UNIVERSITE JOSEPH KI-ZERBO (UJKZ)

VICE-PRESIDENCE CHARGEE DE LA RECHERCHE ET DE LA COOPERATION INTERNATIONALE (VP/RCI)

ECOLE DOCTORALE INFORMATIQUE ET CHANGEMENTS CLIMATIQUES (ED/ICC)

Master Research program in Informatics for climate change (MRP-ICC)

Project

Spatial Data Analysis

Doctor:

Dr. Sarah Schönbrodt-Stitt

Mr. Steven Hill

Student:

ADZAVON koffi Doh David

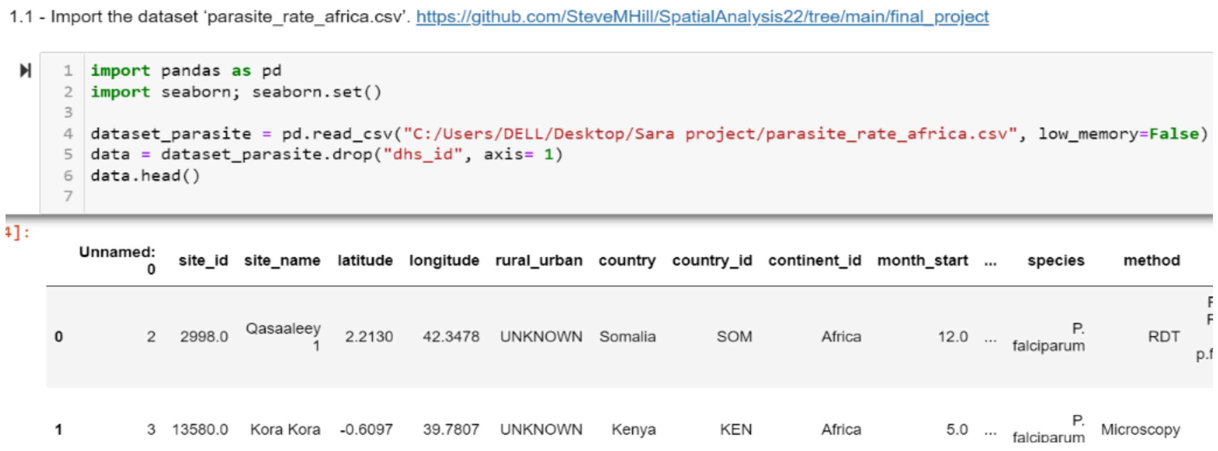
Batch3

2022-2023

PART 1

1 - Import the dataset ‘parasite_rate_africa.csv’.

I created for this project a folder where I store my python code, CSV file, shapefile, and raster data. To import the dataset **‘parasite_rate_africa.csv’,** I imported the pandas package to read the CSV file.



1.1 - Import the dataset ‘parasite_rate_africa.csv’. https://github.com/SteveMHill/SpatialAnalysis22/tree/main/final_project

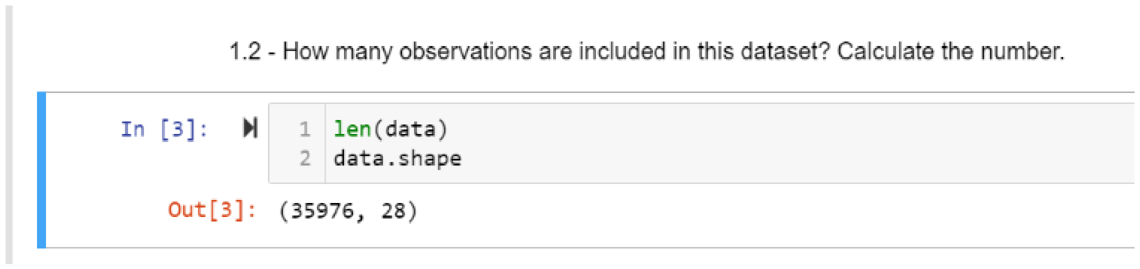
```
import pandas as pd
import seaborn; seaborn.set()

dataset_parasite = pd.read_csv("C:/Users/DELL/Desktop/Sara project/parasite_rate_africa.csv", low_memory=False)
data = dataset_parasite.drop("dhs_id", axis= 1)
data.head()
```

4]:

	Unnamed: 0	site_id	site_name	latitude	longitude	rural_urban	country	country_id	continent_id	month_start	...	species	method	
0	2	2998.0	Qasaaleey 1	2.2130	42.3478	UNKNOWN	Somalia	SOM	Africa	12.0	...	P. falciparum	RDT	F F p.f
1	3	13580.0	Kora Kora	-0.6097	39.7807	UNKNOWN	Kenya	KEN	Africa	5.0	...	P. falciparum	Microscopy	

1.2: How many observations are included in this dataset? Calculate the number.

To have a number of observations that are included in my dataset, it comes to calculate the length of the dataset or to know the shape of my dataset. By doing so, I find that my dataset contains 35976 observations.

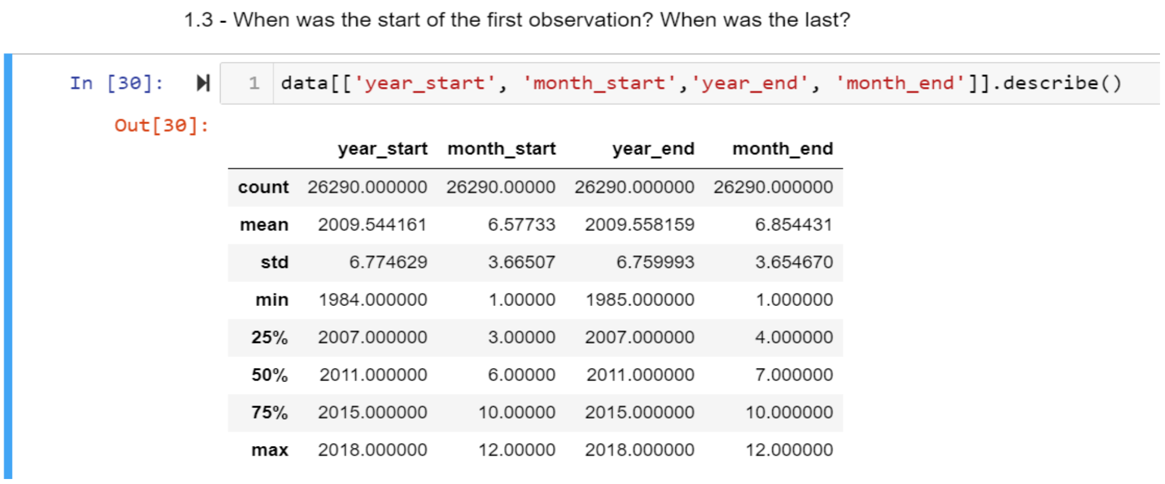


1.2 - How many observations are included in this dataset? Calculate the number.

```
In [3]:
len(data)
data.shape

Out[3]: (35976, 28)
```

1.3: When was the start of the first observation? When was the last?

Here, our work is concerning the date of the first observation and the date of the last observation. If we apply head() function to have a look at the first date we can see that the first date was not right which means that the data are not well arranged in our dataset so it comes for us to come back and rearrange our data in the dataset or to apply the filter technique which I choose. To do so, I use the describe function to know the minimum and the maximum in terms of the year.

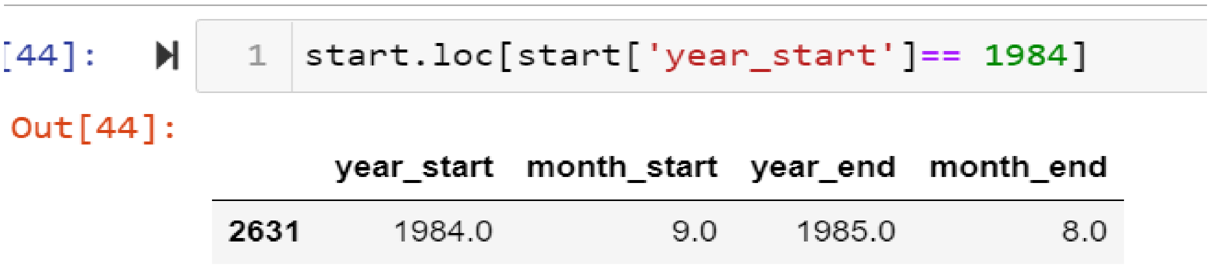


1.3 - When was the start of the first observation? When was the last?

```
In [30]:  data[['year_start', 'month_start','year_end', 'month_end']].describe()
Out[30]:
```

	year_start	month_start	year_end	month_end
count	26290.000000	26290.00000	26290.000000	26290.000000
mean	2009.544161	6.57733	2009.558159	6.854431
std	6.774629	3.66507	6.759993	3.654670
min	1984.000000	1.00000	1985.000000	1.000000
25%	2007.000000	3.00000	2007.000000	4.000000
50%	2011.000000	6.00000	2011.000000	7.000000
75%	2015.000000	10.00000	2015.000000	10.000000
max	2018.000000	12.00000	2018.000000	12.000000

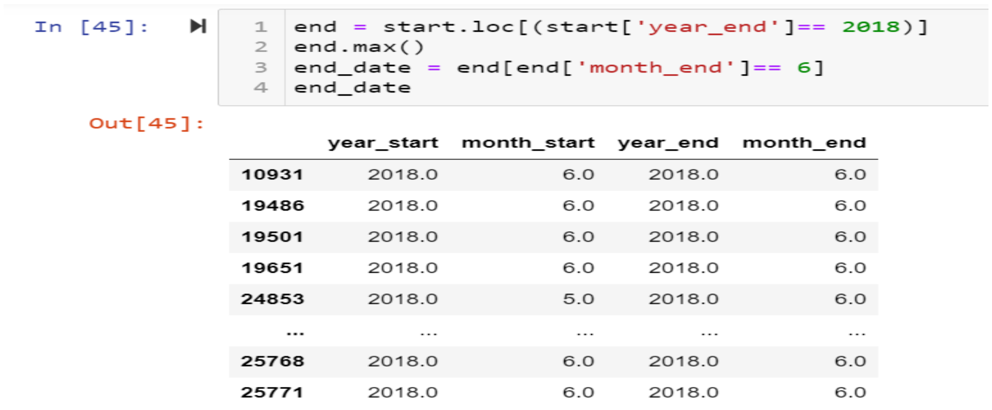
Now I know that my first year of the record is 1984 and the last record has been done in 2018. Therefore, I apply the function loc() to select from my dataset every year starting by 1984 and I found that the first observation has been done in September 1984 (09/1984).



```
[44]:  start.loc[start['year_start']== 1984]
Out[44]:
```

	year_start	month_start	year_end	month_end
2631	1984.0	9.0	1985.0	8.0

So, for the end date also the same technique is applied and I got that my last date of record has been bone in Jun 2018 (06/2018)



