Reporte - Evaluación 1

García Monge Itzel Alexia

08 de Marzo, 2018

1 Descarga de Datos

Se descargan los archivos que se proporcionan en la página de $PbWorks$, creando una carpeta llamada $Evaluacion1$ dentro de $FisicaComputacionalI$ para guardar los archivos.

Verificamos con $Emcas$ que los archivos descargados se encuentran en buen estado. Ya que los datos no necesitan de un reacomodo o filtro, la limpieza de datos puede llevarse acabo con $JupyterNotebook$, así que procedemos a abrir una terminal en la carpeta recién creada y abriendo el kernel.

2 Jupyter Notebook

2.1 Limpieza de datos

Se abre un nuevo Notebook de Python3 y se descargan las bibliotecas necesarias, como **numpy**, **pandas**, **matplotlib**, y **datetime**. Leemos los archivos descargados, llamando al archivo sargento como $dfsar$, mientras que el archivo sargento-salinidad fue llamado $dfsal$. Los archivos deben empezar y terminar con la misma fecha y hora, así que los leemos con pandas, llamando al archivo sargento como $dfsar$ y le ordenamos que se salte las primeras tres líneas de su documento, mientras que el archivo sargento-salinidad fue llamado $dfsal$ y se le ordenó saltar las primeras dos líneas. Con esto, ambos archivos comienzan en la misma fecha y hora.

Al saltar las líneas iniciales estamos borrando los títulos de las columnas, por lo que necesitamos reincorporarlas con:

```
dfsal.columns=['#','Date Time GMT 7','Cond High Rng S/cm','TempC','Specific Conductance
S/cm','Salinity']
dfsar.columns=['#','Date Time GMT 7','Abs Pres kPa','TempC','WaterLevel']
```

Como los dos archivos, aunque tengan algunos datos similares, no comparten todos los datos es importante tener cuidado con el nombre que se le otorga a cada columna.

Ahora falta que ambos archivos acaben en la misma fecha y hora, observando que solamente difieren por una hora de más encontrada en el archivo de $dfsar$, borránodse con:

```
dfsar=dfsar[:-1]
```

Así, ambos documentos están completamente en sincronía de fecha, obteniendo unas tablas de la forma:

	#	Date Time GMT 7	Abs Pres kPa	TempC	WaterLevel
0	2	10/26/2017 13:15:00	105.513	24.351	-0.160
1	3	10/26/2017 13:30:00	105.433	24.351	-0.168
2	4	10/26/2017 13:45:00	105.385	24.351	-0.173
3	5	10/26/2017 14:00:00	105.321	24.351	-0.179
4	6	10/26/2017 14:15:00	105.273	24.351	-0.184
5	7	10/26/2017 14:30:00	105.225	24.351	-0.189
6	8	10/26/2017 14:45:00	105.177	24.351	-0.193
7	9	10/26/2017 15:00:00	105.145	24.351	-0.196
8	10	10/26/2017 15:15:00	105.113	24.351	-0.200
9	11	10/26/2017 15:30:00	105.097	24.351	-0.201
10	12	10/26/2017 15:45:00	105.065	24.351	-0.204
11	13	10/26/2017 16:00:00	105.033	24.351	-0.208
12	14	10/26/2017 16:15:00	105.017	24.351	-0.209
13	15	10/26/2017 16:30:00	104.985	24.351	-0.212
14	16	10/26/2017 16:45:00	104.953	24.351	-0.216
15	17	10/26/2017 17:00:00	104.969	24.351	-0.214
16	18	10/26/2017 17:15:00	104.969	24.351	-0.214
17	19	10/26/2017 17:30:00	104.937	24.351	-0.217

