Data Analysis using Python 3

Data Analyst is now a very lucrative job handle to have in the tech world, with the skills and patience of a data analyst one can analyze large chunks of Data in very short span of time which would have been otherwise impossible or would take up a long time.

Early January this year I did PH526x: Using Python for Research – (Harvard University) via edX, which taught me how to analyze .csv data files using Python 3.I scored well enough and passed the course.So, here I am presenting you, what I learned.

It was fun and exciting to work throughout the course and learn new stuff.My repository – Github_repo holds the codes for study cases I had in the course.Professor Jukka-Pekka "JP" Onnela is an excellent professor who would make you learn stuff with his excellent teaching style.It was an awesome experience.

So, Let's analyze one of the case called: Bird Migration Analysis repo

***Aim*:** Track the movement of three gulls namely – Eric, Nico & Sanne

Dataset: https://inbo.carto.com/u/lifewatch/datasets ; used dataset – csv

Summary: One fascinating area of research uses GPS to track movements of animals. It is now possible to manufacture a small GPS device that is solar charged, so you don't need to change batteries and use it to track flight patterns of birds. The

data for this case study comes from the LifeWatch INBO project. Several data sets have been released as part of this project. We will use a small data set that consists of migration data for three gulls named Eric, Nico, and Sanne. The csv file contains eight columns and includes variables like latitude, longitude, altitude, and time stamps. In this case study, we will first load the data, visualize some simple flight trajectories, track flight speed, learn about daytime and much, much more.

Dependencies :

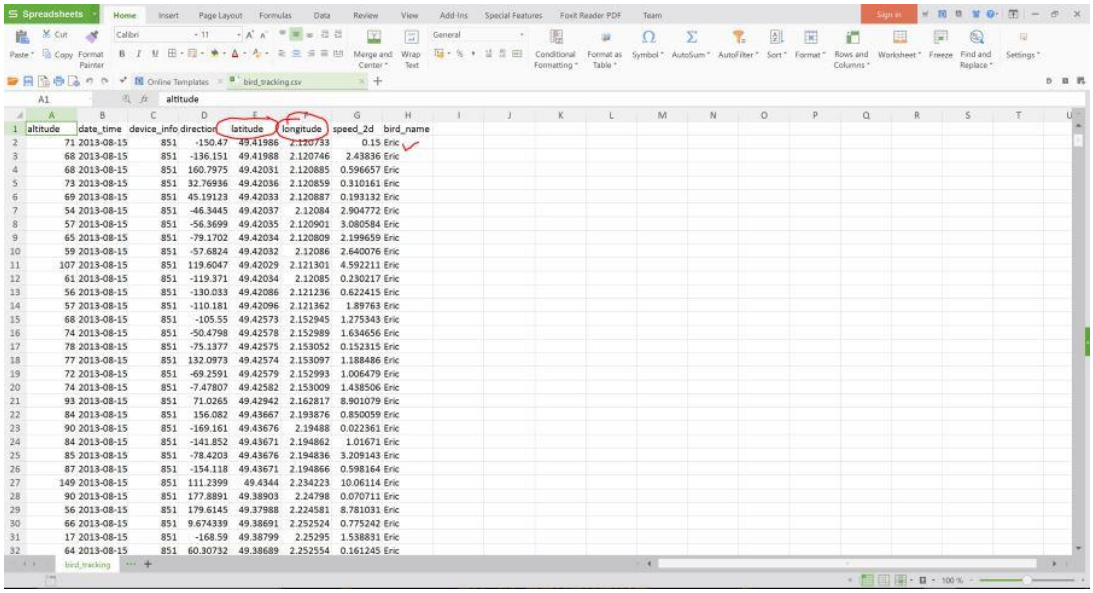
- Matplotlib
- Pandas
- Numpy
- Cartopy
- Shapely

We will divide our case study into five parts:

1. Latitude and Longitude
2. 2D speed vs. Frequency
3. Time and Date
4. Daily Mean Speed
5. Cartographic View

PART (1/5): Latitude and Longitude

In this part, we are going to visualize the location of the birds.We are going to plot latitude and longitude along y and x-axis respectively and visualize the data present in the csv file.



