Evaluación 1: Análisis de las mareas y salinidad en el Manglar El Sargento.

Jesús Antonio González Espinosa

Física Computacional 1

8 de Marzo del 2018

1 Archivos

Después de crear la carpeta Evaluación1,iniciamos la actividad descargando los dos archivos de datos de El Sargento que vienen en la página del curso. La instrucción indica que nos tenemos que asegurar que ambos archivos abarquen el mismo periodo de tiempo, entonces, con Emacs analizamos los datos para ver que tan desfasados estaban. Después de una rápida hojeada a los datos, se pudo observar que solo eran dos renglones los que causaban que los datos no estuvieran en el mismo periodo de tiempo: el primer renglón del archivo $sargento - salinidad - 201117.csv$ y el último renglón de $sargento_2 01117$. Entonces para poder corregirlos, teníamos que deshacernos de esos dos, pero al saber que solo eran renglones a los extremos, se decidió dejarlo la corrección para después, ya que contamos con las herramientas necesarias en Pandas, para hacerlo más rápido.

2 Jupyter Notebook

Lo que sigue de la actividad fue abrir $JupyterNotebook$ en la carpeta de Evaluación1. Ahí, creamos un documento tipo notebook con Python 3. Ahí cargamos las biblotecas de Pandas, Numpy, datetime, matplotlib.pyplt y seaborn, las cuales son necesarias para hacer la actividad.

2.1 Lectura en Pandas

Primero cargamos el archivo $sargento_2 01117.csv$ bajo el data frame de nombre $dfsargento1$, asegurándonos de saltarnos las dos primeros renglones del documento, que son texto innecesario, darle nombre a las columnas y hacer que ignore el último renglón, para que este coincida con el final del otro archivo. Asimismo, cargamos el otro archivo, $sargento - salinidad - 201117.csv$ bajo el data frame $dfsargento2$, hacemos lo mismo, pero ahora saltando los tres primeros renglones, siendo uno de datos pero a una hora que no incluye el otro archivo; y finalmente le damos nombre a las columnas. Seguido a esto, se imprimió en la pantalla la tabla con los últimos datos para asegurarnos que ambos tengan la misma cantidad de renglones y que acaben a la misma hora:

```
#Leyendo datos del primer archivo.
dfsargento1 = pd.read_csv("sargento_201117.csv", skiprows=2, header=None, names=['Num','Date','AbsPres','Temp','Water Level'])
dfsargento1 = dfsargento1[:-1]
dfsargento1.tail()
```

	Num	Date	AbsPres	Temp	Water Level
2389	2390	11/20/2017 10:15:00	106.986	21.855	-0.013
2390	2391	11/20/2017 10:30:00	106.998	21.760	-0.012
2391	2392	11/20/2017 10:45:00	106.998	21.760	-0.012
2392	2393	11/20/2017 11:00:00	106.950	21.760	-0.017
2393	2394	11/20/2017 11:15:00	106.966	21.760	-0.015

```
#Leyendo datos del segundo archivo.
dfsargento2 = pd.read_csv("sargento-salinidad-201117.csv", skiprows=3, header=None, names=['Num','Date','CondHighRng','Temp','Spe
dfsargento2.tail()
```

	Num	Date	CondHighRng	Temp	SpecConduct	Salinity
2389	2391	11/20/2017 10:15:00	54525.5	22.12	57766.8	38.5173
2390	2392	11/20/2017 10:30:00	54525.5	22.09	57802.3	38.5440
2391	2393	11/20/2017 10:45:00	54525.5	22.08	57814.1	38.5530
2392	2394	11/20/2017 11:00:00	54525.5	22.08	57814.1	38.5530
2393	2395	11/20/2017 11:15:00	54525.5	22.06	57837.8	38.5708

