Geography 132 -- Assignment #2

Due Feb. 9, 2022, 12:30 pm

Instructions

When you submit this assignment, replace "GEOG132" with your LAST NAME, then download from Jupyter as an *HTML file*, and upload to GauchoSpace.

For this assignment you will analyze spatial and temporal patterns in water quality in Campus Lagoon. The goals for this assignment are to:

- learn to make plots showing spatial data in Python
- interpret the observations in the context of possible physical drivers
- think about coastal ecosystems using a control volume approach
- link space and time through flow rates

The Situation

You work for an environmental consulting company that has been hired to conduct a study of water quality conditions in UCSB's Campus Lagoon. More specifically, you have been tasked with assessing eutrophiciation and hypoxia concerns. You are in the early stages of Phase I of the project, which involves initial site characterization and baseline (dry weather) monitoring. Your boss asks you to produce a report that:

- describes why and how nutrients and oxygen demand are linked in coastal waters
- gives an overview of the lagoon
- presents the results of your initial sampling effort
- proposes a comprehensive monitoring plan for both dry and wet conditions

You hired someone to collect water quality data in Campus Lagoon. The following were collected with a YSI 2030 Pro along a longitudinal transect (that is, along the length of the lagoon) and sampled at both the surface and near the bed (when the water water deep enough).

- temperature

- conductivity
- dissolved oxygen

Additionally, a PME MiniDOT was deployed for two weeks prior to the longitudinal samling. The resulting timeseries of temperature and dissolved oxygen in the lagoon is availabe to provide context to the point samples collected using the YSI.

Background

Write a two-part introduction. The **first part** should describe how nutrient enrichment leads to eutrophiciation and hypoxia in coastal ecosystems. Your goal is to provide context for the reader to understand the data presented later in the report. This synthesis should come from the reading assignment (there are two PDFs in GauichoSpace) and a review of the lecture notes. The **second part** should introduce the reader to Campus Lagoon. The purpose here is to provide necessary context for a reader unfamiliar with the site. (Is it tidal? Where does the water come from? What are the physical dimensions? What's the local climate? Does wind matter?) In this real-worlkd scenario, there's not necessarily a "right" answer or source of information. As a professional you might search the web for past studeis. For example, you can skim the Lagoon Water Quality report on GauchoSpace. For this class however, it might be a good opportunity to first try making your own personal observations. Take some time to go down to the lagoon and walk around. Do some investigating via Google Earth. What do you think are relevant aspects that should be included for a report?

As part of your investigations/observations, you should figure out the location of any storm drains that enter into the Lagoon. You will need to note these locations on your map.

Eutrophication

Put your reading summary in this cell.

- Eutrophic systems are highly productive and stratification of the water column isolates bottom waters, leading to hypoxia. High nutrient loading is what leads to eutrophication and naturally the shift in biological productivity over a long period of time. And for culturally the rapid shift in productivity is due to human related nutrient pollution. Stratification is also important because its what prevents vertical mixing, potentially leading to hypoxia in bottom waters. Since river flow stratifies estuarine waters, rivers bring nutrients from the watershed to the continental shelf.

Campus Lagoon

Put your site overview in this cell.

- The campus lagoon is constantly receiving a influx of sea water which is pumped in from a pumping facility also the lagoon doesn't have a tidal influence. Also the lagoon is carries mainly salt marsh species. The physical dimensions of the lagoon is 31 acres and the local climate of the lagoon and surrounding areas is tipically warmer during the spring and summer months and cooler during the fall and winter months. I don't think wind matters for the lagoon since there is no tidal influence.

Methods

> Normally in a report like this you might describe your methods. You don't have to write anything here, but it might helpful to understand a little bit about where the data files associated with this assignment come from.
>
> The YSI 2030 Pro is a hand-held sensor and data logger. It has a set of probes at one end for measuring the consituents noted in the introduction. The probes are connected via a cable to a hand-held unit that shows the measurments and is capable of logging the readings. The data were collected in Winter of 2019 by walking to a number of locations around the lagoon. At each "station" we put the probe in the water, allowed the measurement to stabilize, and then recorded the values on a log sheet.
>
> The MiniDOT is a self-contained, submersible logger. It was deployed in the lagoon for two weeks. It was attached to a *mooring*: a small weight and float are connected by a line. The weight keeps the mooring in place; the float keeps the line upright through the water. The sensor (i.e. the MiniDOT) is attached to the line so it is suspended in the water column. After recovering the mooring and cleaning off the MiniDOT, we connect it to a computer and download the data file (creatively called `minidot_data.csv`) provided with the assignment.