altitude	date_time	device_info	direction	latitude	longitude	speed_2d	bird_name
71	2013-08-15	851	-150.47	49.41986	2.120733	0.15	Eric
68	2013-08-15	851	-136.151	49.41988	2.120746	2.43836	Eric
68	2013-08-15	851	160.7975	49.42031	2.120885	0.596657	Eric
73	2013-08-15	851	32.76936	49.42036	2.120859	0.310161	Eric
69	2013-08-15	851	45.19123	49.42033	2.120887	0.193132	Eric
54	2013-08-15	851	-46.3445	49.42037	2.12084	2.904772	Eric
57	2013-08-15	851	-56.3699	49.42035	2.120901	3.080584	Eric
65	2013-08-15	851	-79.1702	49.42034	2.120809	2.199659	Eric
59	2013-08-15	851	-57.6824	49.42032	2.12086	2.640076	Eric
107	2013-08-15	851	119.6047	49.42029	2.121301	4.592211	Eric
61	2013-08-15	851	-119.371	49.42034	2.12085	0.230217	Eric
56	2013-08-15	851	-130.033	49.42086	2.121236	0.622415	Eric
57	2013-08-15	851	-110.181	49.42096	2.121362	1.89763	Eric
68	2013-08-15	851	-105.55	49.42573	2.152945	1.275343	Eric
74	2013-08-15	851	-50.4798	49.42578	2.152989	1.634656	Eric
78	2013-08-15	851	-75.1377	49.42575	2.153052	0.152315	Eric
77	2013-08-15	851	132.0973	49.42574	2.153097	1.188486	Eric
72	2013-08-15	851	-69.2591	49.42579	2.152993	1.006479	Eric
74	2013-08-15	851	-7.47807	49.42582	2.153009	1.438506	Eric
93	2013-08-15	851	71.0265	49.42942	2.162817	8.901079	Eric
84	2013-08-15	851	156.082	49.43667	2.193876	0.850059	Eric
90	2013-08-15	851	-169.161	49.43676	2.19488	0.022361	Eric
84	2013-08-15	851	-141.852	49.43671	2.194862	1.01671	Eric
85	2013-08-15	851	-78.4203	49.43676	2.194836	3.209143	Eric
87	2013-08-15	851	-154.118	49.43671	2.194866	0.598164	Eric
149	2013-08-15	851	111.2399	49.4344	2.234223	10.06114	Eric
90	2013-08-15	851	177.8891	49.38903	2.24798	0.070711	Eric
56	2013-08-15	851	179.6145	49.37988	2.224581	8.781031	Eric
66	2013-08-15	851	9.674339	49.38691	2.252524	0.775242	Eric
17	2013-08-15	851	-168.59	49.38799	2.25295	1.538831	Eric
64	2013-08-15	851	60.30732	49.38689	2.252554	0.161245	Eric

The code:

bird_migration_trajectories_lat.long.py

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

birddata = pd.read_csv("bird_tracking.csv") #make sure,you are in the right directory , check (>>>pwd)

# >>>birddata.info() #look at basic info abot the data frame
# >>>birddata.head() #look for first 5 rows in the data set
# >>>birddata.tail() #look for last 5 rows in the data set

bird_names = pd.unique(birddata.bird_name) #look at the unique names of the birds in the csv_file
# >>>print(bird_name)

ix = birddata.bird_name == "Eric" #storing the indices of the bird Eric
x,y = birddata.longitude[ix], birddata.latitude[ix]
plt.figure(figsize = (7,7))
plt.plot(x,y,"b.")
# >>>plt.show() #if you want to check trajectory of "Eric" only

