axis=1, join_axes=[df1.index])
g = seb.jointplot("WL", "ppm", data=df5, kind="reg",
color="r")
mpl.show(g)
```

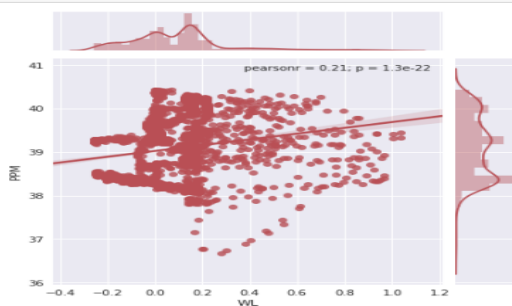


Figure 4: WL/Salinidad

Aquí fue más fácil realizar la gráfica, ya que los datos necesarios eran del mismo documento.

En la Figura 5 la posibilidad de correlación es muy baja.

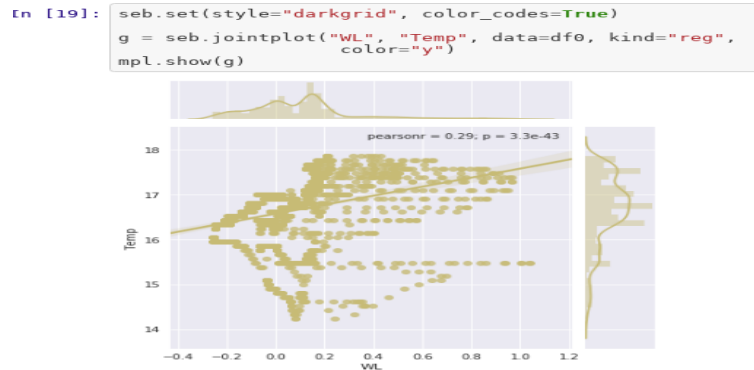


Figure 5: WL/Temperatura

En la figura 6 se acerca un poco más la posibilidad de correlación que en las figuras anteriores

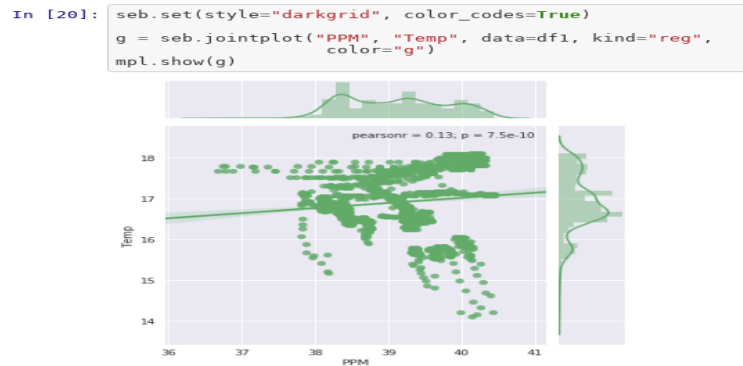


Figure 6: Salinidad/Temperatura

6. Después de ese SÚPER-ESTRÉS las cosas se falicitaron un poco ya que las próximas gráficas eran de una sola variable con respecto al tiempo.

```
In [21]: mpl.plot_date(x=df0.Date, y=df0.WL, fmt="r-")
mpl.title("Variación de Nivel del Agua con respecto al tiempo")
mpl.ylabel("Nivel del Agua(WL)")
mpl.grid(True)
mpl.show()
```



Figure 7: NivelAgua(WL)

```
In [22]: mpl.plot_date(x=df1.Date, y=df1.PPM, fmt="y-")
mpl.title("Salinidad/Tiempo")
mpl.ylabel("PPM")
mpl.grid(True)
mpl.show()
```

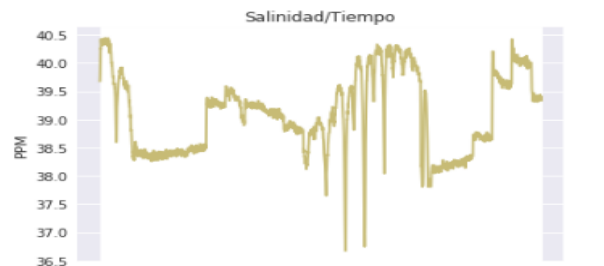


Figure 8: Salinidad

```
In [23]: mpl.plot_date(x=df0.Date, y=df0['Temp'], fmt="g-")
mpl.title("Variación de la Temperatura")
mpl.ylabel("Temp °C")
mpl.grid(True)
mpl.show()
```



Figure 9: Temperatura

7. Una vez de haber experimentado 5min. de desestrés era hora de graficar con algo nuevo, debía graficar dos gráficas sobre-puestas para esto acudí a **San google** y tuve que buscar cómo se hacía eso. La primer gráfica era de WL-Salinidad ambas con respecto al tiempo.