Campus Lagoon dry season monitoring

> Make plots showing the spatial distribution of temperature, salinity, and dissolved oxygen in the lagoon derived from the YSI 2030 Pro. These data are recorded in the PDF sampling sheet (`191023_lagoon_sampling_datasheet_with_data.pdf`). You will need to create a csv text file that can be imported to Python. Then do the following:
>
> - For each parameter, plot the surface measurement from each station on a map. The value should be shown in color, using a scatterplot. There should be three subplot or subpanels, arranged in one row with three columns, such that there is a subplot for each variable. An example of a single figure is provided on GauchoSpace. Call this Figure 1.
> - For each parameter, plot the surface and bottom (where available) measurements as a function of distance along the axis of the lagoon, using different line types for the surface and bottom. There should be one figure with three panels, with each panel containing two lines/set of points. An example is provided on GauchoSpace. Call this Figure 2.
> - Compare the timeseries of dissolved oxygen collected with the MiniDOT logger to weather conditions from the airport. Call this figure 3. Again, an example is provided on GauchoSpace.

In [42]:

```
import matplotlib.pyplot as plt    # basic plotting functions
from mpl_toolkits.axes_grid1.inset_locator import inset_axes
import pandas as pd                # dataframes (sort of like a python spre
import numpy as np                 # math functions (e.g. sin(), cos())
```

In [43]:
```
def distance_in_km(Latitude0, Longitude0, Latitude1, Longitude1):
    R = 6371; # Radius of the Earth in km

    phi = (Latitude0 + Latitude1)*0.5; # The average latitude
    dphi = Latitude0 - Latitude1;    # Change in latitude
    dtheta = Longitude0 - Longitude1;  # Change in longitude

    # Distance between two points on the Earth
    # Result is in km because radius of Earth is specified in km
    dy = R*np.sin(dphi/180*np.pi);                          # Change in y di
    dx = R*np.cos(phi/180*np.pi)*np.sin(dtheta/180*np.pi); # Change in x di
    dist_in_km = np.sqrt(dx*dx + dy*dy);                    # Via pythagore

    return dist_in_km
```

In [44]:
```
distance_in_km(34.40778, -119.85028, 34.41139, -119.85028)
```

Out[44]:
```
0.40141368492068796
```

In [45]:
```
# Place code here for reading in the necessary data.
# ("Reading in" means importing the contents
# of a textfile into memory within Python.)
# Again, use the examples on GauchoSpace.
# You can use as many cells as you'd like

# Read a file containing the measurements
df = pd.read_csv('191023_lagoon_sampling_datasheet_with_data.csv', sep=',',

# We now have a dataframe called "df".
# df contains fields that correspond to the columns of our CSV file (think
# We can display this data frame by simply typing its name with nothing els
df
```

Out[45]:

	Station_name_description	Time	Latitude	Longitude	Max_Depth	Temperature_C_S
0	Site_1_Lagoon_Outflow	12:56	34.40778	-119.85028	20	
1	Site_2_Floating_Dock	13:17	34.41139	-119.85028	120	
2	Site_3_Stormdrain_behind_Ucen	13:28	34.41111	-119.84861	85	
3	Site_4_Eastside_Park	13:45	34.40833	-119.84389	45	

In [46]:
```
df['dist_bw_stations'] = 0.0
df['total_distance'] = 0.0

df
```

Out[46]:

	Station_name_description	Time	Latitude	Longitude	Max_Depth	Temperature_C_S
0	Site_1_Lagoon_Outflow	12:56	34.40778	-119.85028	20	
1	Site_2_Floating_Dock	13:17	34.41139	-119.85028	120	
2	Site_3_Stormdrain_behind_Ucen	13:28	34.41111	-119.84861	85	
3	Site_4_Eastside_Park	13:45	34.40833	-119.84389	45	