In [45]:

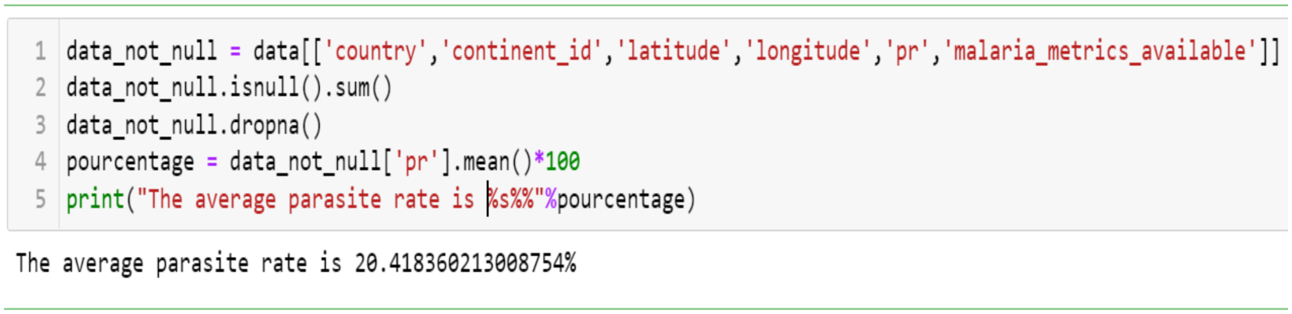
```
end = start.loc[(start['year_end']== 2018)]
end.max()
end_date = end[end['month_end']== 6]
end_date
```

Out[45]:

	year_start	month_start	year_end	month_end
10931	2018.0	6.0	2018.0	6.0
19486	2018.0	6.0	2018.0	6.0
19501	2018.0	6.0	2018.0	6.0
19651	2018.0	6.0	2018.0	6.0
24853	2018.0	5.0	2018.0	6.0
...	...	...	...	...
25768	2018.0	6.0	2018.0	6.0
25771	2018.0	6.0	2018.0	6.0

1.4: What is the average (mean) parasite rate (%)? Calculate.

To calculate the parasite rate, my data contain much missing value. In this case, I choose to remove all the missing values in my dataset. But before that, I tried to reduce my dataset to just the columns that will be useful for me further and then move all the missing values. Therefore, columns like: 'country','continent_id','latitude','longitude','pr','malaria_metrics_available' has been selected. After selection, I use the isnull() fuction to see the missing values. Then I use the dropna() function to drop them in my dataset and calculate the mean for the column parasite rate. The result gives: 20.418 %

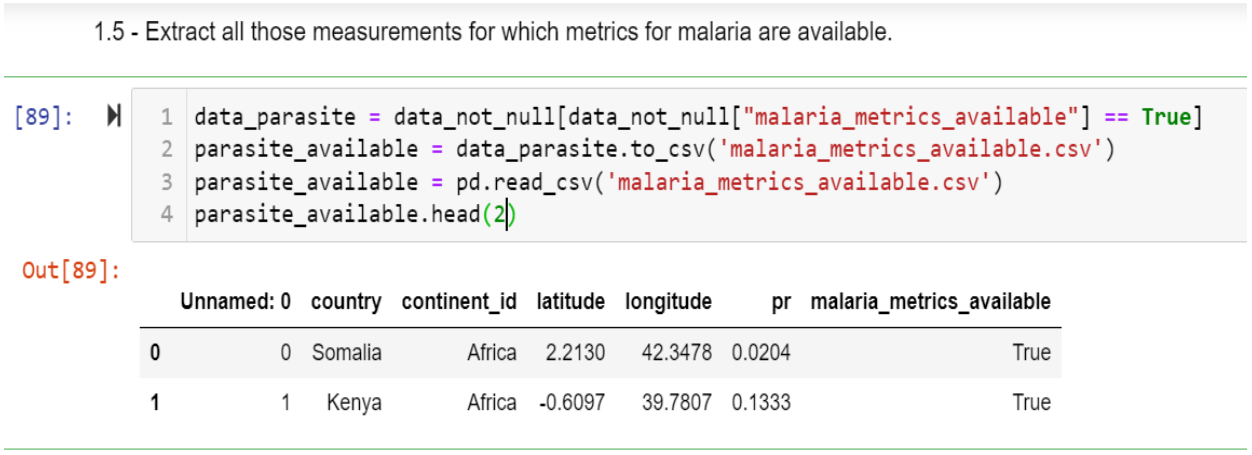


```
data_not_null = data[['country','continent_id','latitude','longitude','pr','malaria_metrics_available']]
data_not_null.isnull().sum()
data_not_null.dropna()
pourcentage = data_not_null['pr'].mean()*100
print("The average parasite rate is %s%%"%pourcentage)
```

```
The average parasite rate is 20.418360213008754%
```

1.5: Extract all those measurements for which metrics for malaria are available.

To extract the measurements for which metrics malaria are available, it comes more easier now, as we have already a sub-dataset. We just have to index the columns 'malaria_metrics_available' and pick where it’s written True.


1.5 - Extract all those measurements for which metrics for malaria are available.

[89]:
```python
data_parasite = data_not_null[data_not_null["malaria_metrics_available"] == True]
parasite_available = data_parasite.to_csv('malaria_metrics_available.csv')
parasite_available = pd.read_csv('malaria_metrics_available.csv')
parasite_available.head(2)
```

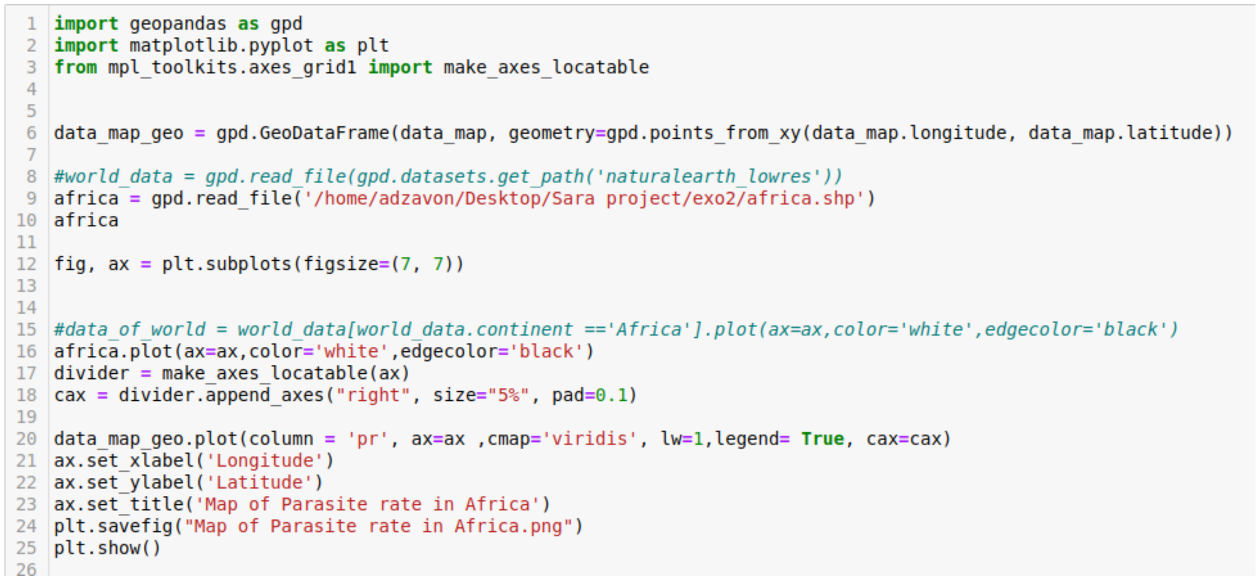
Out[89]:

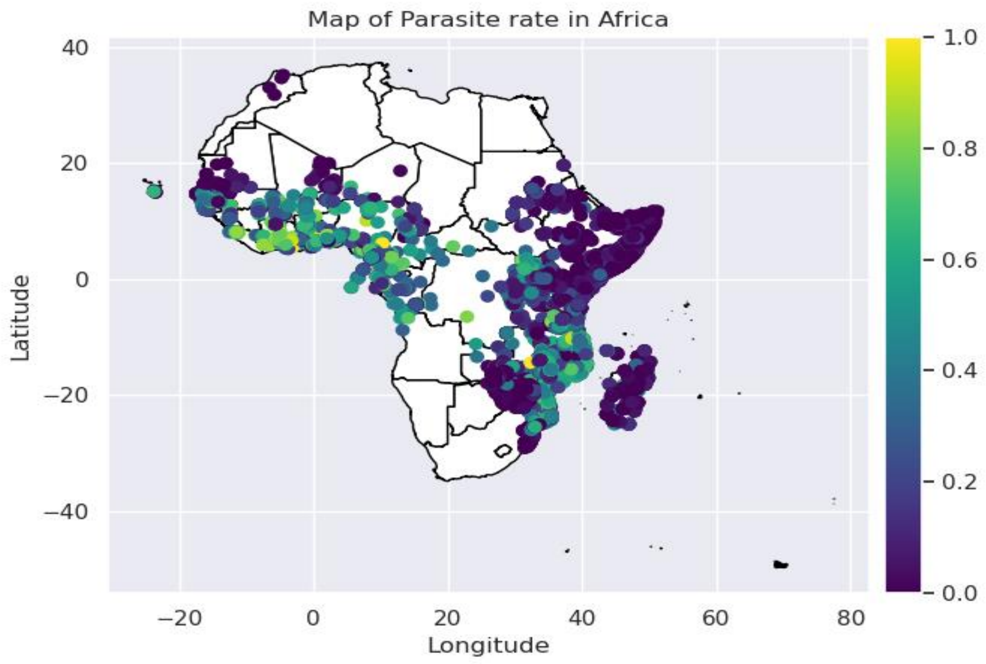
	Unnamed: 0	country	continent_id	latitude	longitude	pr	malaria_metrics_available
0	0	Somalia	Africa	2.2130	42.3478	0.0204	True
1	1	Kenya	Africa	-0.6097	39.7807	0.1333	True

1.7 - Create a map showing the parasite rate in Africa. The map shall show the country's borders, the location of the surveys, and the parasite rate for each point. Think of a smart way to represent this data on a map (You must not forget to provide an appropriate legend and axis description).