dfsar

	#	Date Time GMT 7	Cond High Rng S/cm	TempC	Specific Conductance S/cm	Salinity
0	3	10/26/2017 13:15:00	54525.5	24.82	54719.0	36.2311
1	4	10/26/2017 13:30:00	54525.5	24.76	54783.8	36.2794
2	5	10/26/2017 13:45:00	54525.5	24.75	54794.6	36.2875
3	6	10/26/2017 14:00:00	54525.5	24.73	54816.2	36.3036
4	7	10/26/2017 14:15:00	54525.5	24.72	54827.0	36.3117
5	8	10/26/2017 14:30:00	54525.5	24.70	54848.7	36.3279
6	9	10/26/2017 14:45:00	54525.5	24.69	54859.5	36.3360
7	10	10/26/2017 15:00:00	54525.5	24.67	54881.2	36.3521
8	11	10/26/2017 15:15:00	54525.5	24.67	54881.2	36.3521
9	12	10/26/2017 15:30:00	54525.5	24.66	54892.0	36.3602
10	13	10/26/2017 15:45:00	54525.5	24.64	54913.7	36.3764
11	14	10/26/2017 16:00:00	54525.5	24.64	54913.7	36.3764
12	15	10/26/2017 16:15:00	54525.5	24.64	54913.7	36.3764
13	16	10/26/2017 16:30:00	54525.5	24.63	54924.6	36.3845
14	17	10/26/2017 16:45:00	54525.5	24.62	54935.4	36.3926
15	18	10/26/2017 17:00:00	54525.5	24.62	54935.4	36.3926
16	19	10/26/2017 17:15:00	54525.5	24.60	54957.1	36.4089
17	20	10/26/2017 17:30:00	54525.5	24.59	54968.0	36.4170

dfsal

Pero esto no significa que puedan leerse luego para graficar. Escribimos un *.dtypes* para observar qué tipo de variables tenemos, encontrando que todas se leen como punto flotante a excepción de la columna *Date*, la cual se lee como objeto en ambos archivos.

Para cambiar de objeto a punto flotante, usamos el código:

```
dfsal['Date'] = pd.to_datetime(dfsal['Date Time GMT 7'], format='%m/%d/%Y %H:%M:%S')
dfsal['month'] = dfsal['Date'].dt.month
```

En donde *Date* es la nueva variable que creamos para que se lea como punto flotante de la variable que seleccionamos, en este caso $dfsal$ y dentro de los corchetes seleccionamos la columna que vamos a cambiar a punto flotante, siendo cuidadosos de nombrar el formato como aparece en la columna que nombramos. Con *month* ahora podemos separar los dos meses y llamarlos por separado a graficar si se deseara.