In [47]:

```
# Now we will use a for loop to apply our function to multiple rows of our
# Note that we skip the first location (where the index equals 0)
for n in range(1, len(df.index)):

    # Distance between two points on the Earth, using the function defined
    dist_in_km = distance_in_km(df.Latitude[n-1], df.Longitude[n-1], df.Lat

    # PUT THE RESULTS OF THE CALCULATION into the dataframe
    # this line says: put the value stored as "dist_in_km" into [row, colum
    df.at[n,'dist_bw_stations'] = dist_in_km
    df.at[n,'total_distance'] = df.total_distance[n-1] + dist_in_km

# Inspect the dataframe to see the new columns
df
```

Out[47]:

	Station_name_description	Time	Latitude	Longitude	Max_Depth	Temperature_C_S
0	Site_1_Lagoon_Outflow	12:56	34.40778	-119.85028	20	
1	Site_2_Floating_Dock	13:17	34.41139	-119.85028	120	
2	Site_3_Stormdrain_behind_Ucen	13:28	34.41111	-119.84861	85	
3	Site_4_Eastside_Park	13:45	34.40833	-119.84389	45	

In [48]:

```
df_FL_drains = pd.read_csv('storm_drains.csv', sep=',', header=[0], na_valu

df_SL_shoreline = pd.read_csv('shoreline.csv', sep=',', header=[0], na_valu
```

In [49]:

```
# We will need to import a few new packages:
from datetime import datetime    # For analyzing time series data
import requests       # For grabbing data via API (application programming
import json           # For parsing strings in json format (file format use
```

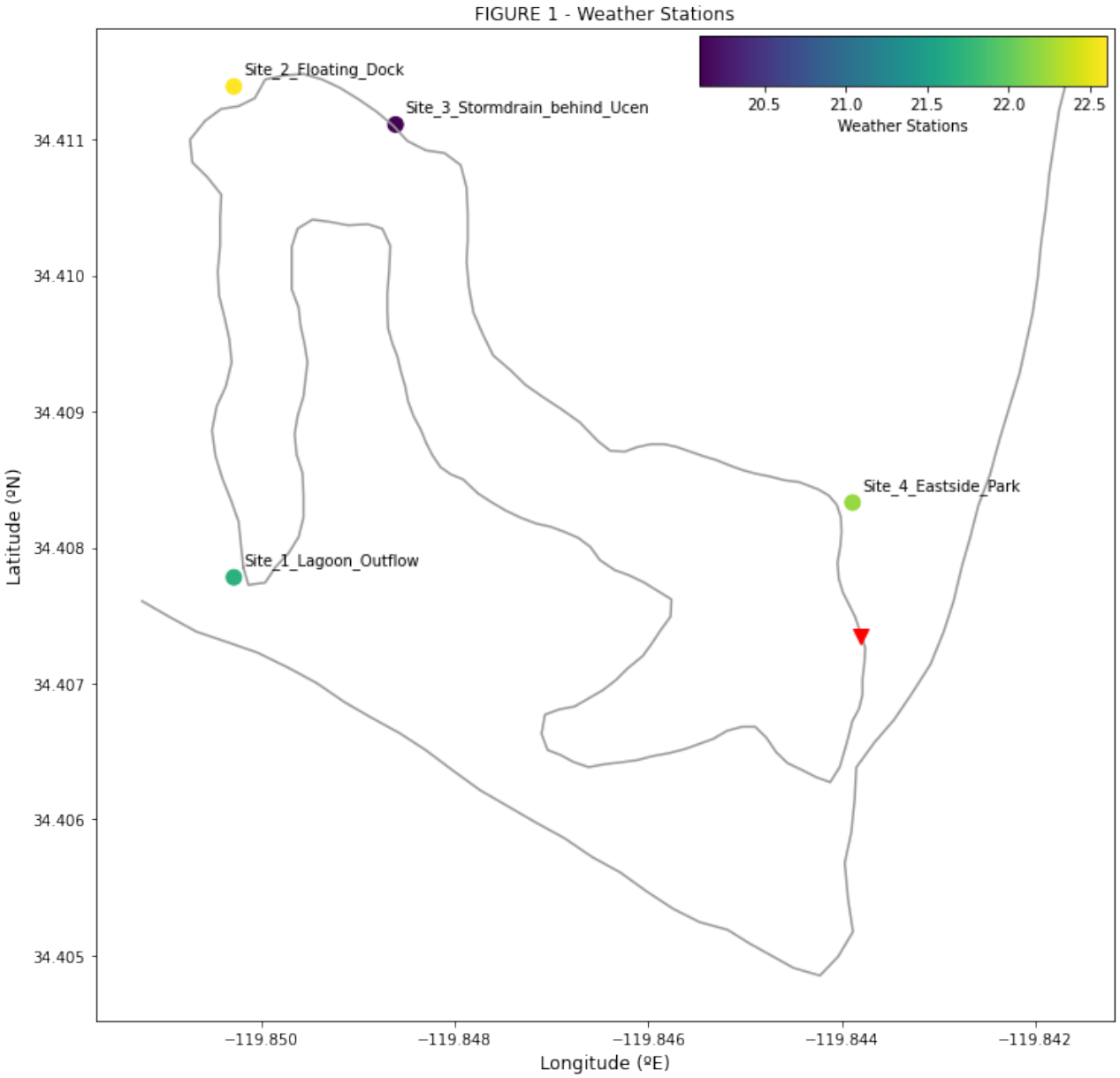
In [66]:
```
# Put code here to get data from the Santa Barbara Airport

# The following code will grab weather data from the Santa Barbara Airport
# for the time period specified within the "start" and "end" points.
# The timestamp is referenced to UTC. Temperature is in ºC.
#https://developers.synopticdata.com/mesonet/v2/stations/timeseries/

parameters = {'token': 'f0c7febd7f634e09a2de8b3a16119db6',
              'stid':'ksba',
              'start':'201910100000',
              'end':'201910240000',
              'obtimezone':'local',
              'vars':'wind_speed,air_temp',
              'output':'json'}
response = requests.get('https://api.synopticdata.com/v2/stations/timeserie
            params=parameters)
data = response.json()  # parse out json structure

# Let's take a look at this raw data that we pulled from the website:
```