''' To look at all the birds trajectories, we plot each bird in the same plot '''
plt.figure(figsize = (7,7))
for bird_name in bird_names:
    ix = birddata.bird_name == bird_name  #storing the indices of the bird Eric
    x,y = birddata.longitude[ix], birddata.latitude[ix]
    plt.plot(x,y,".", label=bird_name)
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.legend(loc="lower right")
plt.show()

```

In the code, Firstly we import the modules – pandas,matplotlib, and numpy. Then we import the csv file from the default directory (check default directory, >>>pwd or else change the path to the directory holding the csv file using >>>cd directory_address) into the variable birddata.

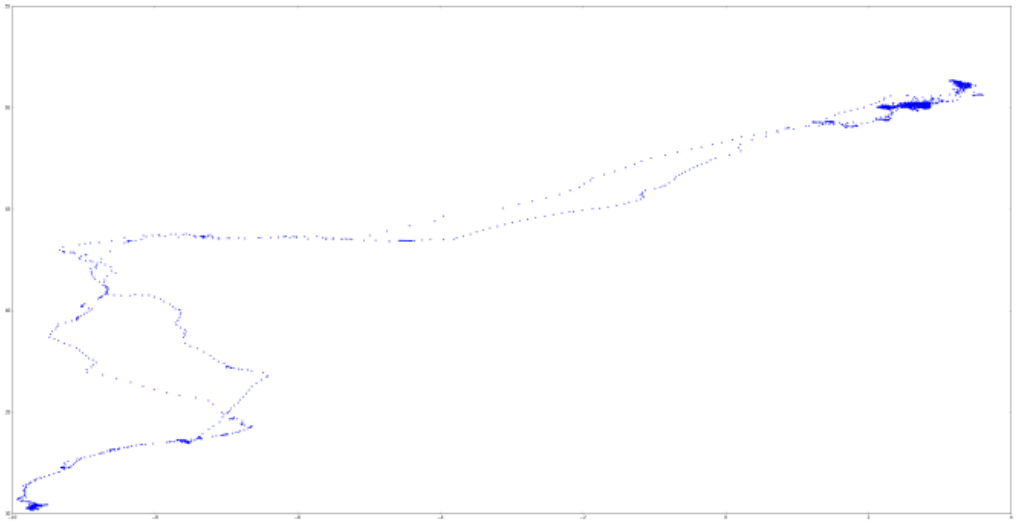
♦bird_names = pd.unique(birddata.bird_name) is used to find all the unique bird names from the csv file and save it to the variable bird_names using pandas dataframe and unique() function.

Next, we are going to select the latitude and longitude data only for the gull named “Eric”.We code,

♦ ix = birddata.bird_name == “Eric”
x,y = birddata.longitude[ix], birddata.latitude[ix]
plt.figure(figsize = (7,7))
plt.plot(x,y,”b.”)

Here, we are specifying the variable ix to contain the data of the column named “bird_name” from the csv file having the name of the bird as “Eric”.Next, we are specifying x to hold longitude data and y to hold latitude data of “Eric”.We use the matplotlib function, figure() to initialize it’s size as 7 x 7 and plot it using the plot() function(learn matplotlib).The parameters inside the function plot() i.e x, y and “b.” are specifying to use longitude data along x axis, latitude along y and b=blue, . = circles in the vizualization.

Output: enlarged_view



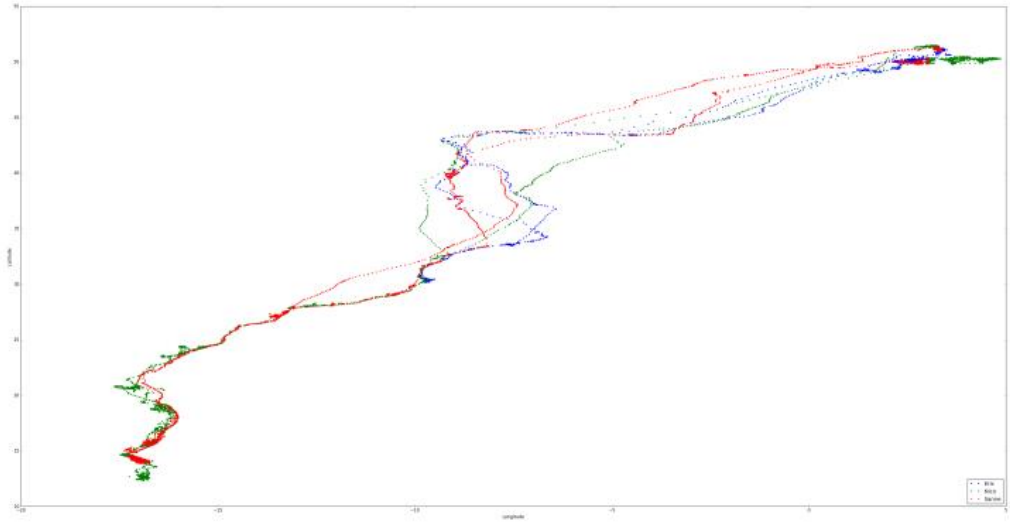
But Now, to look at all the bird’s trajectories, we plot each bird in the same figure.We code,