Ahora, vimos el tipo de datos que son, para asegurar que todos sean variables, y se pudo notar que la fecha era de tipo objeto, pero gracias a la libereria datetime pudimos arreglar eso y crear dos nuevas columnas, una para todas las fechas, y otra para los meses. Finalmente, los datos están listos para ser utilizados y graficados.

```
#Cambiando el formato de la fecha, para poder usarlo como variable.

#Archivo 1.
dfsargento1['Ndate'] = pd.to_datetime(dfsargento1['Date'], format='%m/%d/%Y %H:%M:%S')
dfsargento1['Month'] = dfsargento1['Ndate'].dt.month

#Archivo 2.
dfsargento2['Ndate'] = pd.to_datetime(dfsargento2['Date'], format='%m/%d/%Y %H:%M:%S')
dfsargento2['Month'] = dfsargento2['Ndate'].dt.month
```

```
#Revisando las nuevas variables de tiempo del primer archivo.
dfsargento1.dtypes
```

```
Num                    int64
Date                  object
AbsPres              float64
Temp                 float64
Water Level          float64
Ndate         datetime64[ns]
Month                  int64
dtype: object
```

```
#Revisando las nuevas variables de tiempo del segundo archivo.
dfsargento2.dtypes
```

```
Num                    int64
Date                  object
CondHighRng          float64
Temp                 float64
SpecConduct          float64
Salinity             float64
Ndate         datetime64[ns]
Month                  int64
dtype: object
```

2.2 Gráficas Seaborn

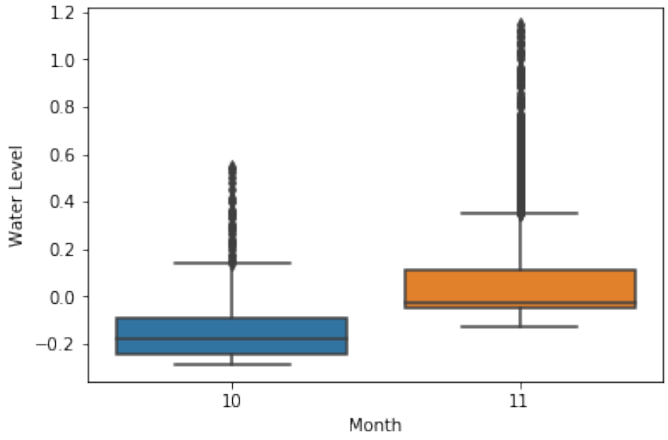
Ahora, con ayuda de la librería Seaborn, vamos a graficar varios tipo de gráficas, para observar los datos.

2.2.1 Boxplot

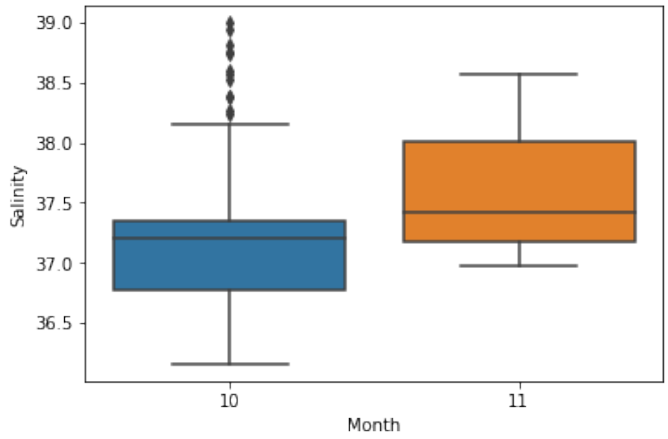
El primer tipo de gráfica es boxplot. Para cada dato que se grafique, se van a mostrar dos, una correspondiente a cada mes. El código utilizado fue: Cada una creó las siguientes gráficas:

```
#3
#Boxplot Meses/Water level.
ax = sns.boxplot(x="Month", y="Water Level", data=dfsargento1)
plt.show()

#Boxplot Meses/Salinity.
ax = sns.boxplot(x="Month", y="Salinity", data=dfsargento2)
plt.show()

#Boxplot Meses/Temperatura.
ax = sns.boxplot(x="Month", y="Temp", data=dfsargento1)
plt.show()
```

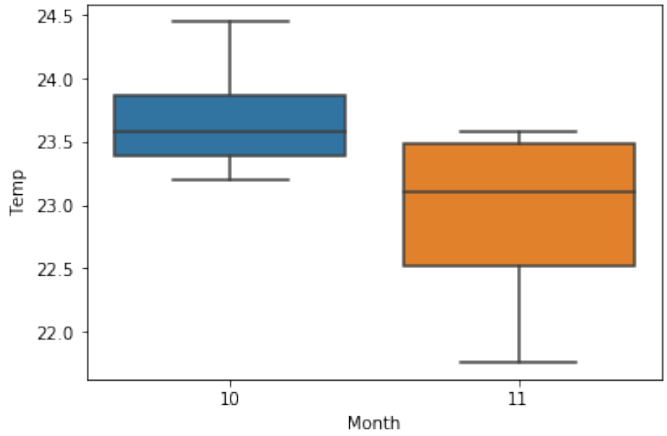
La primer es del Nivel del Mar:



El siguiente boxplot es de Salinidad:



El último de lox boxplot es de la Temperatura del Agua:



También podemos mostrar más información con la función *describe*, que muestra la posición de la mediana, los cuarteles, los máximos y mínimos.

```
#Exactitud de la posición de la mediana, cuarteles, máximos y mínimos del primer archivo.
dfsargento1.describe()
```

	Num	AbsPres	Temp	Water Level	Month
count	2394.000000	2394.000000	2394.000000	2394.000000	2394.000000
mean	1197.500000	107.430007	23.120883	0.030863	10.781119
std	691.232595	2.371844	0.563555	0.235974	0.413574
min	1.000000	104.229000	21.760000	-0.288000	10.000000
25%	599.250000	106.407000	22.525000	-0.071000	11.000000
50%	1197.500000	106.764000	23.388000	-0.035000	11.000000
75%	1795.750000	107.305000	23.484000	0.018750	11.000000
max	2394.000000	118.641000	24.448000	1.146000	11.000000

```
#Exactitud de la posición de la mediana, cuarteles, máximos y mínimos del primer archivo del segundo archivo.
dfsargento2.describe()
```

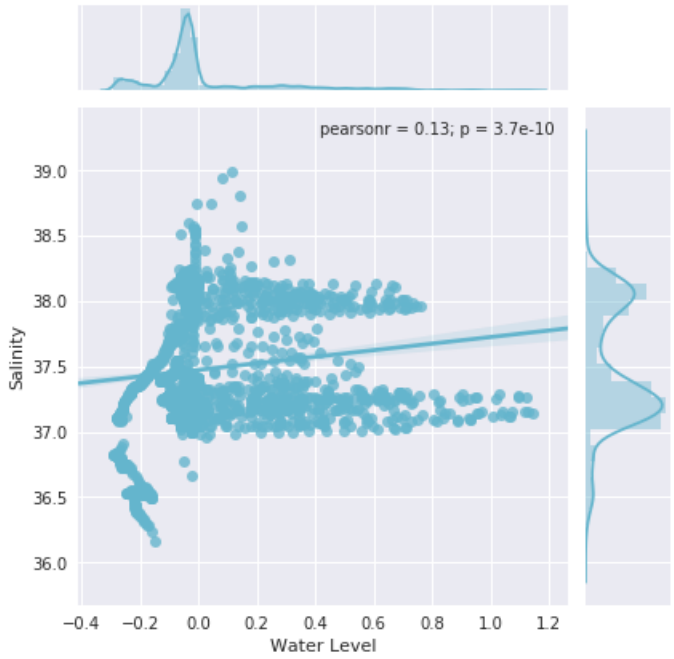
	Num	CondHighRng	Temp	SpecConduct	Salinity	Month
count	2394.000000	2394.000000	2394.000000	2394.000000	2394.000000	2394.000000
mean	1198.500000	54524.972807	23.316646	56386.831662	37.479737	10.781119
std	691.232595	11.876669	0.547033	619.501987	0.464974	0.413574
min	2.000000	54105.700000	21.490000	54622.100000	36.158800	10.000000
25%	600.250000	54525.500000	22.730000	55949.700000	37.151400	11.000000
50%	1198.500000	54525.500000	23.490000	56185.600000	37.328300	11.000000
75%	1796.750000	54525.500000	23.700000	57053.700000	37.980300	11.000000
max	2395.000000	54525.500000	24.910000	58398.700000	38.994200	11.000000