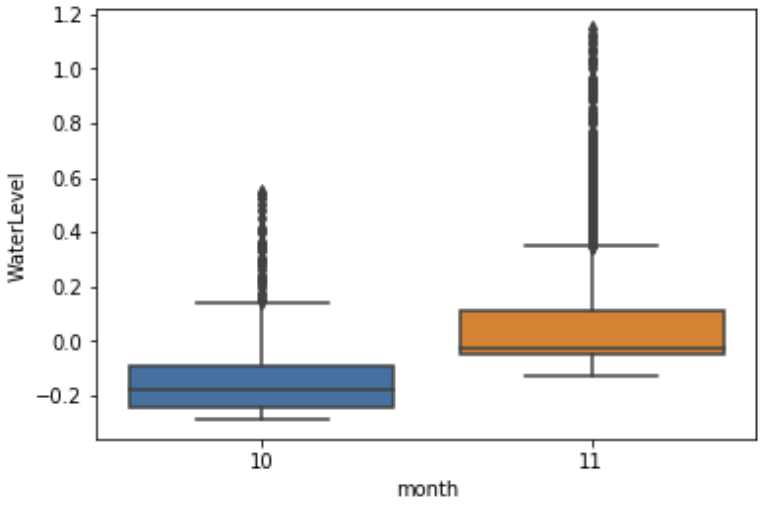
3 Gráficas

3.1 Boxplot

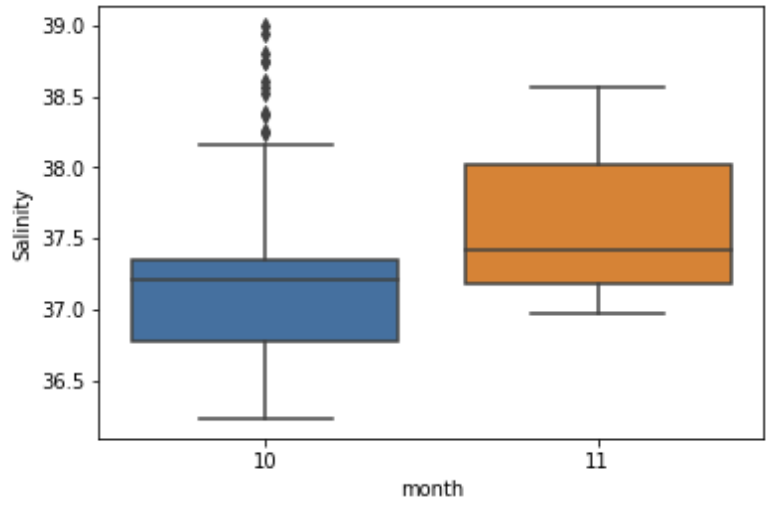
Para realizar las gráficas de boxplot es necesario importar la biblioteca **Seaborn**, lo cual se realiza antes de iniciar. Una vez descargada la biblioteca, se escribe un boxplot que crea dos gráficas, una para octubre y otra para noviembre.

El primer boxplot se del nivel del mar, así que usamos la variable $dfsar$ para graficar. El segundo boxplot es con respecto a la salinidad, por lo que se usa la variable $dfsal$. La tercera y última gráfica es con la temperatura, ya que ambas variables tienen esta columna podemos usar cualquiera, en este caso se eligió graficar la varibale $dfsar$.

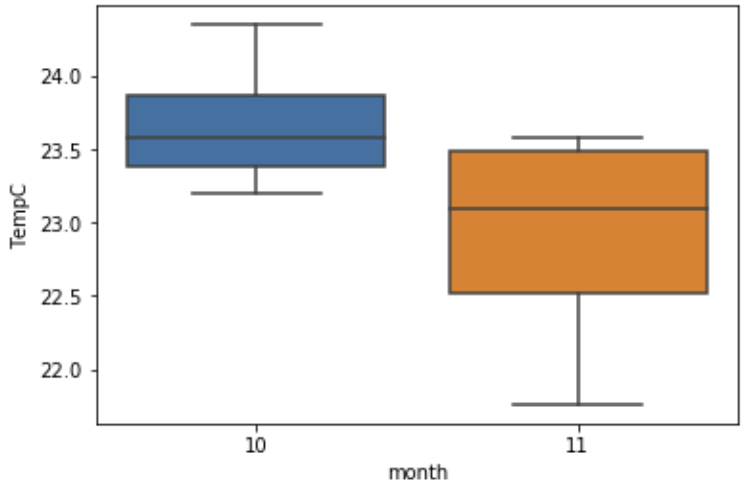
Las gráficas obtenidas son:



Nivel del Mar



Salinidad



Temperatura

Si se quiere saber con más detalle el valor de los datos, se usa el comando *.describe*() para obtener los valores precisos sobre la los cuartiles, la media, el número de datos, entre otros. Los valores de los

archivos son:

	#	Cond High Rng S/cm	TempC	Specific Conductance S/cm	Salinity	month
count	2393.00000	2393.000000	2393.000000	2393.000000	2393.000000	2393.000000
mean	1199.00000	54524.972587	23.315980	56387.569118	37.480289	10.781446
std	690.94392	11.879147	0.546177	618.579547	0.464286	0.413352
min	3.00000	54105.700000	21.490000	54719.000000	36.231100	10.000000
25%	601.00000	54525.500000	22.730000	55949.700000	37.151400	11.000000
50%	1199.00000	54525.500000	23.490000	56185.600000	37.328300	11.000000
75%	1797.00000	54525.500000	23.700000	57053.700000	37.980300	11.000000
max	2395.00000	54525.500000	24.820000	58398.700000	38.994200	11.000000

Resumen de datos: dfsal

	#	Abs Pres kPa	TempC	WaterLevel	month
count	2393.00000	2393.000000	2393.000000	2393.000000	2393.000000
mean	1198.00000	107.430767	23.120328	0.030939	10.781446
std	690.94392	2.372048	0.563019	0.235994	0.413352
min	2.00000	104.229000	21.760000	-0.288000	10.000000
25%	600.00000	106.407000	22.525000	-0.071000	11.000000
50%	1198.00000	106.764000	23.388000	-0.035000	11.000000
75%	1796.00000	107.307000	23.484000	0.019000	11.000000
max	2394.00000	118.641000	24.351000	1.146000	11.000000