```
♦ plt.figure(figsize = (7,7))
  for bird_name in bird_names:
    ix = birddata.bird_name == bird_name
    x,y = birddata.longitude[ix], birddata.latitude[ix]
  plt.plot(x,y,”.”, label=bird_name)
  plt.xlabel(“Longitude”)
  plt.ylabel(“Latitude”)
  plt.legend(loc=”lower right”)
  plt.show()
```

Here, we are plotting the location for all the three gulls namely Eric, Nico and Sanne.We create a 7 x 7 figure using plt.plot(figsize = (7,7)).We store unique bird’s name in the variable ix, longitude and latitude data in the variales x and y repectively and we over every data using a for loop.Next we, plot the data x and y using “.” = circular marks and we add a label named “bird_data”.We also use labels Longitude and Latitude along x and y axis respectively using xlabel() and ylabel() functions.legend() is used to locate the info bar in the plot, which is initialized to

lower right.Finally, we use the show() function to get the visualized data for all the three gulls.

Output : enlarged_view



PART (2/5):_2D Speed Vs Frequency

In this second part of the case study, we are going to visualize 2D speed Vs Frequency for the gull named “Eric”.

The Code:

bird_migration_speed.py

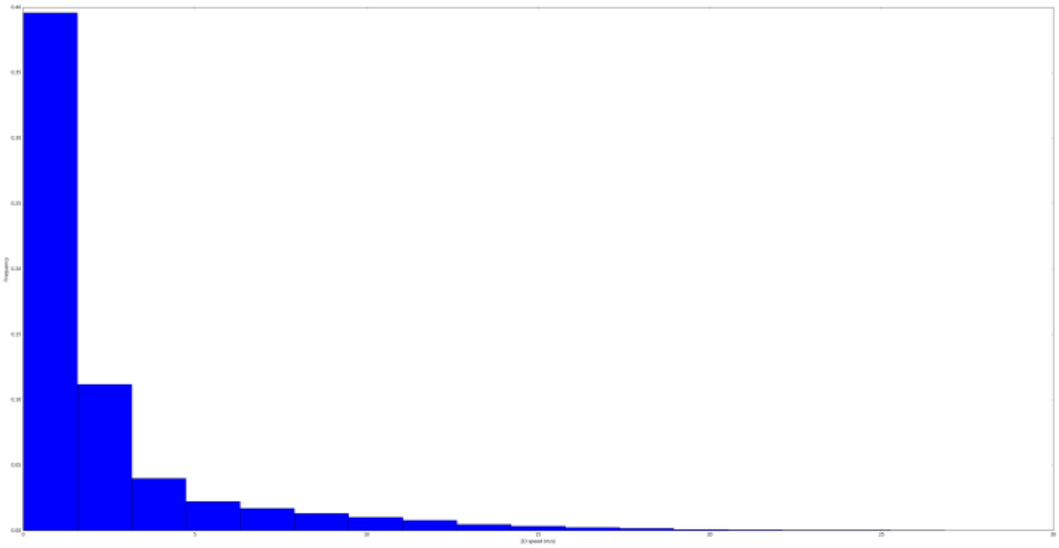
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

ix = birddata.bird_name == "Eric" #storing the indices of the bird Eric
speed = birddata.speed_2d[ix]
# >>>plt.hist(speed) #[F.A.I.L.S] ,since we have non numeric numbers in the speed array
# >>>plt.hist(speed[:10])  #plot a histogram using the first 10 observations of speed [works]
# >>>np.isnan(speed) #>>>np.isnan(speed).any()  #I am looking for any non number objects in the speed column using numpy's isnan function.
# >>>np.sum(np.isnan(speed)) # I find out the count of non numeric entries, False=0 & True =1 from isnan()

# >>>ind = np.isnan(speed) #return true if entry is not numerical or (not a NaN)
# >>>plt.hist(speed[~ind]) #we will include only those entries for which ind != True

plt.figure(figsize = (8,4))
# >>>speed = birddata.speed_2d[birddata.bird_name == "Eric"] #step 5 & 6 combined
ind = np.isnan(speed)
plt.hist(speed[~ind], bins=np.linspace(0,30,20), normed=True)
plt.xlabel(" 2D speed (m/s) ")
plt.ylabel(" Frequency ")
plt.show()


'''
    We can also plot a similar histogram using the pandas module instead of pyplot.
    The benefit of using pandas is that we do not have to deal with NaNs explicitly.
    Instead, all of that happens under the hood.

    NaNs - Not-a-Number

    >>>birddata.speed_2d.plot(kind='hist', range=[0,30])
    >>>plt.xlabel("2D speed")
    >>>plt.savefig("hist_birdmig_speed.pdf")
'''