In [67]:
```
# We don't want our data in that format, so instead we make an empty data f
# and pull out the useful info

df_ksba = pd.DataFrame() # Create empty dataframe
df_ksba['timestamp'] = datetime.now() # Initialize timestamp field (or colu

obs_time = data['STATION'][0]['OBSERVATIONS']['date_time']
# Loop through each line and convert the string into a "datetime" value
for n in range(0, len(obs_time)):
    df_ksba.at[n,'timestamp'] = datetime.strptime(obs_time[n][0:-5], "%Y-%m

# Add the data from the wind speed and the air temperature
df_ksba['windspd'] = data['STATION'][0]['OBSERVATIONS']['wind_speed_set_1']
df_ksba['airtemp'] = data['STATION'][0]['OBSERVATIONS']['air_temp_set_1']

# Check out the table we made
```

In [68]:
```
# Put your code here to LOAD the MiniDOT data (found in minidot_data.csv),
df_minidot = pd.read_csv('minidot_data.csv', sep=',', header=[0], na_values
```

In [69]:

```
# FIGURE 1. A map!
# Create three color-coded scatter plots, overlaid on the shoreline points
# Be sure to include the storm drain locations.

fig = plt.figure(figsize=(12,12))
ax = fig.add_subplot(1,1,1)


# Plot surface temperature as a scatter plot on the map.
# scatter(X, Y, SIZE, COLOR)
pts = plt.scatter(
        df.Longitude,
        df.Latitude,
        100,df.Temperature_C_Surface)

# Annotate the points with the station names
for n, txt in enumerate(df.Station_name_description):
    ax.annotate(txt, (df.Longitude[n]+0.0001, df.Latitude[n]+0.0001))


# Shoreline points
ax.plot(df_SL_shoreline.longitude, df_SL_shoreline.latitude,
        '-', color=('0.6')) # solid gray line, no symbol
# See https://matplotlib.org/2.1.1/api/_as_gen/matplotlib.pyplot.plot.html

# storm drains
ax.plot(df_FL_drains.longitude, df_FL_drains.latitude, 'rv', markersize = 1

        # Red triangle point is for Storm drains


# Add some labels
ax.set_ylabel("Latitude (ºN)", fontsize=12)   # Set the y axis label
ax.set_xlabel("Longitude (ºE)", fontsize=12)  # x axis label
ax.set_title("FIGURE 1 - Weather Stations", fontsize=12)    # TITLE

# Add a colorbar
axins = inset_axes(ax, width="40%", height="5%", loc='upper right')
fig.colorbar(pts, cax=axins, orientation="horizontal", label="Weather Stati

ax.ticklabel_format(useOffset = False)
```