To be able to do that, we need the geopandas package to work with the geospatial data. We import the geopandas package and also matplotlib for a plot. Also, from matplotlib, I imported the module "make_axes_locatable" so that I will be able to plot many axis on the given side of my original axis.

I downloaded Africa shapefile from geopandas that I use for the plot, and I plot on that, the observation of the parasite rate and set the legend which is the different value of the record of the parasite rate.

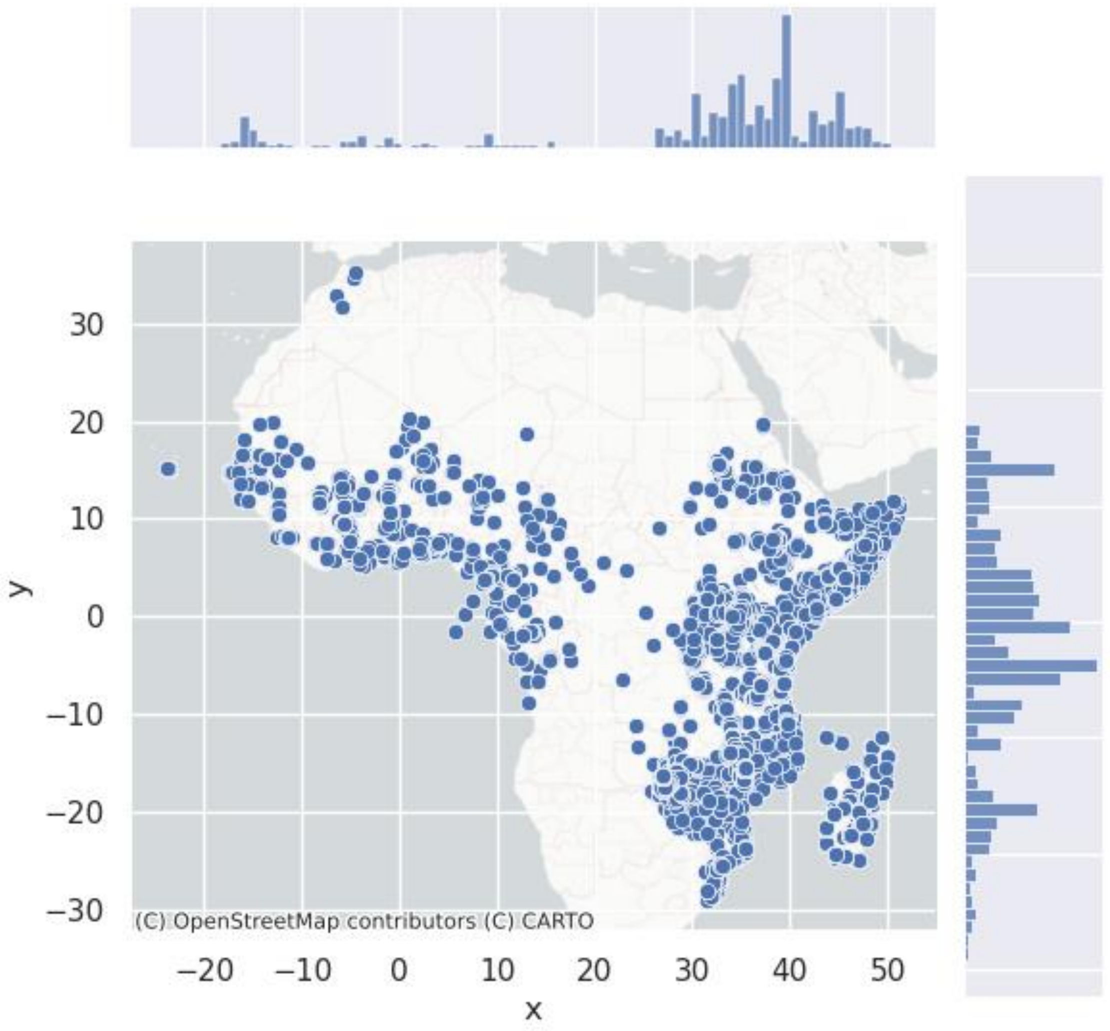

```python
import geopandas as gpd
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable


data_map_geo = gpd.GeoDataFrame(data_map, geometry=gpd.points_from_xy(data_map.longitude, data_map.latitude))

#world_data = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
africa = gpd.read_file('/home/adzavon/Desktop/Sara project/exo2/africa.shp')
africa

fig, ax = plt.subplots(figsize=(7, 7))


#data_of_world = world_data[world_data.continent =='Africa'].plot(ax=ax,color='white',edgecolor='black')
africa.plot(ax=ax,color='white',edgecolor='black')
divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.1)

data_map_geo.plot(column = 'pr', ax=ax ,cmap='viridis', lw=1,legend= True, cax=cax)
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
ax.set_title('Map of Parasite rate in Africa')
plt.savefig("Map of Parasite rate in Africa.png")
plt.show()

```



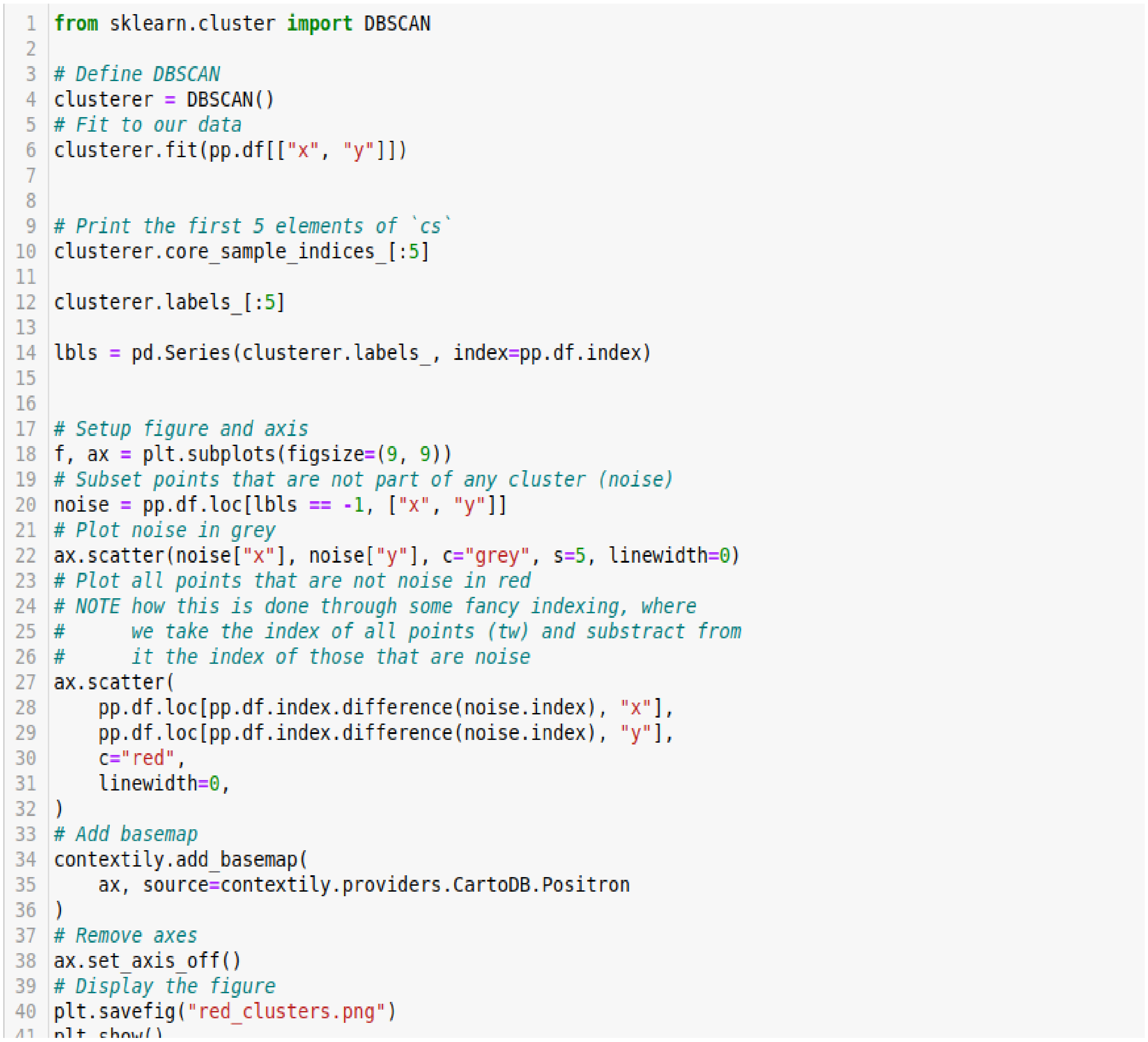
1.8 - Analyze the point pattern of observations, e.g., analyze if the observations are clustered or not AND explain the result.

For this purpose, I try first to display first, the observation and use the histogram for a basic analysis.