```

♦ ix = birddata.bird_name == “Eric”
speed = birddata.speed_2d[ix]
Here, we load bird data for the gull “Eric” into the variable ix and speed data of the same gull “Eric” into the variable speed.

♦ plt.figure(figsize = (8,4))
ind = np.isnan(speed)
plt.hist(speed[~ind], bins=np.linspace(0,30,20), normed=True)
plt.xlabel(” 2D speed (m/s) “)
plt.ylabel(” Frequency “)
plt.show()

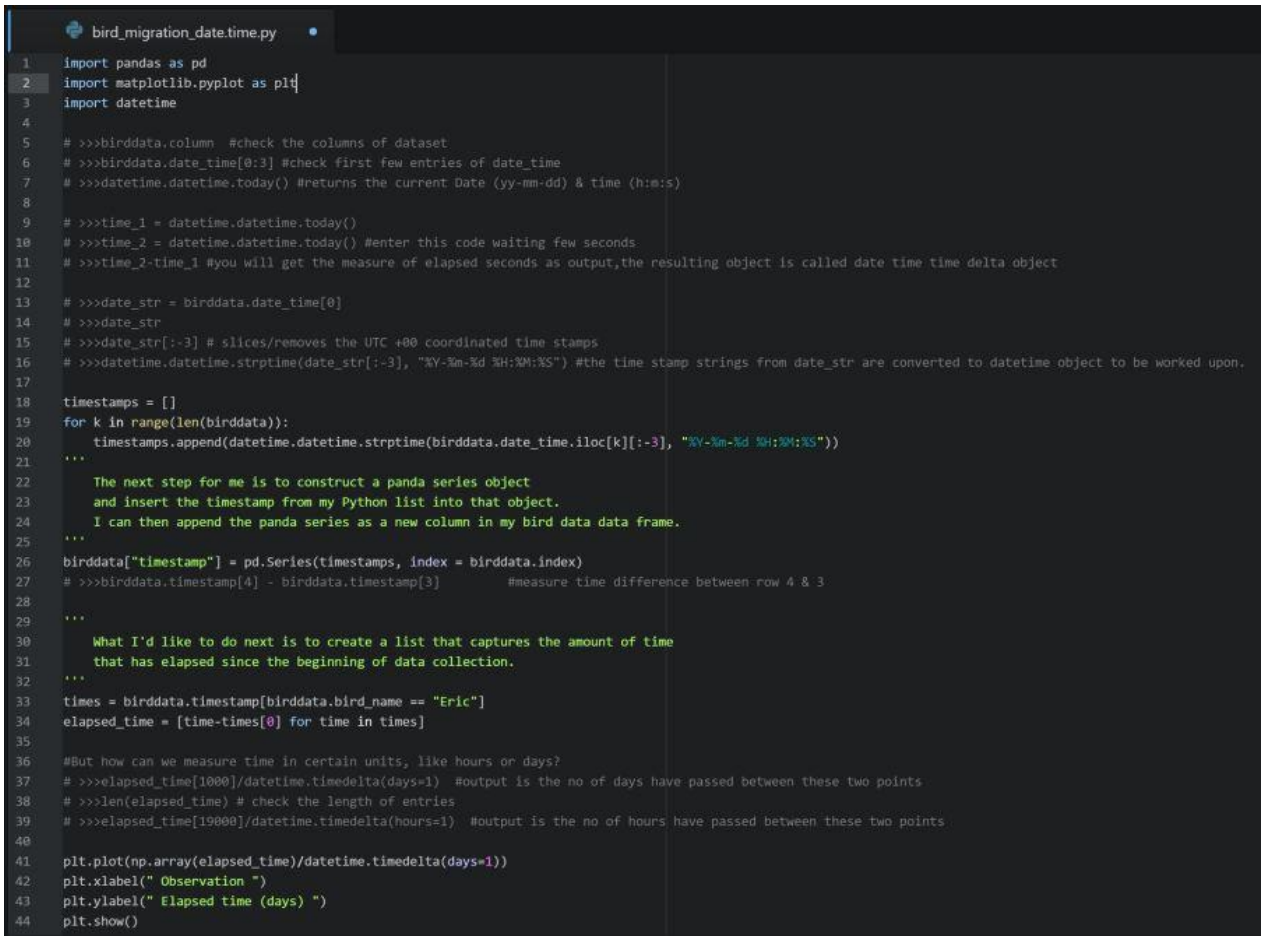
We plot a 8 x 4 figure and allot isnan speed data into ind.We find out the count of non numeric entries, False=0 & True =1 using the isnan() function.Next, we plot a histogram using the hist() function.The parameters speed[~ind] indicates that we will include only those entries for which ind != True, bins=np.linspace(0,30,20) indicates the bins along x axis will vary from 0 to 30 with 20 bins within them linearly spaced.Lastly, we plot 2D speed in m/s along x-axis and Frequency along y-axis using the xlabel() and ylabel() functions respectively and plot the data using plt.show().