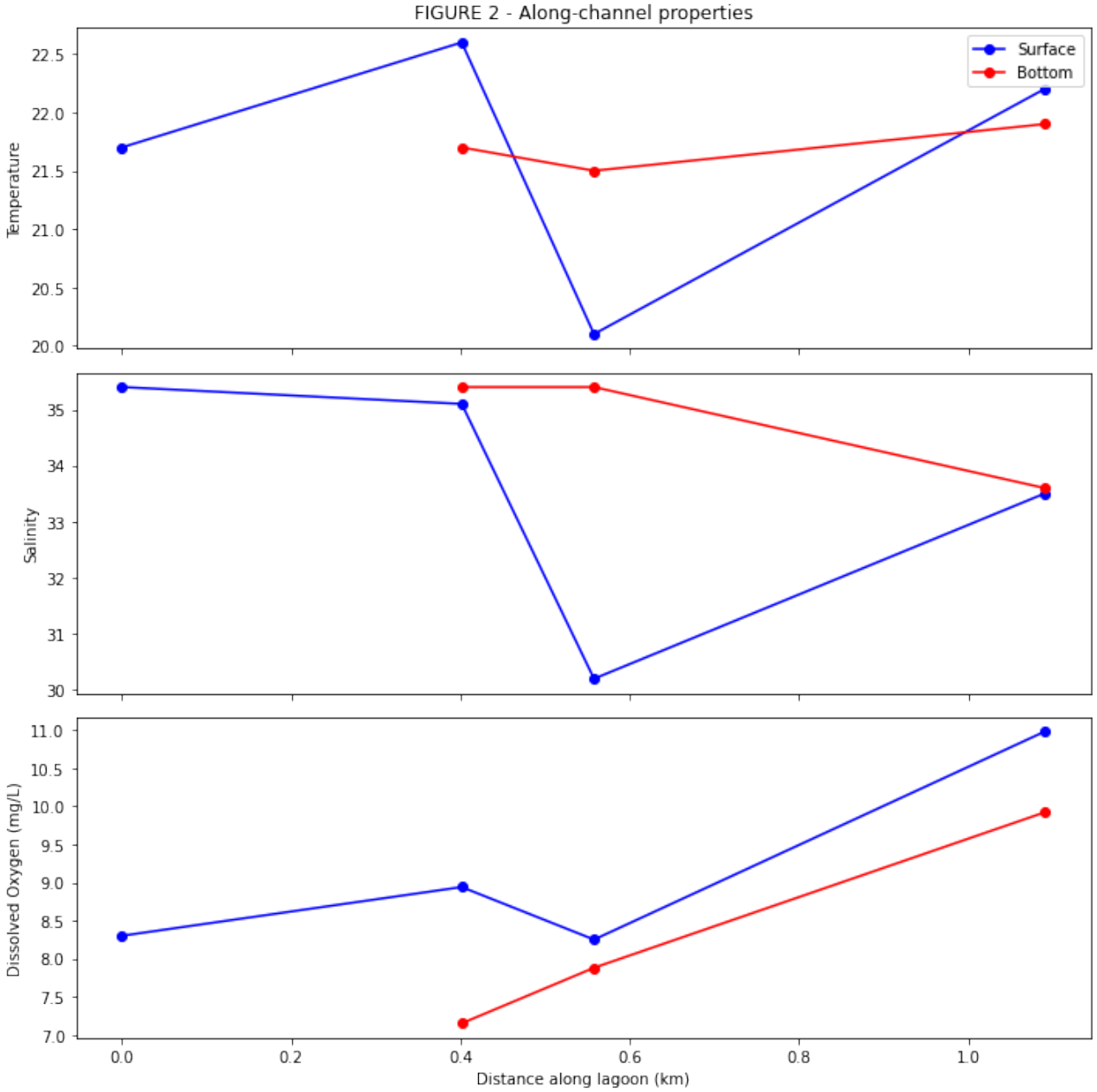
In [54]:

```
# Figure 2. Spatial plots
# Make a plot of water quality parameters along the axis of the estuary
# You should have three subplots, stacked on top of one another.
# -- Top subplot(3,1,1): Temperature at the surface and bottom
# -- Middle subplot(3,1,2): Salinity at the surface and bottom
# -- Bottom subplot(3,1,3): Dissolved oxygen at the surface and bottom
# Only label the horizontal (distance) axis on the bottom plot, but be sure
# Also, be sure that you use the same color coding for the surface and bott
```