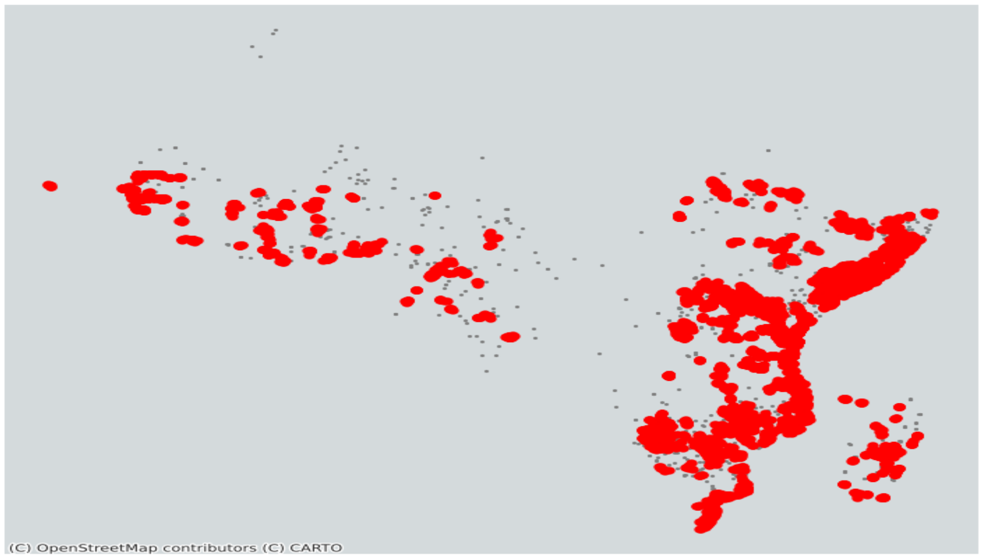
```python
import seaborn as sns
import contextily
import matplotlib.pyplot as plt
plot = sns.jointplot(x="x", y="y", data=pp.df)
contextily.add_basemap( plot.ax_joint, crs="EPSG:4326",
                        source=contextily.providers.CartoDB.PositronNoLabels)
plt.savefig("cluster.png")
```



This map is showing where are the different clusters. We can see that with the histogram plot, in the east of Africa, there are many observations, and also in west Africa. So, this is the simplest way to identify with histogram the clusters. To avoid confusion in the way, the observation is clustered, we can go further and distinguish the cluster's location. For that, I use DBSCAN Density-based spatial clustering of applications with noise technique which is usually used in machine learning and also in data mining.



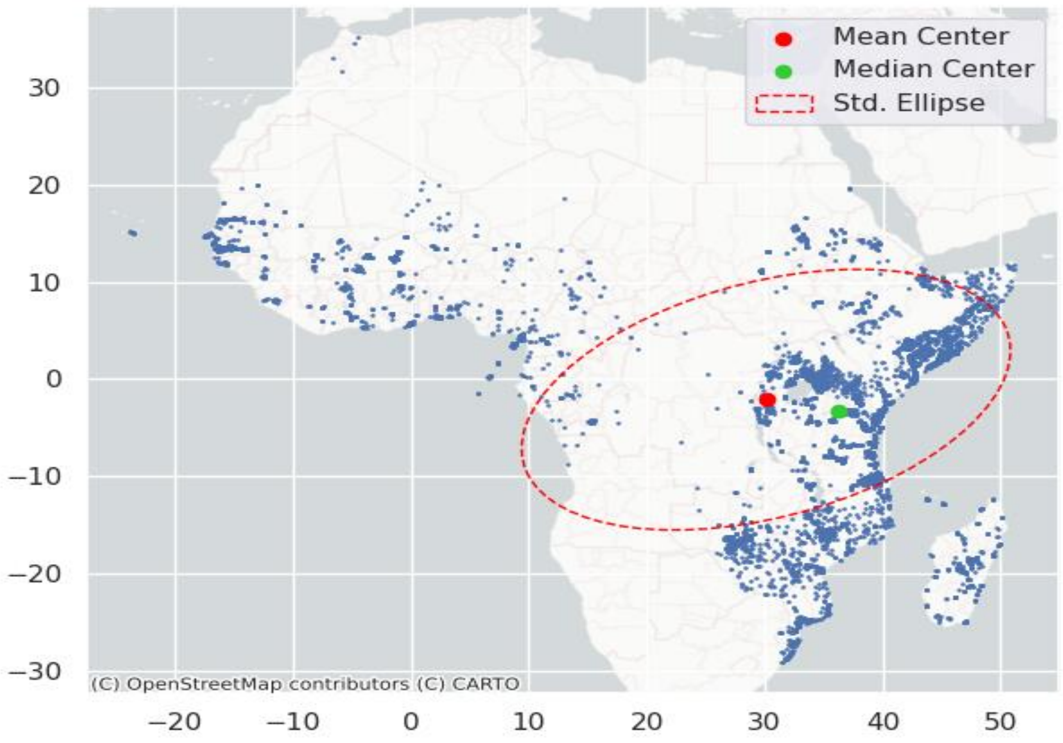
```python
from sklearn.cluster import DBSCAN

# Define DBSCAN
clusterer = DBSCAN()
# Fit to our data
clusterer.fit(pp.df[["x", "y"]])


# Print the first 5 elements of `cs`
clusterer.core_sample_indices_[:5]

clusterer.labels_[:5]

lbls = pd.Series(clusterer.labels_, index=pp.df.index)


# Setup figure and axis
f, ax = plt.subplots(figsize=(9, 9))
# Subset points that are not part of any cluster (noise)
noise = pp.df.loc[lbls == -1, ["x", "y"]]
# Plot noise in grey
ax.scatter(noise["x"], noise["y"], c="grey", s=5, linewidth=0)
# Plot all points that are not noise in red
# NOTE how this is done through some fancy indexing, where
#      we take the index of all points (tw) and substract from
#      it the index of those that are noise
ax.scatter(
    pp.df.loc[pp.df.index.difference(noise.index), "x"],
    pp.df.loc[pp.df.index.difference(noise.index), "y"],
    c="red",
    linewidth=0,
)
# Add basemap
contextily.add_basemap(
    ax, source=contextily.providers.CartoDB.Positron
)
# Remove axes
ax.set_axis_off()
# Display the figure
plt.savefig("red_clusters.png")
plt.show()
```



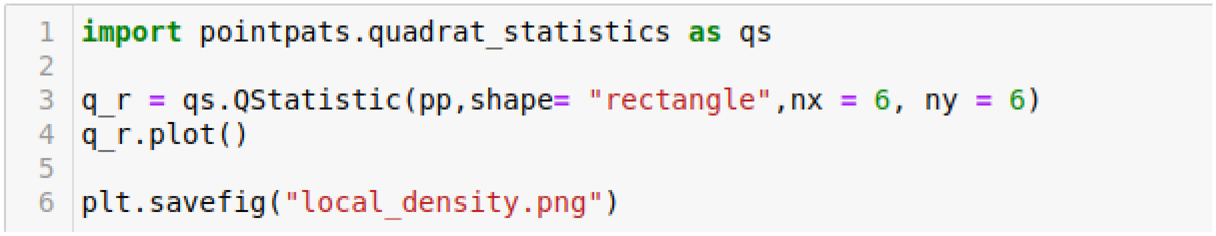
From this, we can directly see the two types of the cluster the first in the east Africa which is the biggest and the second one which is in the west Africa. Of course we have all the observations but where is not considered as a cluster is in a black colors only the red specifies the clusters.

Many techniques can be used. The Methods for Pattern point analysis are Centrography, Density analysis, Distance analysis, K function or Modeling. In my case, I choose to do the centrography where I will calculate and plot the mean center, median center, and also the standard deviation ellipse on the map with the observation. This method help to obtain the basic descriptive statistics of point data.

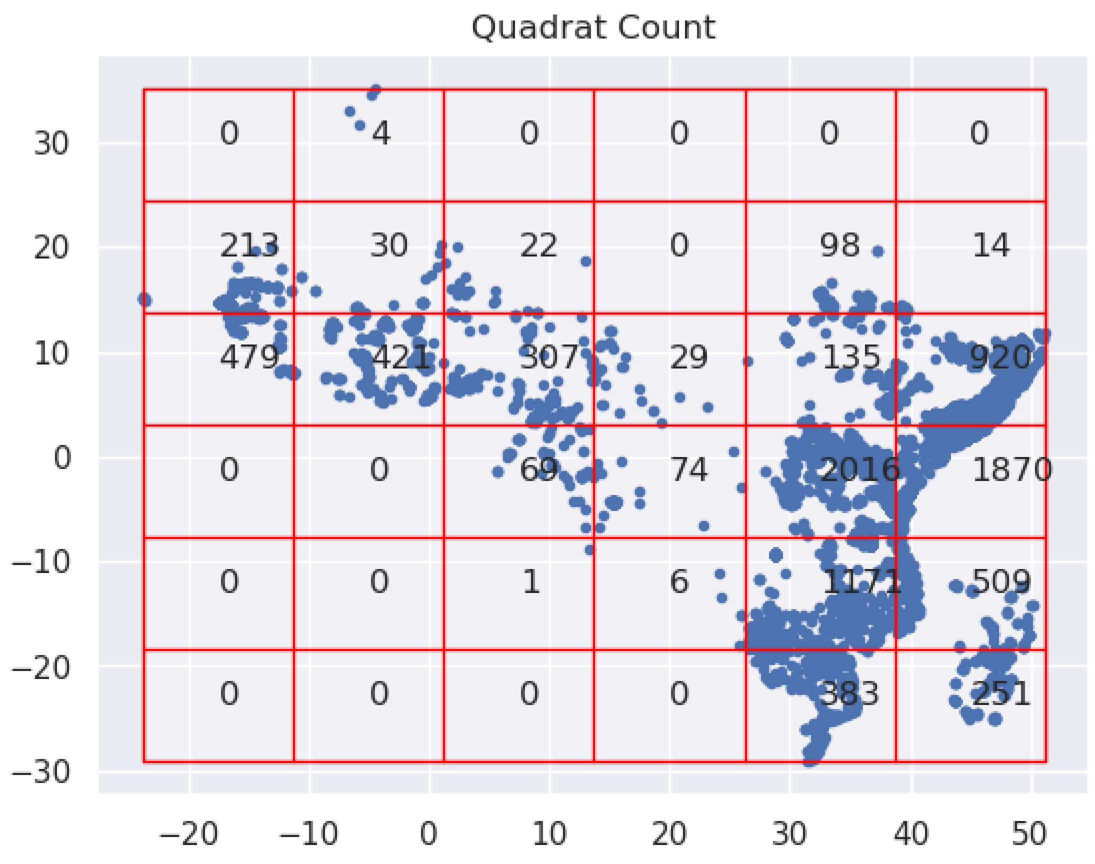
```python
from matplotlib.patches import Ellipse
import numpy
import contextily

# Set up figure and axis
f, ax = plt.subplots(1, figsize=(7, 7))
# Plot photograph points
ax.scatter(pp.df["x"], pp.df["y"], s=0.75)

contextily.add_basemap(ax, crs="EPSG:4326",
                       source=contextily.providers.CartoDB.PositronNoLabels)

ax.scatter(*mean_center, color="red", marker="o", label="Mean Center")
ax.scatter(*med_center, color="limegreen", marker="o", label="Median Center")
#ax.scatter(std_distance, color="yellow", label="standard distance")

# Construct the standard ellipse using matplotlib

ellipse = Ellipse(
    xy=mean_center,  # center the ellipse on our mean center
    width=major * 2,  # centrography.ellipse only gives half the axis
    height=minor * 2,
    angle=numpy.rad2deg(
        rotation
    ),  # Angles for this are in degrees, not radians
    facecolor="none",
    edgecolor="red",
    linestyle="--",
    label="Std. Ellipse",
)
ax.add_patch(ellipse)


ax.legend()

plt.savefig("ellipse.png")

plt.show()
```