Output: enlarged_view



PART (3/5): Time and Date

The third part is associated with date and time.We are going to visualize the time(in days) required by Eric to reach constant distances.If he requires same time to cover almost same distances, then the curve will be linear.



bird_migration_date.time.py

```
import pandas as pd
import matplotlib.pyplot as plt
import datetime

# >>>birddata.column  #check the columns of dataset
# >>>birddata.date_time[0:3] #check first few entries of date_time
# >>>datetime.datetime.today() #returns the current Date (yy-mm-dd) & time (h:m:s)

# >>>time_1 = datetime.datetime.today()
# >>>time_2 = datetime.datetime.today() #enter this code waiting few seconds
# >>>time_2-time_1 #you will get the measure of elapsed seconds as output,the resulting object is called date time time delta object

# >>>date_str = birddata.date_time[0]
# >>>date_str
# >>>date_str[:-3] # slices/removes the UTC +00 coordinated time stamps
# >>>datetime.datetime.strptime(date_str[:-3], "%Y-%m-%d %H:%M:%S") #the time stamp strings from date_str are converted to datetime object to be worked upon.

timestamps = []
for k in range(len(birddata)):
    timestamps.append(datetime.datetime.strptime(birddata.date_time.iloc[k][:-3], "%Y-%m-%d %H:%M:%S"))
'''
    The next step for me is to construct a panda series object
    and insert the timestamp from my Python list into that object.
    I can then append the panda series as a new column in my bird data data frame.
'''
birddata["timestamp"] = pd.Series(timestamps, index = birddata.index)
# >>>birddata.timestamp[4] - birddata.timestamp[3]         #measure time difference between row 4 & 3

'''
    What I'd like to do next is to create a list that captures the amount of time
    that has elapsed since the beginning of data collection.
'''
times = birddata.timestamp[birddata.bird_name == "Eric"]
elapsed_time = [time-times[0] for time in times]

#But how can we measure time in certain units, like hours or days?
# >>>elapsed_time[1000]/datetime.timedelta(days=1)  #output is the no of days have passed between these two points
# >>>len(elapsed_time) # check the length of entries
# >>>elapsed_time[19000]/datetime.timedelta(hours=1)  #output is the no of hours have passed between these two points

plt.plot(np.array(elapsed_time)/datetime.timedelta(days=1))
plt.xlabel(" Observation ")
plt.ylabel(" Elapsed time (days) ")
plt.show()
```

We import the libraries matplotlib, pandas, and datetime.

∇ timestamps = []
for k in range(len(birddata)):

timestamps.append(datetime.datetime.strptime(birddata.date_time.iloc[k][:-3],″ %Y-%m-%d %H:%M:%S″))

we create an empty list called timestamps and append date-time data of the birds to it.

”’ >>>datetime.datetime.today() #returns the current Date (yy-mm-dd) & time (h:m:s).

>>>date_str[:-3] #slices/removes the UTC +00 coordinated time stamps.