In [55]:

```
fig, (ax1, ax2, ax3) = plt.subplots(3, 1, sharex=True, figsize=(10,10))
                                # number of rows, number of columns

# Plot the surface data into the first subplot
ax1.plot(df.total_distance,
```

```
        df.Temperature_C_Surface,
        'b-o',
        label="Surface")  # b-o = blue circle data point connected by a lin

# Plot the bottom data into the first subplot. Use a different color
ax1.plot(df.total_distance,
        df.Temperature_C_Bottom,
        'r-o',
        label="Bottom")

# Add some labels and a legend
# Note that we only want to set the xlabel on the last plot in the set of s
# We can also get away with using a single legend, because all of the plots
# show similar information
ax1.set_ylabel("Temperature");   # Set the y axis label
ax1.legend()
ax1.set_title("FIGURE 2 - Along-channel properties")


ax2.plot(df.total_distance,
        df.Salinity_Surface,
        'b-o',
        label="Surface")

ax2.plot(df.total_distance,
        df.Salinity_Bottom,
        'r-o',
        label="Bottom")

ax2.set_ylabel("Salinity");   # Set the y axis label

#
ax3.plot(df.total_distance,
        df.Dissolved_Oxygen_Surface,
        'b-o',
        label="Surface")  # b-o = blue circle data point connected by a lin

# Plot the bottom data into the first subplot. Use a different color
ax3.plot(df.total_distance,
        df.Dissolved_Oxygen_Bottom,
        'r-o',
        label="Bottom")

ax3.set_ylabel("Dissolved Oxygen (mg/L)");   # Set the y axis label
ax3.set_xlabel("Distance along lagoon (km)");  # x axis label


fig.tight_layout()
```



In [56]:
```
# Figure 3. Time series!
# Make a timeseries plot that includes two subplots, arranged on on top of
# -- Top subplot: Plot Airport air temperature and lagoon water temperature
# -- Bottom subplot: Plot airport wind speed and lagoon dissolved oxygen
# Be sure that both subplots have the same bounds for the horizontal (time)
```

In [57]:
```
df_minidot['datetime_local'] = pd.to_datetime(df_minidot['datetime_local'])

df_minidot.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20189 entries, 0 to 20188
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype         
---  ------           --------------  -----         
 0   datetime_local   20189 non-null  datetime64[ns]
 1   water_temp_degC  20189 non-null  float64       
 2   do_mgL           20189 non-null  float64       
 3   batt_volt        20189 non-null  float64       
dtypes: datetime64[ns](1), float64(3)
memory usage: 631.0 KB
```

In [58]:
```
df_minidot.rename(columns={'datetime_local': 'timestamp'}, inplace=True)
```

In [59]:
```
ksba_minidot_merge = pd.merge(df_ksba,df_minidot, on = 'timestamp')
```

In [60]:
```
ksba_minidot_merge
```

Out[60]:

	timestamp	windspd	airtemp	water_temp_degC	do_mgL	batt_volt
0	2019-10-09 17:00:00	3.09	18.0	22.85	10.65	3.20
1	2019-10-09 17:05:00	3.09	18.0	22.87	10.75	3.20
2	2019-10-09 17:10:00	2.57	19.0	22.78	10.80	3.20
3	2019-10-09 17:15:00	3.09	18.0	22.80	10.79	3.20
4	2019-10-09 17:20:00	3.60	18.0	22.78	10.84	3.20
...	...	...	...	...	...	...
4316	2019-10-23 12:55:00	1.54	28.0	21.93	12.48	3.14
4317	2019-10-23 13:00:00	2.06	28.0	21.98	12.62	3.14
4318	2019-10-23 13:05:00	0.00	28.0	22.00	12.65	3.14
4319	2019-10-23 13:10:00	1.54	28.0	21.99	12.02	3.14
4320	2019-10-23 13:15:00	2.57	29.0	22.08	11.60	3.14