For the Density analysis which is a technique to characterize the pattern of observed objects (points) regarding to their distribution in the sampled window. This technique contains two types of techniques local density or a global density. In this work, I did the local density technique which consists of analyzing the density of point patterns at different locations within the sampled window e.g., by using quadrat (sub-regions) or kernel (moving window) density.



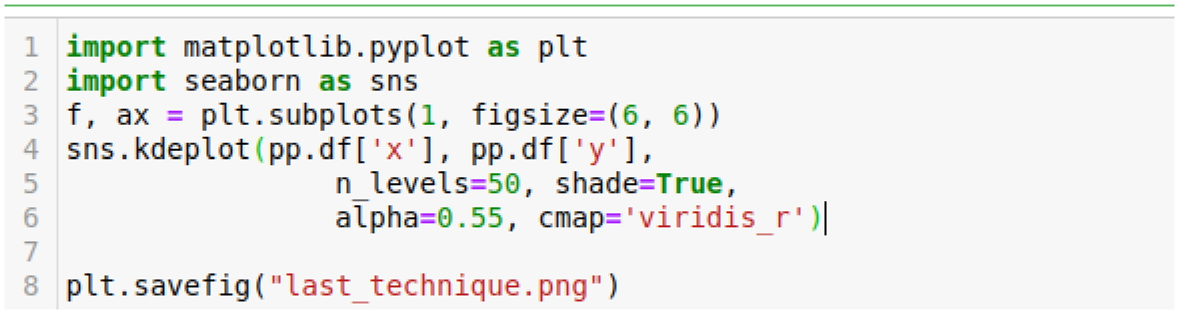
```
import pointpats.quadrat_statistics as qs

q_r = qs.QStatistic(pp,shape= "rectangle",nx = 6, ny = 6)
q_r.plot()

plt.savefig("local_density.png")
```



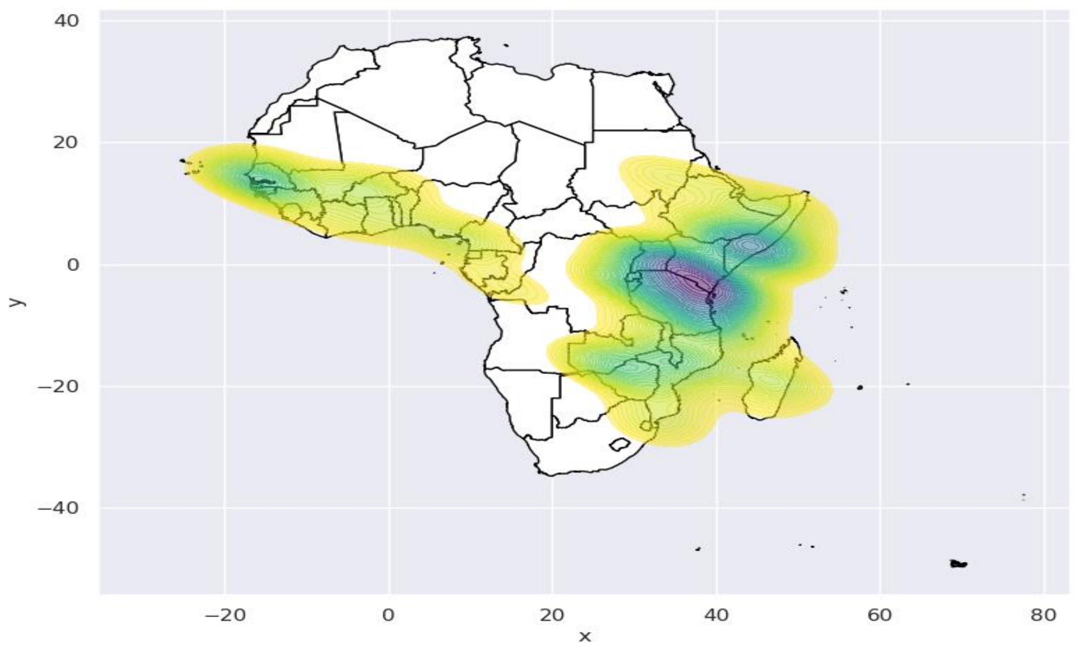
With this method, we can clearly have a number of observations at each location. The locations that have many observations are considered as clusters and in our case specifically, the highest value are located in the east and west Africa this is showing that we have two clusters.

1.9 - Create a map showing the main clusters of observations.

We can also use the technique of Kernel density which is almost the same as the local density method but in this case, instead of using the number of observations, we will rather plot them using a color. This is one of the best ways to locate clusters.

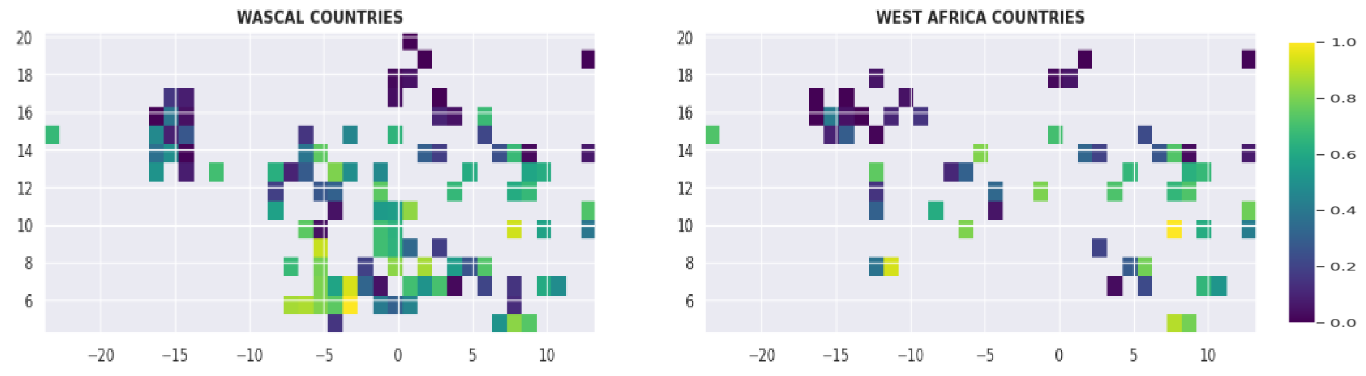


```python
import matplotlib.pyplot as plt
import seaborn as sns
f, ax = plt.subplots(1, figsize=(6, 6))
sns.kdeplot(pp.df['x'], pp.df['y'],
                n_levels=50, shade=True,
                alpha=0.55, cmap='viridis_r')

plt.savefig("last_technique.png")
```



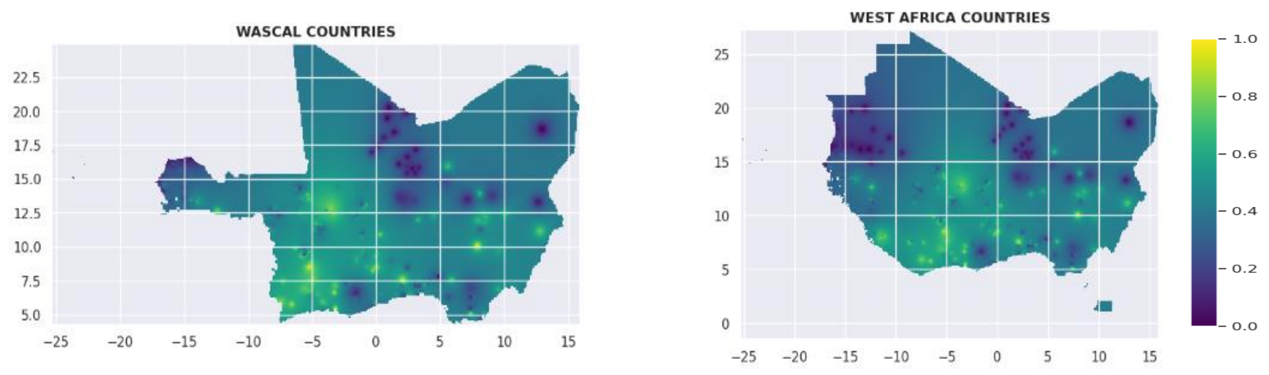
1.10.1:Extract all those observations for (i) West African countries (16 countries!) and for (ii) the WASCAL member countries. Create a raster of the parasite rate for each of the two regions (i and ii) using spatial interpolation and validate your result.

from my dataset, I select all the WASCAL countries in one file and also in the one another file I selected the West African countries. Then I use QGIS to do the raster using the spatial interpolation.