>>>datetime.datetime.strptime(date_str[:-3], “%Y-%m-%d %H:%M:%S”) ,the time-stamp strings from date_str are converted to datetime object to be worked upon. “%Y-%m-%d %H:%M:%S” is the Year-Month-Date and Hour-Minute-Second format”’

The next step for us is to construct a panda series object and insert the timestamp from our Python list into that object. We can then append the panda series as a new column in my bird data, data frame.

∇ birddata[“timestamp”] = pd.Series(timestamps, index = birddata.index)

What we’d like to do next is to create a list that captures the amount of time that has elapsed since the beginning of data collection.

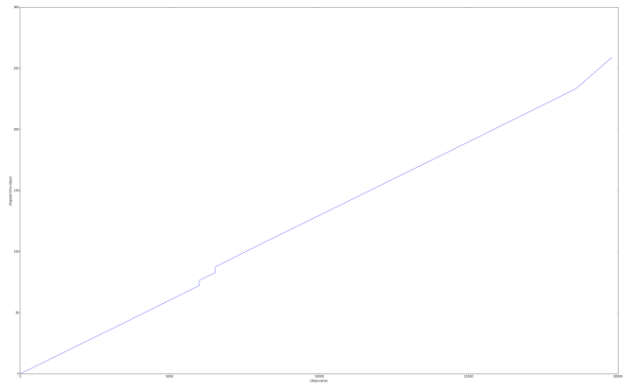
∇ times = birddata.timestamp[birddata.bird_name == “Eric”]
elapsed_time = [time-times[0] for time in times]

we calculated the elapsed time for the gull Eric

∇ plt.plot(np.array(elapsed_time)/datetime.timedelta(days=1))
plt.xlabel(” Observation “)
plt.ylabel(” Elapsed time (days) “)
plt.show()

We plot the observation(reference points at constatnt distances) along x axis vs elapsed time(in days) along y axis.We label our plot using xlabel() and ylabel() as Observation and Elapsed time (days) respectively along x ang y axis.We Observe the curve.

Output : enlarged_view



PART (4/5): Daily Mean Speed

We are going to visualize Daily mean speed of the gull named “Eric” for the total number of days of recorded flight.

bird_migration_daily_mean_speed.py

```
1  import pandas as pd
2  import matplotlib.pyplot as plt
3  import datetime
4  import numpy as np
5
6  #code from bird_migration_date.time ***********************************
7  timestamps = []
8  for k in range(len(birddata)):
9      timestamps.append(datetime.datetime.strptime(birddata.date_time.iloc[k][:-3], "%Y-%m-%d %H:%M:%S"))
10 birddata["timestamp"] = pd.Series(timestamps, index = birddata.index)
11
12 #code from bird_migration_date.time ***********************************
13 data =  birddata[birddata.bird_name == "Eric"]
14 times = data.timestamp
15 elapsed_time = [time-times[0] for time in times]
16 elapsed_days = np.array(elapsed_time)/datetime.timedelta(days=1)
17
18 next_day = 1
19 inds = []
20 daily_mean_speed = []
21 for (i,t) in enumerate(elapsed_days):
22     if t < next_day:
23         inds.append(i)
24     else:
25         daily_mean_speed.append(np.mean(data.speed_2d[inds]))
26         next_day += 1
27         inds = []
28
29 plt.figure(figsize = (8,6))
30 plt.plot(daily_mean_speed, "rs-")
31 plt.xlabel(" Day ")
32 plt.ylabel(" Mean Speed (m/s) ");
33 plt.show()
34
```

Up to line 16, we borrowed the code from part (3/5).

Next, we enumerate the elapsed_days and hold its returned tuple of an index and elapsed days in i and t respectively.Until and unless the elapsed day has not reached the next day we append the index to the empty list inds.