2.2.2 Correlación de Pearson

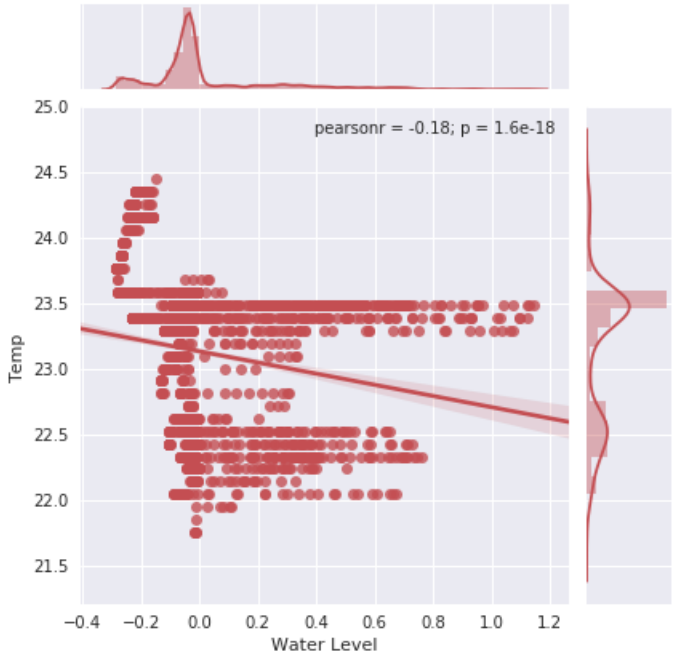
El segundo tipo de gráficas que se hacen son de Correlación de Pearson, en la que se graficaron parejas de variables. El código utilizado fue: Cada una creó las siguientes gráficas:

```
#4
#Correlación de Pearson entre Water Level y Salinidad.
sns.set(style="darkgrid", color_codes=True)
dfsargento3=pd.concat([dfsargento1, dfsargento2], axis=1, join_axes=[dfsargento2.index])
sns.jointplot("Water Level", "Salinity", data=dfsargento3,kind="reg", color="c")
plt.show()

#Correlación de Pearson entre Water Level y Temperatura.
sns.set(style="darkgrid", color_codes=True)
sns.jointplot("Water Level", "Temp", data=dfsargento1,kind="reg", color="r")
plt.show()

#Correlación de Pearson entre Salinidad y Temperatura.
sns.set(style="darkgrid", color_codes=True)
sns.jointplot("Salinity", "Temp", data=dfsargento2,kind="reg", color="g")
plt.show()
```

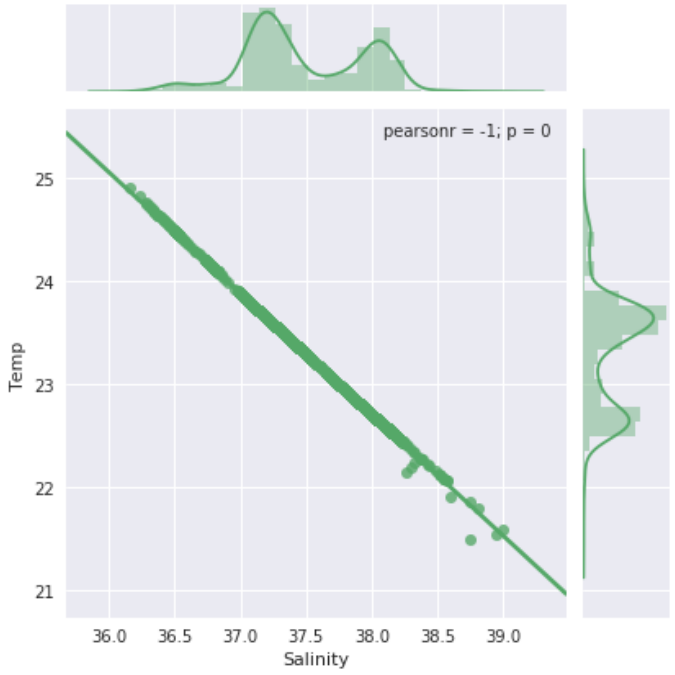
El primer par de datos son Nivel de Mar - Salinidad:



El siguiente es Nivel de Mar - Temperatura de Agua:



Finalmente tenemos Salinidad - Temperatura del Agua:



Con esto, terminamos las gráficas de Seaborn.