1.10.2 - Create two maps showing your results. Explain the results.

To do so, I use my created csv file which is the one that contains observation, Therefore, I use a loc function to select the different countries and created a new csv file based on it. I did the work in QGIS which I master a bit, just to find the simplest way to do my work, and after I imported my result in python where I save the result and displayed it. We can see that country like Senegal Cabo Verde the parasite rate is very low but in the other West African country that is also WASCAL countries the parasite rate is higher

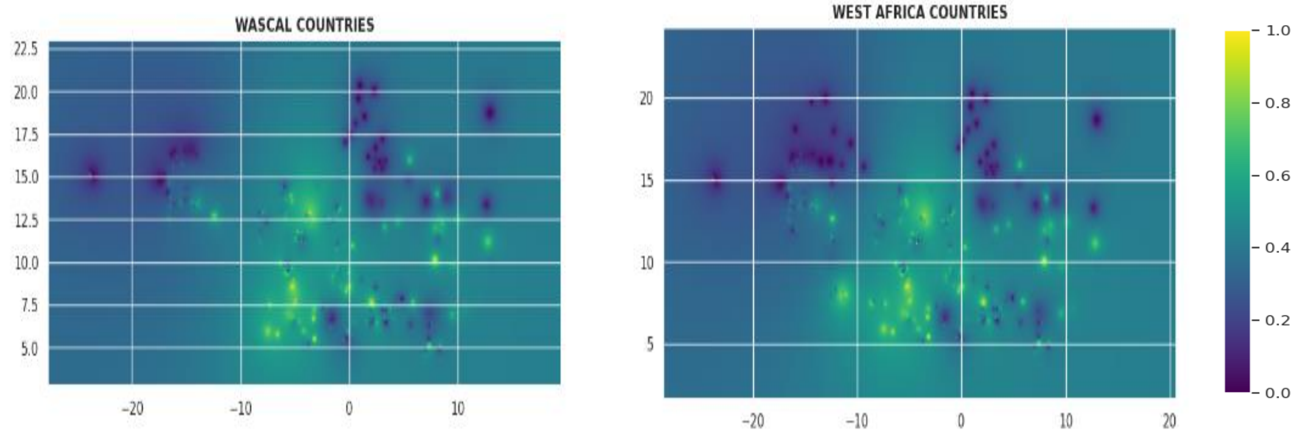


1.10.3 - Discuss possible errors in your interpolation.

Spatial interpolation is the process of using points with known values to estimate values at other unknown points. Based on this apply interpolation method, we can now have at least an idea or a value of some other area that are not located in our dataset. The, first error that I raise is that the estimation in certain zones is not really true.

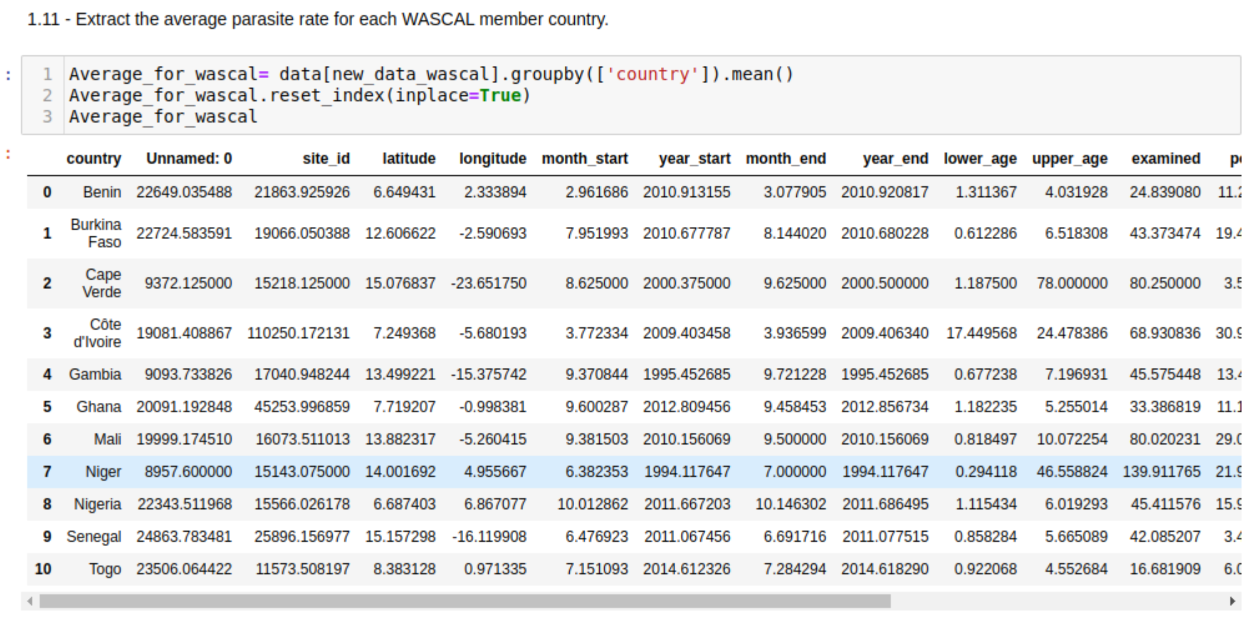
1.10.4 - Export your results as spatial rasters.

This is a raster I made, for west African countries and also, for WASCAL countries. The rasters are a grid of regularly sized pixels, they are showing continually varying information.



1.11- Extract the average parasite rate for each WASCAL member country

As the question is, I group each country by the different variables in my dataset. This has been done for all the variables (columns) and I did the mean. So, after I display the dataframe content.



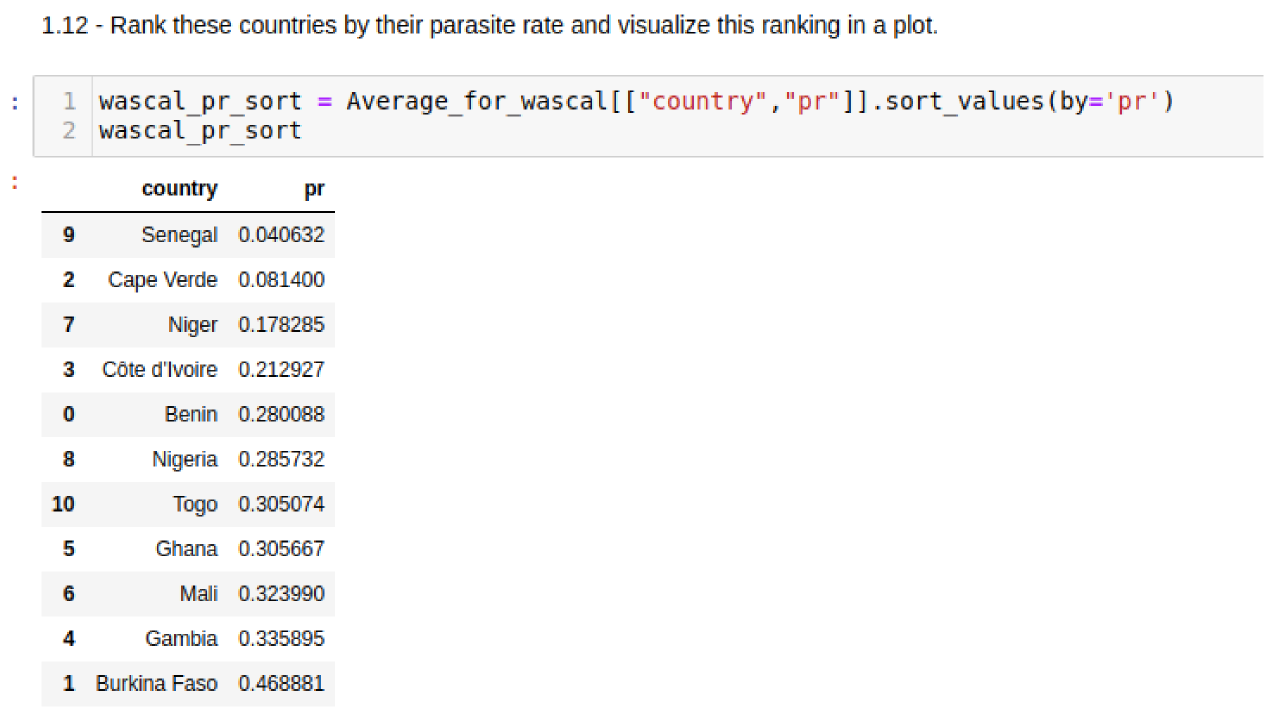
1.11 - Extract the average parasite rate for each WASCAL member country.

```
Average_for_wascal= data[new_data_wascal].groupby(['country']).mean()
Average_for_wascal.reset_index(inplace=True)
Average_for_wascal
```

	country	Unnamed: 0	site_id	latitude	longitude	month_start	year_start	month_end	year_end	lower_age	upper_age	examined	p
0	Benin	22649.035488	21863.925926	6.649431	2.333894	2.961686	2010.913155	3.077905	2010.920817	1.311367	4.031928	24.839080	11.
1	Burkina Faso	22724.583591	19066.050388	12.606622	-2.590693	7.951993	2010.677787	8.144020	2010.680228	0.612286	6.518308	43.373474	19.
2	Cape Verde	9372.125000	15218.125000	15.076837	-23.651750	8.625000	2000.375000	9.625000	2000.500000	1.187500	78.000000	80.250000	3.
3	Côte d'Ivoire	19081.408867	110250.172131	7.249368	-5.680193	3.772334	2009.403458	3.936599	2009.406340	17.449568	24.478386	68.930836	30.
4	Gambia	9093.733826	17040.948244	13.499221	-15.375742	9.370844	1995.452685	9.721228	1995.452685	0.677238	7.196931	45.575448	13.
5	Ghana	20091.192848	45253.996859	7.719207	-0.998381	9.600287	2012.809456	9.458453	2012.856734	1.182235	5.255014	33.386819	11.1
6	Mali	19999.174510	16073.511013	13.882317	-5.260415	9.381503	2010.156069	9.500000	2010.156069	0.818497	10.072254	80.020231	29.
7	Niger	8957.600000	15143.075000	14.001692	4.955667	6.382353	1994.117647	7.000000	1994.117647	0.294118	46.558824	139.911765	21.
8	Nigeria	22343.511968	15566.026178	6.687403	6.867077	10.012862	2011.667203	10.146302	2011.686495	1.115434	6.019293	45.411576	15.
9	Senegal	24863.783481	25896.156977	15.157298	-16.119908	6.476923	2011.067456	6.691716	2011.077515	0.858284	5.665089	42.085207	3.
10	Togo	23506.064422	11573.508197	8.383128	0.971335	7.151093	2014.612326	7.284294	2014.618290	0.922068	4.552684	16.681909	6.