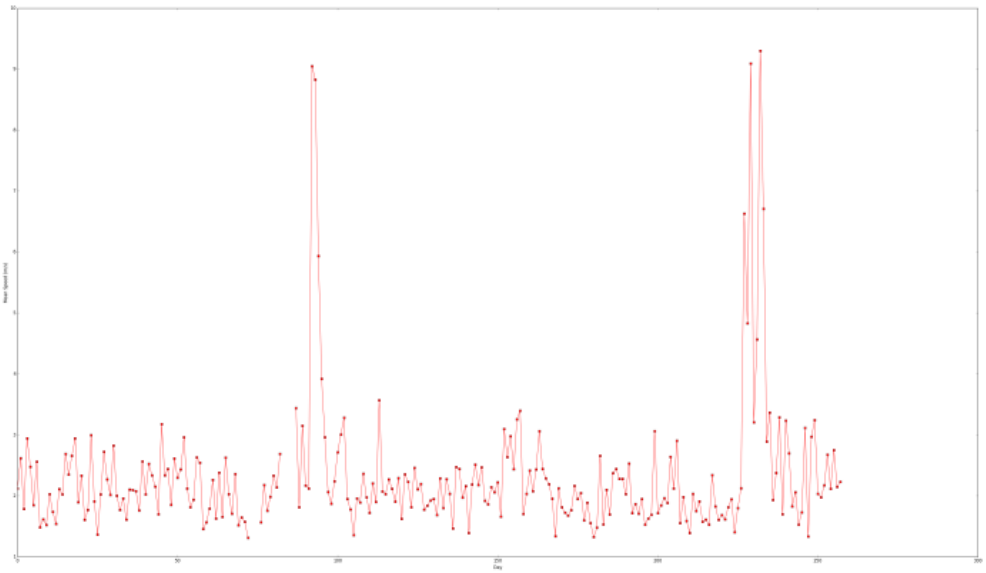
Or else If we reach the next day, we append speed data to daily mean speed and increment next_day by 1.

Lastly, we plot the figure of size 8 x 6

```
∇  plt.plot(daily_mean_speed, “rs-“)
   plt.xlabel(” Day “)
   plt.ylabel(” Mean Speed (m/s) “);
   plt.show()
```

In the plot code, r represent red, s represents square (for the turning points) and – represents the visualization style of the curve.We next, label the x and y axis using the xlabel() and ylabel() as “Day” and ” Mean Speed (m/s) ” respectively.Lastly, we show() the final plot.

Output: enlarged_view



PART (5/5): Cartographic View

In this last part, i.e part 5, we are going to track the Birds over Political Map.

bird_migration_cartographic.py

```
import cartopy.crs as ccrs
import cartopy.feature as cfeature
import matplotlib.pyplot as plt

proj = ccrs.Mercator() #To move forward, we need to specify a specific projection that we're interested in using.

plt.figure(figsize=(10,10))
ax = plt.axes(projection=proj)
ax.set_extent((-25.0, 20.0, 52.0, 10.0))
ax.add_feature(cfeature.LAND)
ax.add_feature(cfeature.OCEAN)
ax.add_feature(cfeature.COASTLINE)
ax.add_feature(cfeature.BORDERS, linestyle=':')
for name in bird_names:
    ix = birddata['bird_name'] == name
    x,y = birddata.longitude[ix], birddata.latitude[ix]
    ax.plot(x,y,'.', transform=ccrs.Geodetic(), label=name)
plt.legend(loc="upper left")
plt.show()
```

We import the cartopy and matplotlib module along with its salient libraries.

∇ proj = ccrs.Mercator()

To move forward, we need to specify a specific projection that we're interested in using.So we are using the cartopy Mercator() function and initializing it to proj.

```
∇  plt.figure(figsize=(10,10))
   ax = plt.axes(projections=proj)
   ax.set_extent((-25.0, 20.0, 52.0, 10.0))
   ax.add_feature(cfeature.LAND)
   ax.add_feature(cfeature.OCEAN)
   ax.add_feature(cfeature.COASTLINE)
   ax.add_feature(cfeature.BORDERS, linestyle=':')
```

We plot a 10 x 10 figure and draw an axis with projection along the variable proj.Next, we add the political features like Land, Ocean, Coastline and borders into our plot.We automatically get the Political shapes and features mentioned above according to the gps locations (i.e, latitude and longitude) present in our data.

```
for name in bird_names:
    ix = birddata['bird_name'] == name
    x,y = birddata.longitude[ix], birddata.latitude[ix]
    ax.plot(x,y,'.', transform=ccrs.Geodetic(), label=name)

plt.legend(loc="upper left")
plt.show()
```

For every gull we plot its latitude and longitude data and use the Geodic() function to conform with the Geographical features.

Lastly, we visualize the mapped data of the gulls.

Output : enlarged_view

Resources :

- edX – Course
- Python Functions
- GitHub repo
- Tutorials Point – Python
- Python Doc
- Google Developers – Python
- Learnpythonthehardway.org
- Codecademy – Python