4321 rows × 6 columns

In [61]:
```
ksba_minidot_merge['timestamp'] =  pd.to_datetime(ksba_minidot_merge['times
```

In [62]:
```
ksba_minidot_merge.timestamp
```

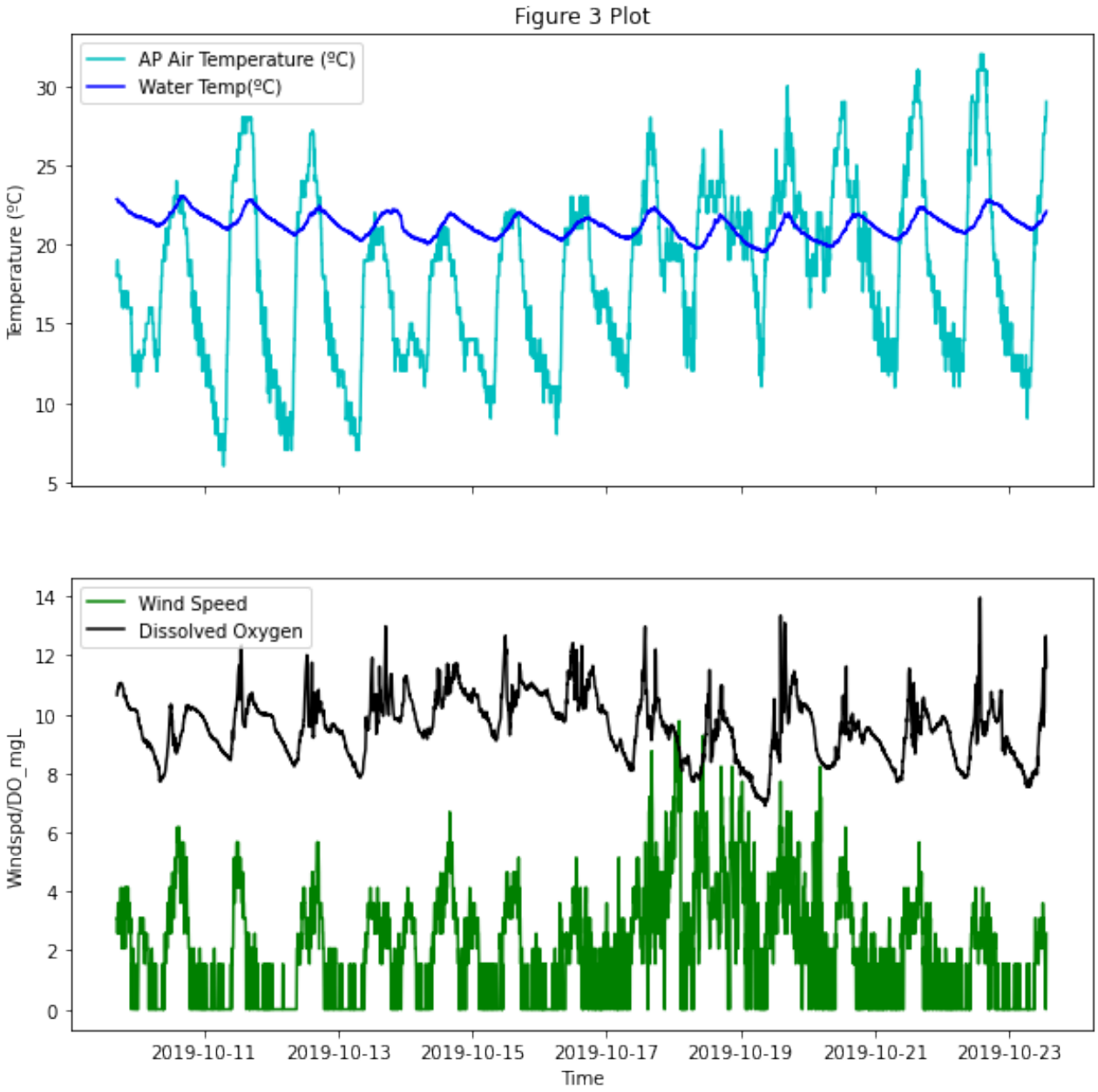
Out[62]:
```
0       2019-10-09 17:00:00
1       2019-10-09 17:05:00
2       2019-10-09 17:10:00
3       2019-10-09 17:15:00
4       2019-10-09 17:20:00
                ...
4316    2019-10-23 12:55:00
4317    2019-10-23 13:00:00
4318    2019-10-23 13:05:00
4319    2019-10-23 13:10:00
4320    2019-10-23 13:15:00
Name: timestamp, Length: 4321, dtype: datetime64[ns]
```