1.12 - Rank these countries by their parasite rate and visualize this ranking in a plot.

To answer this question which is to rank and create a plot for visualization, I try to rank everything in the ascendant rank so that it can be aligned with the question requirement, for that also the question has been done in python. Countries are selected and also the related parasite rate has been selected.

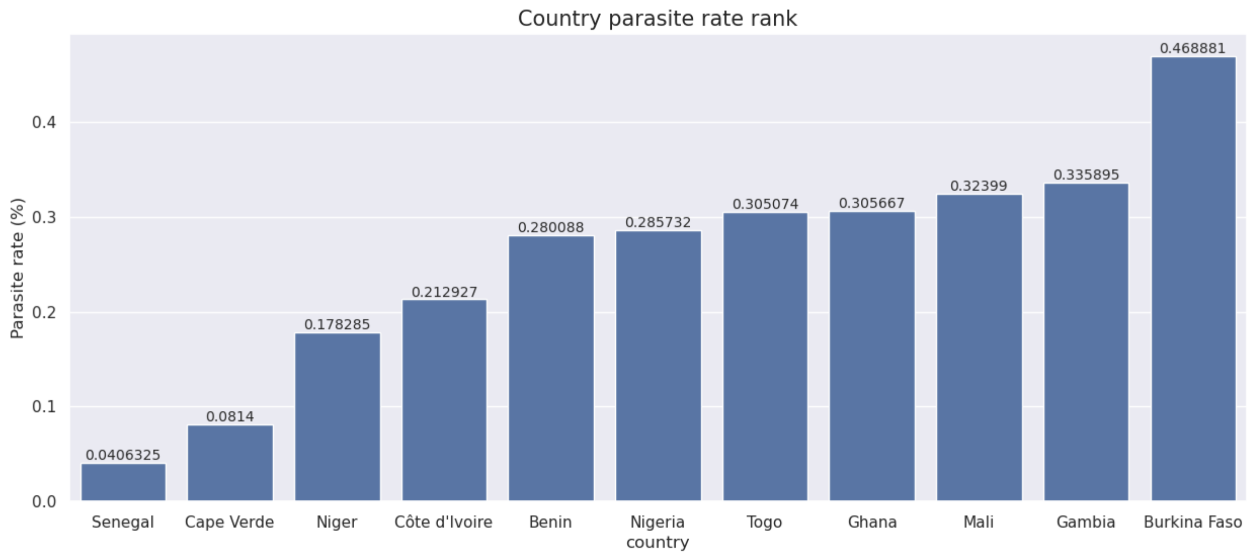


1.12 - Rank these countries by their parasite rate and visualize this ranking in a plot.

```
wascal_pr_sort = Average_for_wascal[["country","pr"]].sort_values(by='pr')
wascal_pr_sort
```

	country	pr
9	Senegal	0.040632
2	Cape Verde	0.081400
7	Niger	0.178285
3	Côte d'Ivoire	0.212927
0	Benin	0.280088
8	Nigeria	0.285732
10	Togo	0.305074
5	Ghana	0.305667
6	Mali	0.323990
4	Gambia	0.335895
1	Burkina Faso	0.468881

Therefore, we can see the ranking.

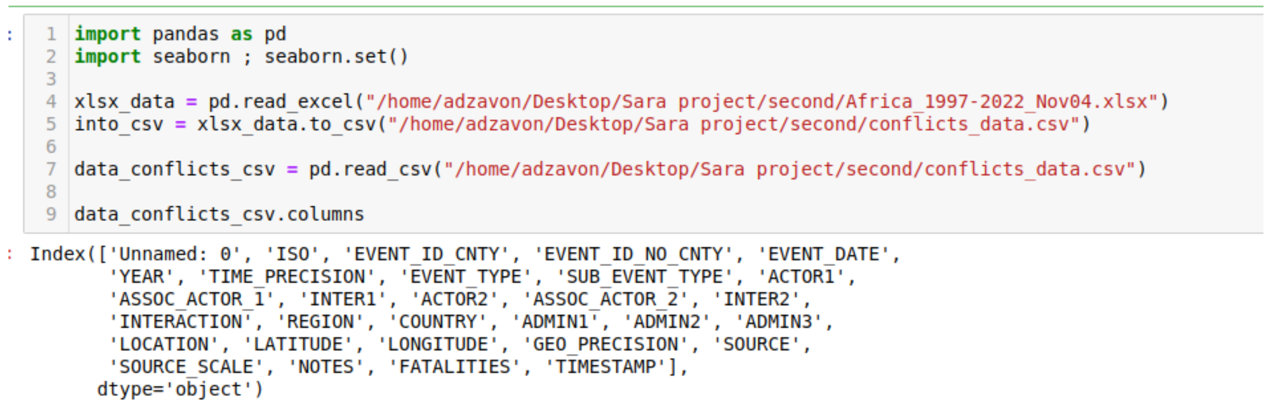


PART 2

For the second part, I use my own dataset which is a dataset that I downloaded from ACLED database which is the Armed Conflict Location & Event Data Project (https://acleddata.com/). The data contain the different countries where we have the incidents of the conflict, the fatalities, and the location of the incidents… in my work I am interested on the occurrence of the incident (the timestamp).

1 - Import the dataset.

My data have been downloaded in Excel (xlsx) format. Therefore, I change them in my notebook into CSV format and at the same time, I also, showed the different columns in my dataset to know the columns to select and work with.

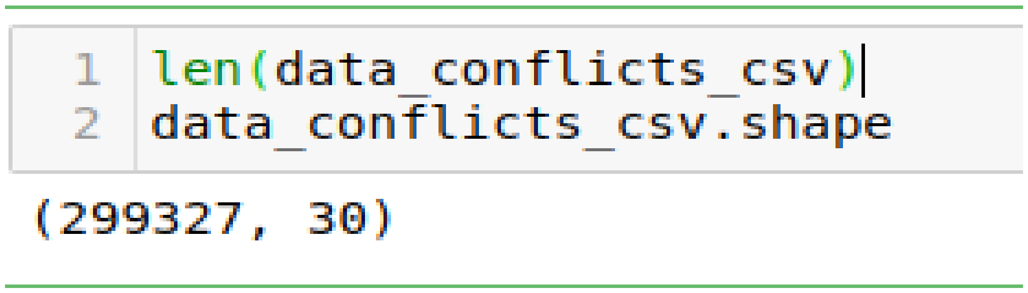


```
import pandas as pd
import seaborn ; seaborn.set()

xlsx_data = pd.read_excel("/home/adzavon/Desktop/Sara project/second/Africa_1997-2022_Nov04.xlsx")
into_csv = xlsx_data.to_csv("/home/adzavon/Desktop/Sara project/second/conflicts_data.csv")

data_conflicts_csv = pd.read_csv("/home/adzavon/Desktop/Sara project/second/conflicts_data.csv")

data_conflicts_csv.columns
```

```
Index(['Unnamed: 0', 'ISO', 'EVENT_ID_CNTY', 'EVENT_ID_NO_CNTY', 'EVENT_DATE',
       'YEAR', 'TIME_PRECISION', 'EVENT_TYPE', 'SUB_EVENT_TYPE', 'ACTOR1',
       'ASSOC_ACTOR_1', 'INTER1', 'ACTOR2', 'ASSOC_ACTOR_2', 'INTER2',
       'INTERACTION', 'REGION', 'COUNTRY', 'ADMIN1', 'ADMIN2', 'ADMIN3',
       'LOCATION', 'LATITUDE', 'LONGITUDE', 'GEO_PRECISION', 'SOURCE',
       'SOURCE_SCALE', 'NOTES', 'FATALITIES', 'TIMESTAMP'],
      dtype='object')
```

1.2 - How many observations are included in this dataset? Calculate the number.

To get the number of observations in my dataset**,** I apply the same method as the first part. I did len(), which gives me the exact number of observations in my dataset. I try to do it in another way by applying to my dataset shape(), the get the number of rows and also the number of columns. The result gives 299327 observations

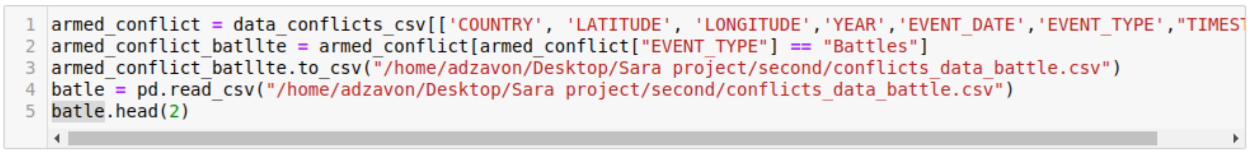


```
len(data_conflicts_csv)
data_conflicts_csv.shape
```

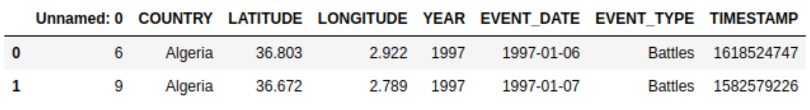
```
(299327, 30)
```

1.3 - When was the start of the first observation? When was the last?

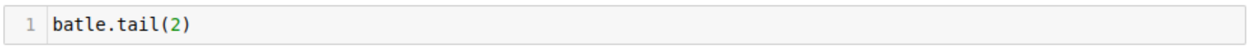
In my dataset, the date is already organized. So, to not struggle with the method, I show the first line which is the result and the first date. Also, the second case, for the end date, I try to show the tail of the dataframe. So, to answer the question, our first observation has been recorded in 06 January 1997 (1997-01-06) and the last observation has been recorded in 12 August 2021 (2021-08-12).