2.3 Gráficas Matplotlib

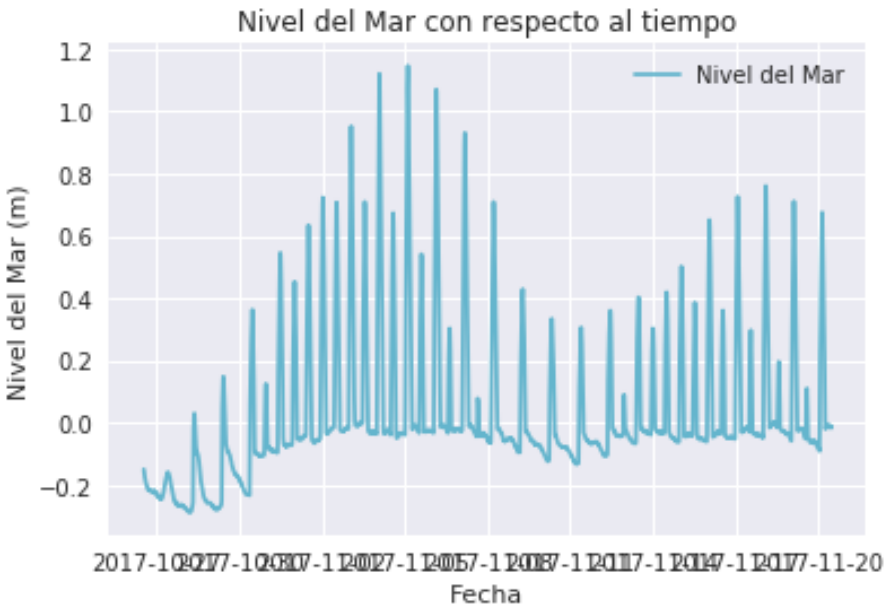
Ahora, empezamos a usar la librería Matplotlib para gráficar.

2.3.1 Gráficas Independientes

Aquí se usó la bibloteca matplotlib para graficar datos en función del tiempo. El código utilizado fue: Los cuales forman las siguientes gráficas:

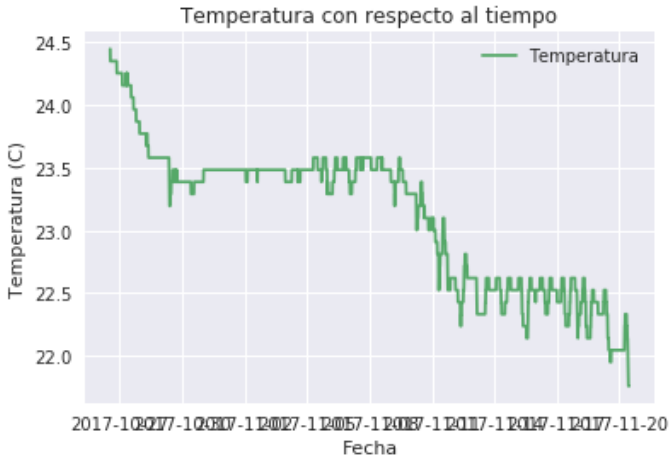
```
#5
#Gráfica independiente de Nivel del Mar en función del tiempo.
WL = dfsargento1['Water Level']
Date = dfsargento1['Ndate']
plt.plot(Date, WL, "c", label='Nivel del Mar'); plt.legend(loc='best')
plt.title('Nivel del Mar con respecto al tiempo')
plt.ylabel('Nivel del Mar (m)')
plt.xlabel('Fecha')
plt.grid(True)
plt.show()

#Gráfica independiente de Salinidad en función del tiempo.
Sal = dfsargento2['Salinity']
Date = dfsargento2['Ndate']
plt.plot(Date, Sal, "r", label='Salinidad'); plt.legend(loc='best')
plt.title('Salinidad con respecto al tiempo')
plt.ylabel('Salinidad (ppt)')
plt.xlabel('Fecha')
plt.grid(True)
plt.show()

#Gráfica independiente de Temperatura en función del tiempo.
Temp = dfsargento1['Temp']
Date = dfsargento1['Ndate']
plt.plot(Date, Temp, "g", label='Temperatura'); plt.legend(loc='best')
plt.title('Temperatura con respecto al tiempo')
plt.ylabel('Temperatura (C)')
plt.xlabel('Fecha')
plt.grid(True)
plt.show()
```

La primera es del Nivel del Mar como función del tiempo:



La siguiente es de Salinidad como función del tiempo:

La última gráfica creada fue de Temperatura del Agua como función del tiempo:



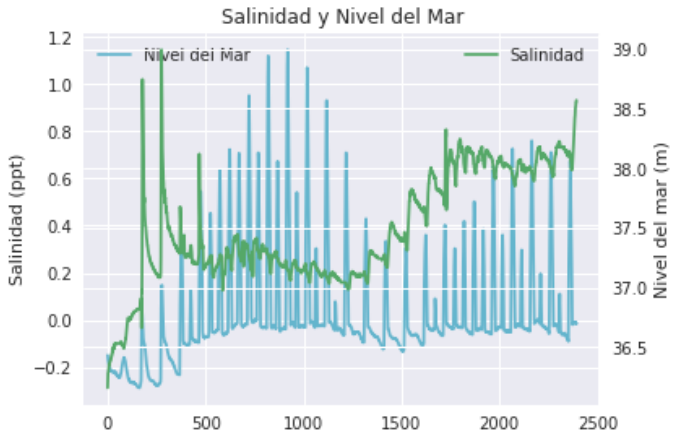
2.3.2 Gráficas con doble Eje

De nuevo, utilizando Matplotlib se generaron gráficas superpuestas con doble eje vertical. Para estas se hizo el siguiente código:

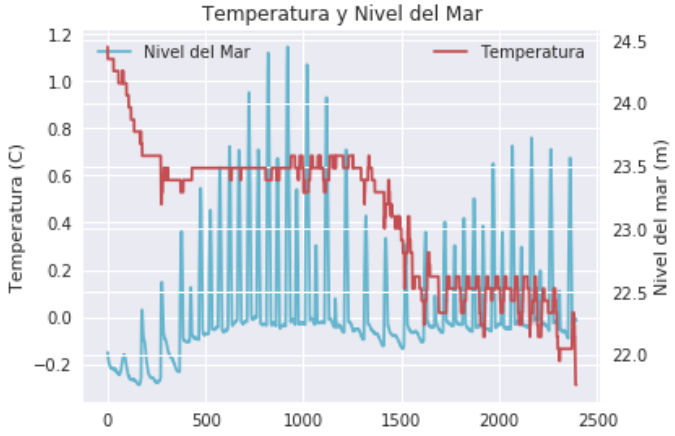
```
#6
#Grafica de doble eje Salinidad y nivel del mar
fig, ax1 = plt.subplots()
sal=dfsargento2['Salinity']
WL=dfsargento1['Water Level']
plt.title('Salinidad y Nivel del Mar')
ax1.plot(WL,'c', label='Nivel del Mar'); plt.legend(loc='upper left')
ax1.set_ylabel('Salinidad (ppt)')
ax2 = ax1.twinx()
ax2.plot(sal , 'g', label='Salinidad'); plt.legend(loc='upper right')
ax2.set_ylabel('Nivel del mar (m)')
fig.tight_layout()
plt.show()

#Grafica de doble eje Nivel del Mar y Temperatura.
fig, ax1 = plt.subplots()
temp=dfsargento1['Temp']
WL=dfsargento1['Water Level']
plt.title('Temperatura y Nivel del Mar')
ax1.plot(WL,'c', label='Nivel del Mar'); plt.legend(loc='upper left')
ax1.set_ylabel('Temperatura (C)')
ax2 = ax1.twinx()
ax2.plot(temp , 'r', label='Temperatura'); plt.legend(loc='upper right')
ax2.set_ylabel('Nivel del mar (m)')
fig.tight_layout()
plt.show()
```

El cual creó las siguientes gráficas:

La primera es de Nivel del Mar y Salinidad:



Y la segunda, y última, es de Nivel de Mar y Temperatura:



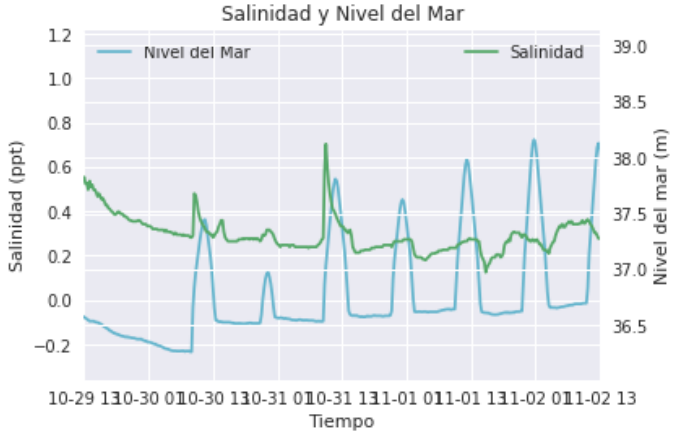
2.3.3 Gráficas con doble Eje con xlim

Ahora, de nuevo repetimos el código y proceso de las últimas dos gráficas, pero vamos a hacer que avancen en función del tiempo y vamos a usar la función xlim, para limitar las imágenes de las gráficas a cinco días. El código modificado a las especificaciones nuevas fue:

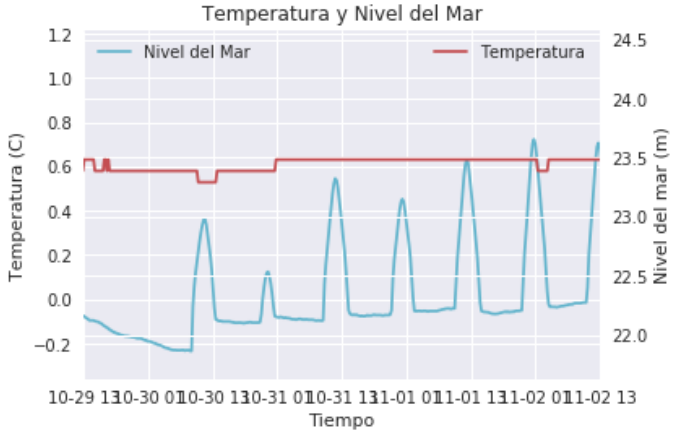
```
#7
#Grafica de doble eje Salinidad y nivel del mar del 29/10/2017 - 2/11/2017.
fig, ax1 = plt.subplots()
date=dfsargento1['Ndate']
sal=dfsargento2['Salinity']
WL=dfsargento1['Water Level']
plt.title('Salinidad y Nivel del Mar')
ax1.plot(date,WL,'c', label='Nivel del Mar'); plt.legend(loc='upper left')
ax1.set_xlabel('Tiempo')
ax1.set_ylabel('Salinidad (ppt)')
ax2 = ax1.twinx()
ax2.plot(date, sal , 'g', label='Salinidad'); plt.legend(loc='upper right')
ax2.set_ylabel('Nivel del mar (m)')
fig.tight_layout()
plt.xlim(("2017-10-29 13:00:00","2017-11-2 13:00:00"))
plt.show()

#Grafica de doble eje Nivel del Mar y Temperatura del 29/10/2017 - 2/11/2017.
fig, ax1 = plt.subplots()
date=dfsargento1['Ndate']
temp=dfsargento1['Temp']
WL=dfsargento1['Water Level']
plt.title('Temperatura y Nivel del Mar')
ax1.plot(date, WL ,'c', label='Nivel del Mar'); plt.legend(loc='upper left')
ax1.set_xlabel('Tiempo')
ax1.set_ylabel('Temperatura (C)')
ax2 = ax1.twinx()
ax2.plot(date, temp , 'r', label='Temperatura'); plt.legend(loc='upper right')
ax2.set_ylabel('Nivel del mar (m)')
fig.tight_layout()
plt.xlim(("2017-10-29 13:00:00","2017-11-2 13:00:00"))
plt.show()
```

El cual creó las siguientes gráficas:

La primera, Salinidad y Nivel del Mar en función del tiempo, del día 29/10/2017, hasta el día 2/11/2017:



La última gráfica de esta parte y toda la actividad, que es del Nivel del Mar y la Temperatura en función del tiempo, del día 29/10/2017, hasta el día 2/11/2017:



Al ver las gráficas, podemos observar que en la de la Salinindad y el Nivel del Mar sí parece haber ciertos indicios de una relación, ya que en la primera mitad de la gráfica parece mostrar que si una aumenta, la otra también parece aumentar, pero en la siguiente mitad, parece hacer lo contrario. Por lo tanto, podemos decir que sí parece existir una relación entre los datos, solo que esta no es muy aparente. Por otra parte, en la gráfica de Temperatura y Nivel del Mar no parece haber relación alguna entre los datos.

3 Conclusión

Fue interesante trabajar en esta evaluación, ya que por una parte trabajamos con datos locales, lo cual a mi parecer es muy interesante, y por otro lado, porque unificamos todas las herramientas que adquirimos en lo que lleva el curso. Otro factor importante fue el hecho de que la evaluación fue como hacer una actividad pero contrarreloj, lo que lo hizo parecer como un reto, que a mi parecer lo hizo divertido. Al final de cuentas, el tiempo sí fue suficiente, y el trabajo realizado se ha sentido satisfactorio.