In [63]:
```
ksba_minidot_merge.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4321 entries, 0 to 4320
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype         
---  ------           --------------  -----         
 0   timestamp        4321 non-null   datetime64[ns]
 1   windspd          4309 non-null   float64       
 2   airtemp          4321 non-null   float64       
 3   water_temp_degC  4321 non-null   float64       
 4   do_mgL           4321 non-null   float64       
 5   batt_volt        4321 non-null   float64       
dtypes: datetime64[ns](1), float64(5)
memory usage: 236.3 KB
```

In [65]:

```
fig, (ax1, ax2) = plt.subplots(2, 1, sharex=True, figsize=(10,10))

ax1.plot(ksba_minidot_merge.timestamp,
         ksba_minidot_merge.airtemp,
        'c', label = "AP Air Temperature (ºC)")
ax1.plot(ksba_minidot_merge.timestamp,
         ksba_minidot_merge.water_temp_degC,
        'blue', label = "Water Temp(ºC)")
ax1.set_ylabel("Temperature (ºC)");
ax1.set_title("Figure 3 Plot")
ax1.legend()


ax2.plot(ksba_minidot_merge.timestamp,
         ksba_minidot_merge.windspd,
        'g-',
        label="Wind Speed")
ax2.plot(ksba_minidot_merge.timestamp,
         ksba_minidot_merge.do_mgL,
        'k-',
        label="Dissolved Oxygen")

ax2.set_ylabel("Windspd/DO_mgL");
ax2.set_xlabel("Time")
ax2.legend()

ax.ticklabel_format(useOffset = False)
```



Discussion of results

> Describe the observations you made, speculating about the possible reasons for the observed patterns in temperature, salinity, and dissolved oxygen. How/why are the patterns between these three variables similar or different spatially across the lagoon? Do conditions in the lagoon appear to be affected by the weather conditions? What other information might be useful to help interpret the timeseries?

- The patterns I observed in the 3 subplots for temperature, salintiy, and dissolved oxygen is for temperature the surface increased then fell off(decreased) then increased again towards the end and the bottom temperture seemed to start off at 0.4 km and continue to increase as the distance goes on. The salinity surface started of with a lot of salintiy and then dropped off at 0.4 km then increased in salinity from 0.5-1.0. The salinity bottom started off at 0.4 km with high amount of salinity and then decreases slowly. Lastly the dissolved oxygen increased then had a decrease in dissolved oxygen then once again increased and the bottom dissolved oxygen started off at 0.4 km and only increased from there. These 3 different observed patterns have similar traits in terms of the way the data is plotted and I do think that conditions in the lagoon appear to be affected by weather condtions such as if some days are extremely hot then the water will warm and not hold a lot of oxygen also the temperature of the water would be affected by this.

Sampling recommendations

> Our sampling efforts were clearly limited by resources (what we could measure) and access (where/how we could measure). Provide a recommendation for where, how, and when you should conduct a sampling campaign to properly characterize water quality in the lagoon during both dry and wet seasons. For example, would you change the spacing of the stations? When/where you measured? What you measured?

- A recommendation we should use when conducting a sample campaign to properly charaterize water quality in the lagoon during both dry and wet seasons is to space out the stations futher away from each other to get a better collection of data that way the stations are far enough apart for one another that they cover a great amount of the lagoon. Also measuring right after it rained and during also dry days would be good for data collection since you can compare the rainy(after) days with the dry days and note differences and I would take masurements once a month for a year to compare the sampling data each month.