```
In [24]: fig, ax1 = mpl.subplots()
s=df0['#']
t=df1.PPM
u=df0.WL
ax1.plot(s,t,'r-')
ax1.set_xlabel('No. de Mediciones')
ax1.set_ylabel('Salinidad-WL')
ax2 = ax1.twinx()
ax2.plot(s,u,'b-')
ax2.set_ylabel('WL-Temp', color='y')
fig.tight_layout()
mpl.show()
```

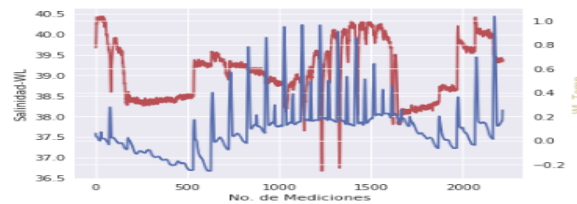


Figure 10: Salinidad-WL

La segunda gráfica era de WL-Temp en función del tiempo

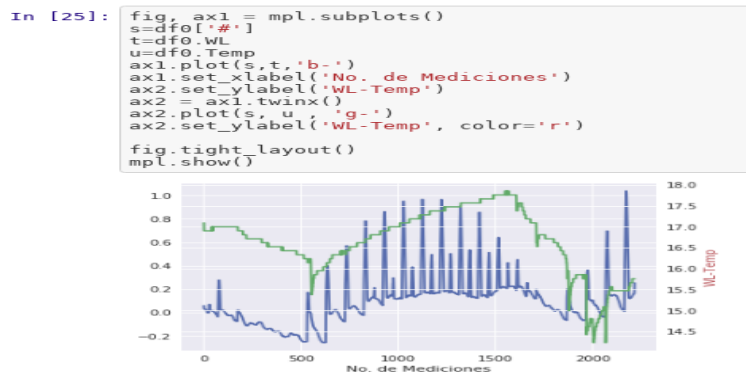


Figure 11: WL-Temp

8. Por último tuvimos que utilizar el comando *xlim* , el cual, como su nombre lo dice, podemos poner un límite de rango, de tal forma que limitamos las mediciones a 5 días para poder observar como se comportaban las funciones y ver que relación existía entre una y la otra. como podemos observar en la figura12 a medida que el nivel del agua aumenta la salinidad decrese.

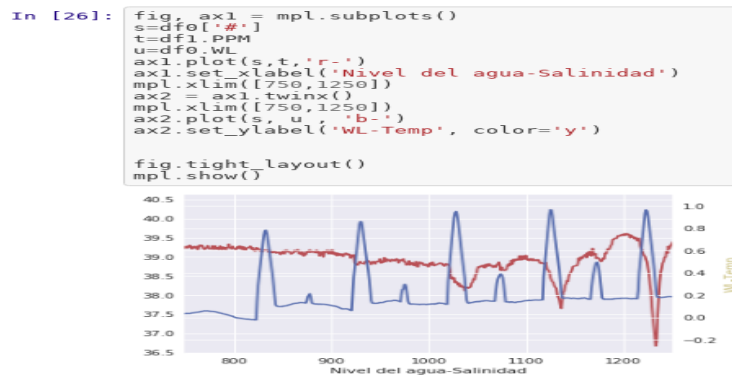


Figure 12: WL-Temp



En el caso de la Figura 13 es una gráfica de la temperatura y el nivel del agua en función del tiempo, de tal forma que al observar bien la gráfica podemos notar que conforme decrece la temperatura aumenta el nivel del mar

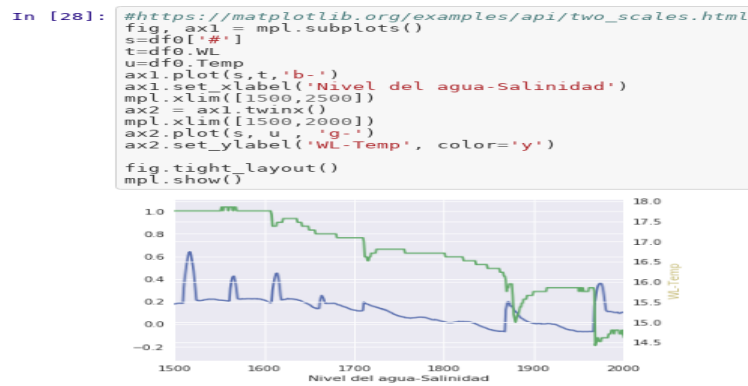


Figure 13: WL-Temp