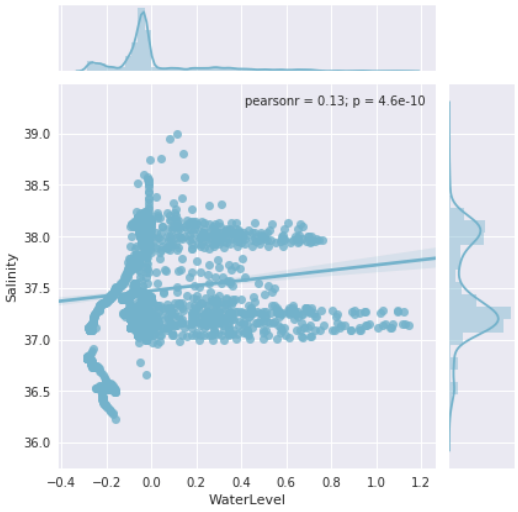
Resumen de datos: dfsar

3.2 Regresión Lineal

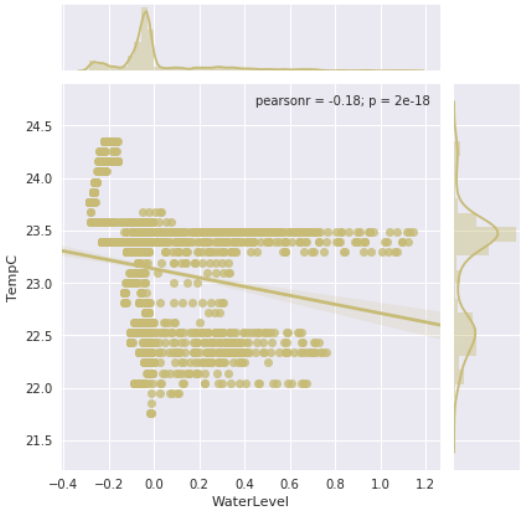
Para realizar las gráficas de regresión lineal se sigue usando la biblioteca **Seaborn**. La primera gráfica que se nos pide dice quiere que sea una relación entre el nivel del agua y la salinidad. Para poder graficar esta regresión debemos concatar ambas variables, ya que un valor se encuentra en cada una. Para concatar ambas variables se crea una variable nueva que vaya a guardar la información de ambos.

Para las siguientes dos gráficas de regresión no se necesita concatar ni usar la grafica concatada, pues nos piden una gráfica con ajuste lineal del nivel de mar y temperatura del agua, valores que podemos encontrar en la variable $df sar$; mientras que la tercera gráfica nos pide una regresión lineal de la salinidad y la temperatura, valores que se encuentran en la variable $df sal$.

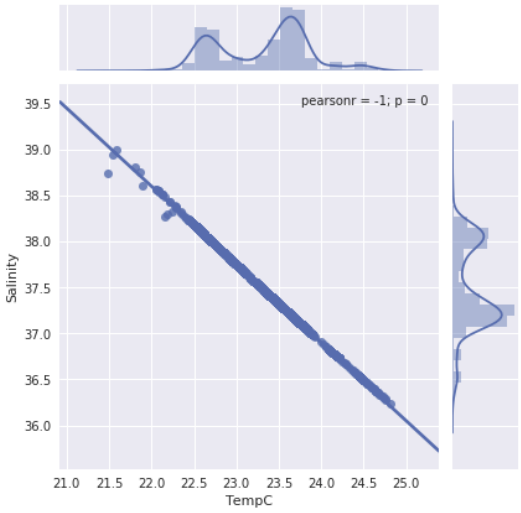
Las gráficas adquieren la forma de:



Nivel del Mar



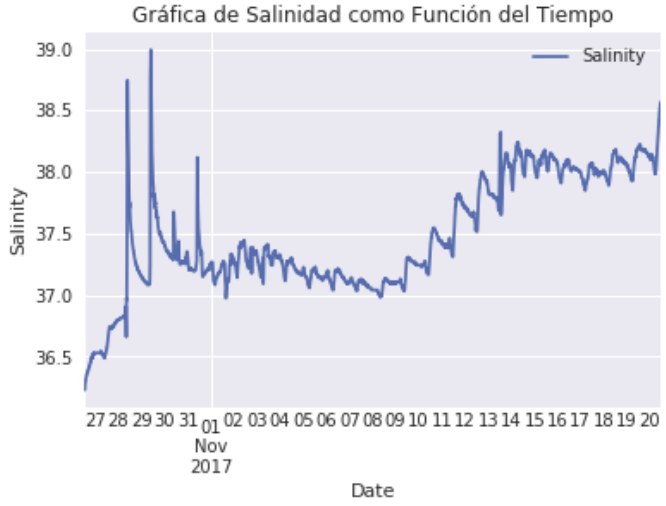
Salinidad

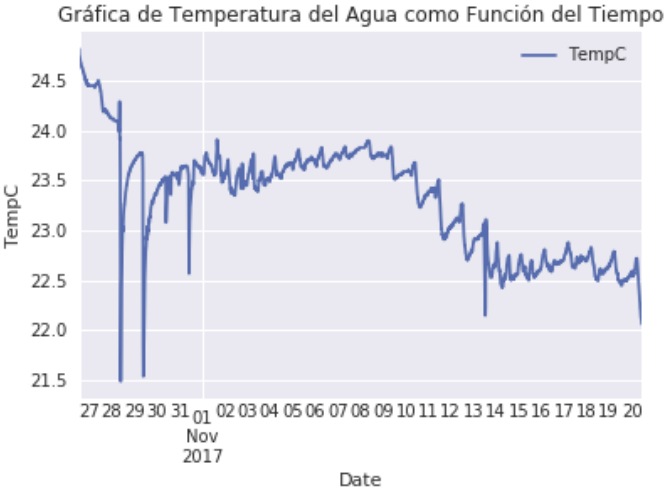


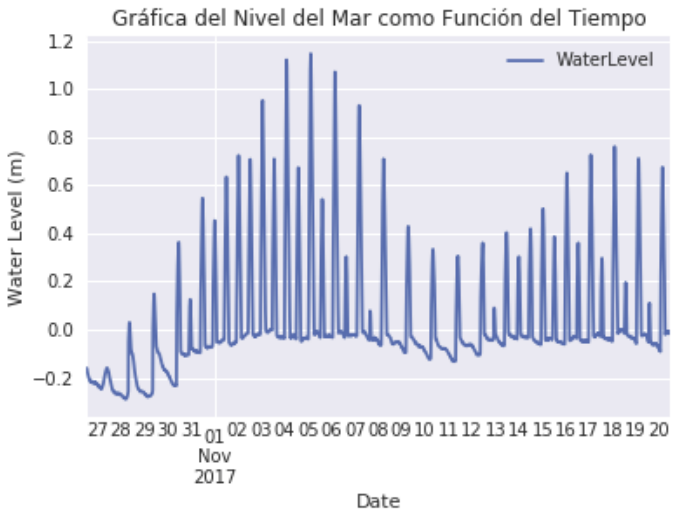
Temperatura

3.3 Gráficas Independientes

Además de **Seaborn** todavía tenemos otra biblioteca con la que podemos graficar relaciones. Se quiere graficar la relación que tiene la temperatura, la salinidad y el nivel del agua como función del tiempo de manera independiente. Para eso, solo necesitamos de **matplotplib**, creando una nueva variable que será la gráfica y obteniendo las relaciones siguientes:





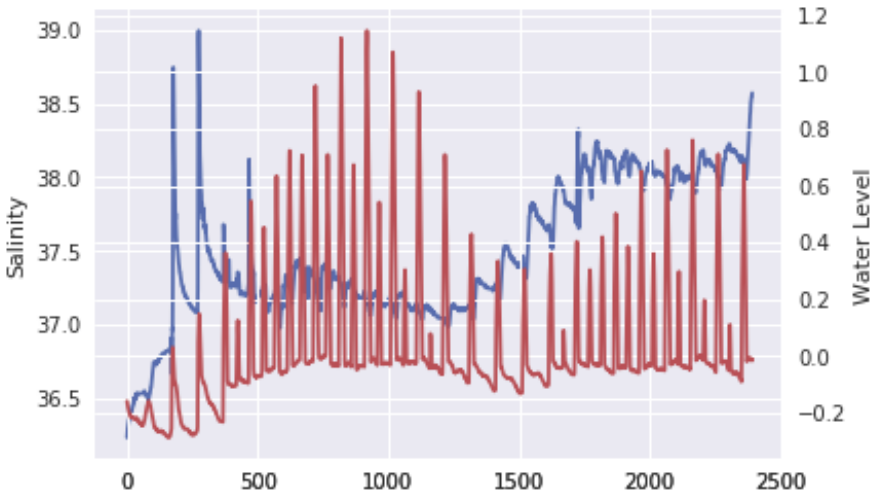


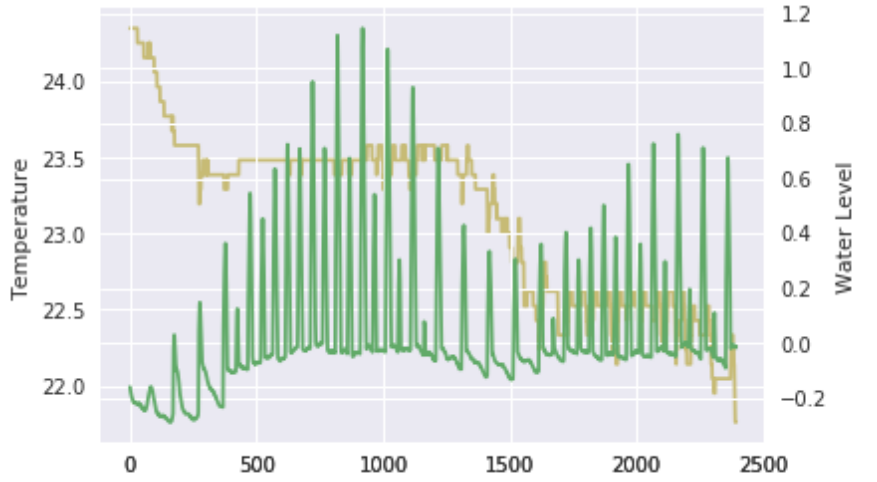
3.4 Doble Eje

Cuando se desea graficar de forma que no ambos valores se encuentren en el eje y, y no de la manera convencional eje x con eje y, se debe importar una nueva biblioteca llamada **pylab**. Con este biblioteca puedes seleccionar qué quieres que vaya en el lado izquierdo del eje y y qué deseas que vaya en el lado derecho, construyendo un código, por ejemplo, de la forma:

```
from pylab import figure, show, legend, ylabel
fig1 = figure()

ax1 = fig1.add_subplot(111)
line1 = ax1.plot(dfconjunto['Salinity'], 'b-')
ylabel("Salinity")

ax2 = fig1.add_subplot(111, sharex=ax1, frameon=False)
line2 = ax2.plot(dfconjunto['WaterLevel'], 'xr-')
ax2.yaxis.tick_right()
ax2.yaxis.set_label_position("right")
ylabel("Water Level")

show()
```

Se realizaron dos gráficas de doble eje, una con nivel del mar y salinidad–la cual requirió usar la variable *dfconjunto* una vez más–y la segunda con temperatura y nivel del mar.





3.5 Límites

A las gráficas recién creadas de doble eje se les quiere tener graficadas de una manera aún más específica, pidiendo que de todos los días de los dos meses que se tienen disponibles, se seleccionen 5. Lo que al principio puede complicar es el hecho de que se pida una restricción que trata con las fechas cuando ninguno de los ejes graficados lo son, pero se resuelve fácilmente al agregar la fecha en el eje x inferior no utilizado, usando tres variables en lugar de dos al escribir el código de la forma:

```
fig, ax1 = plt.subplots()
x=dfsal['Date']
Sal=dfconjunto.Salinity
Water=dfconjunto.WaterLevel

ax1.plot(x,Sal,'g-', label='Salinity'); plt.legend(loc='upper left')
ax1.set_xlabel('Date')
ax1.set_ylabel('Salinity (ppt)')

ax2 = ax1.twinx()
```

```
ax2.plot(x, Water, 'c-', label='WaterLevel'); plt.legend(loc='upper right')
ax2.set_ylabel('Water Level (m)')

fig.tight_layout()
plt.xlim(("2017-10-26 13:00:00","2017-10-30 13:00:00"))
plt.show()
```

Y obteniendo unas gráficas más claras.