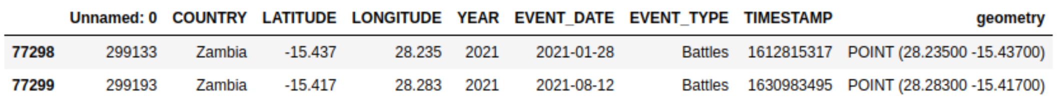
```
armed_conflict = data_conflicts_csv[['COUNTRY', 'LATITUDE', 'LONGITUDE','YEAR','EVENT_DATE','EVENT_TYPE',"TIMEST
armed_conflict_batllte = armed_conflict[armed_conflict["EVENT_TYPE"] == "Battles"]
armed_conflict_batllte.to_csv("/home/adzavon/Desktop/Sara project/second/conflicts_data_battle.csv")
batle = pd.read_csv("/home/adzavon/Desktop/Sara project/second/conflicts_data_battle.csv")
batle.head(2)
```



	Unnamed: 0	COUNTRY	LATITUDE	LONGITUDE	YEAR	EVENT_DATE	EVENT_TYPE	TIMESTAMP
0	6	Algeria	36.803	2.922	1997	1997-01-06	Battles	1618524747
1	9	Algeria	36.672	2.789	1997	1997-01-07	Battles	1582579226



```
batle.tail(2)
```



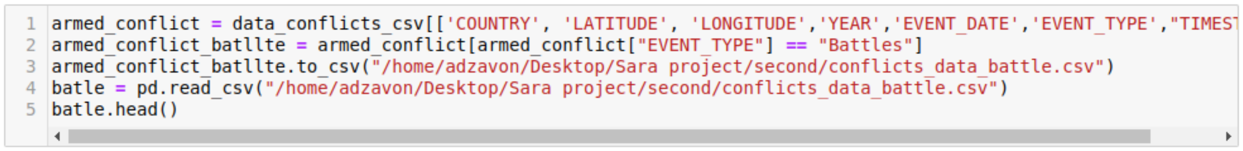
	Unnamed: 0	COUNTRY	LATITUDE	LONGITUDE	YEAR	EVENT_DATE	EVENT_TYPE	TIMESTAMP	geometry
77298	299133	Zambia	-15.437	28.235	2021	2021-01-28	Battles	1612815317	POINT (28.23500 -15.43700)
77299	299193	Zambia	-15.417	28.283	2021	2021-08-12	Battles	1630983495	POINT (28.28300 -15.41700)

1.4 - What is the average (mean) parasite rate (%)? Calculate.

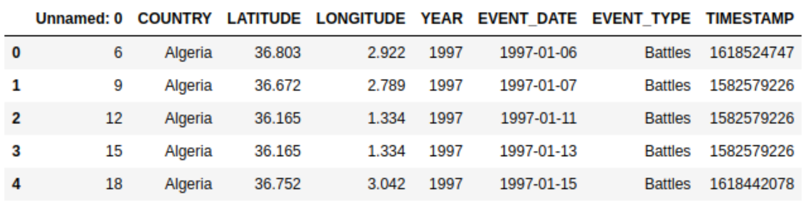
In this case, we cannot calculate the mean because the work is to count the occurrence of the different events.

1.5 - Extract all those measurements for which the characteristic battle is available

For the old session it was metric malaria but in my case us we are not dealing with the same variable I choose to deal with the battle occurrence which is the timestamp. Therefore, I extract all observations where the conflict is a battle among many others. There for after extraction, I put them in my new csv file that I created. Thus, I display the head of my new dataset.



```
armed_conflict = data_conflicts_csv[['COUNTRY', 'LATITUDE', 'LONGITUDE','YEAR','EVENT_DATE','EVENT_TYPE',"TIMEST
armed_conflict_batllte = armed_conflict[armed_conflict["EVENT_TYPE"] == "Battles"]
armed_conflict_batllte.to_csv("/home/adzavon/Desktop/Sara project/second/conflicts_data_battle.csv")
batle = pd.read_csv("/home/adzavon/Desktop/Sara project/second/conflicts_data_battle.csv")
batle.head()
```

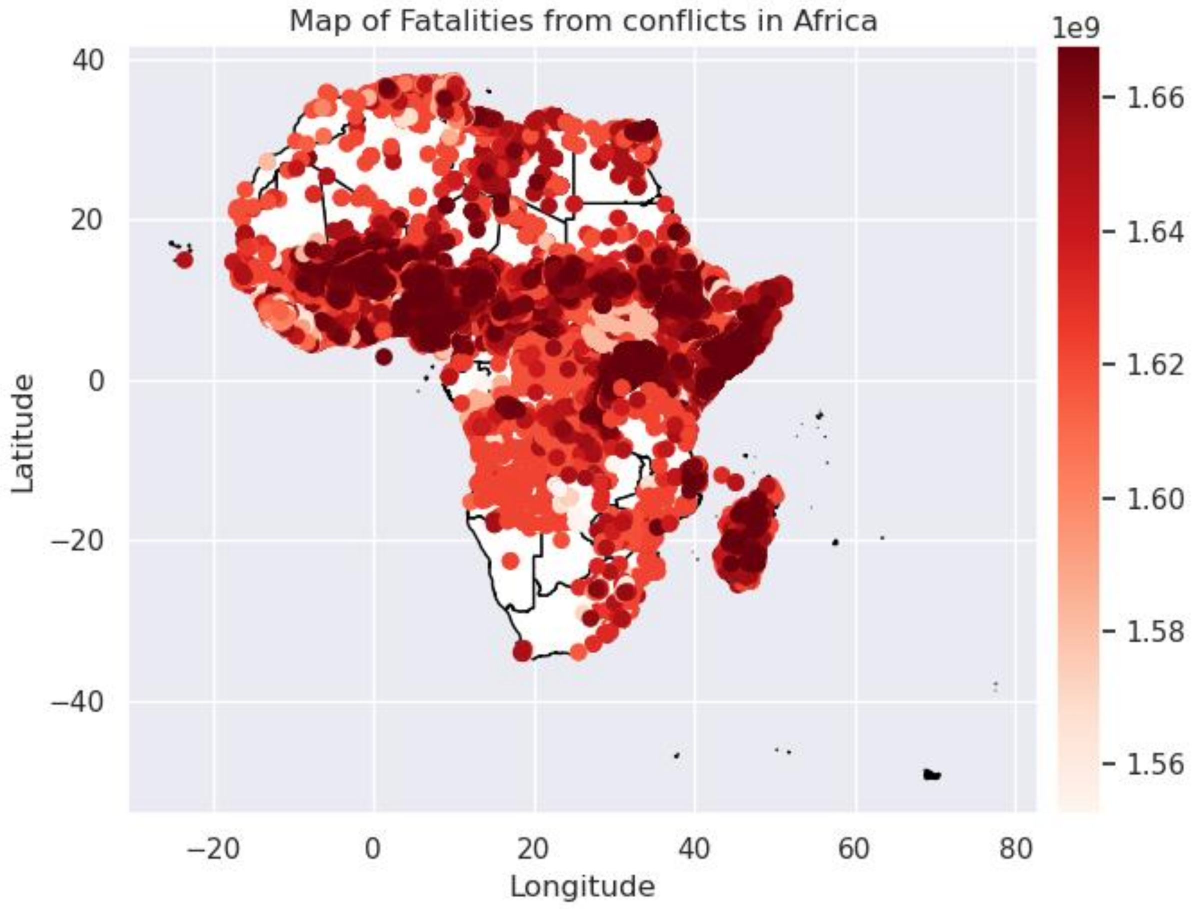


	Unnamed: 0	COUNTRY	LATITUDE	LONGITUDE	YEAR	EVENT_DATE	EVENT_TYPE	TIMESTAMP
0	6	Algeria	36.803	2.922	1997	1997-01-06	Battles	1618524747
1	9	Algeria	36.672	2.789	1997	1997-01-07	Battles	1582579226
2	12	Algeria	36.165	1.334	1997	1997-01-11	Battles	1582579226
3	15	Algeria	36.165	1.334	1997	1997-01-13	Battles	1582579226
4	18	Algeria	36.752	3.042	1997	1997-01-15	Battles	1618442078

1.7 - Create a map showing the battle timestamp in Africa. The map shall show the country's borders, the location of the surveys, and the battle occurrence for each point. Think of a smart way to represent this data on a map (You must not forget to provide an appropriate legend and axis description).

For my map, I based on the timestamp, and plot the battle occurrence event in the whole of Africa. We can see here that the battle is occurring everywhere in Africa. Where there are many occurrences, we can see that the point is very red and where it's not the point is light red. That's why we can see that in east Africa, center and in west Africa.

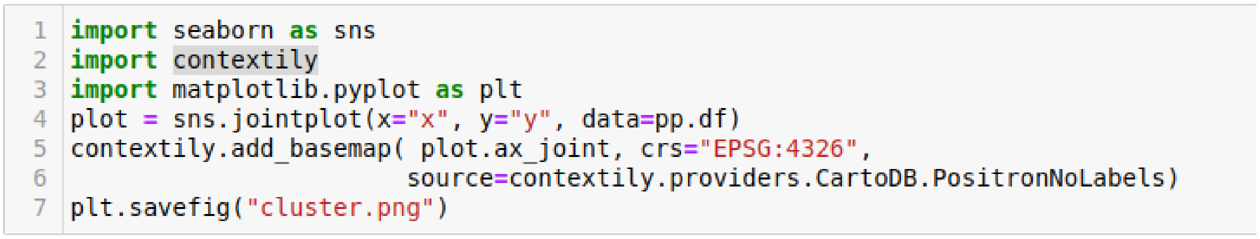
For the methodology, after having my csv file which contains the occurrence of the battle, I use the geopandas to change them in point so that I can plot them on my Africa shapefile which was downloaded from the internet.



1.8 - Analyze the point pattern of observations, e.g., analyze if the observations are clustered or not AND explain the result.

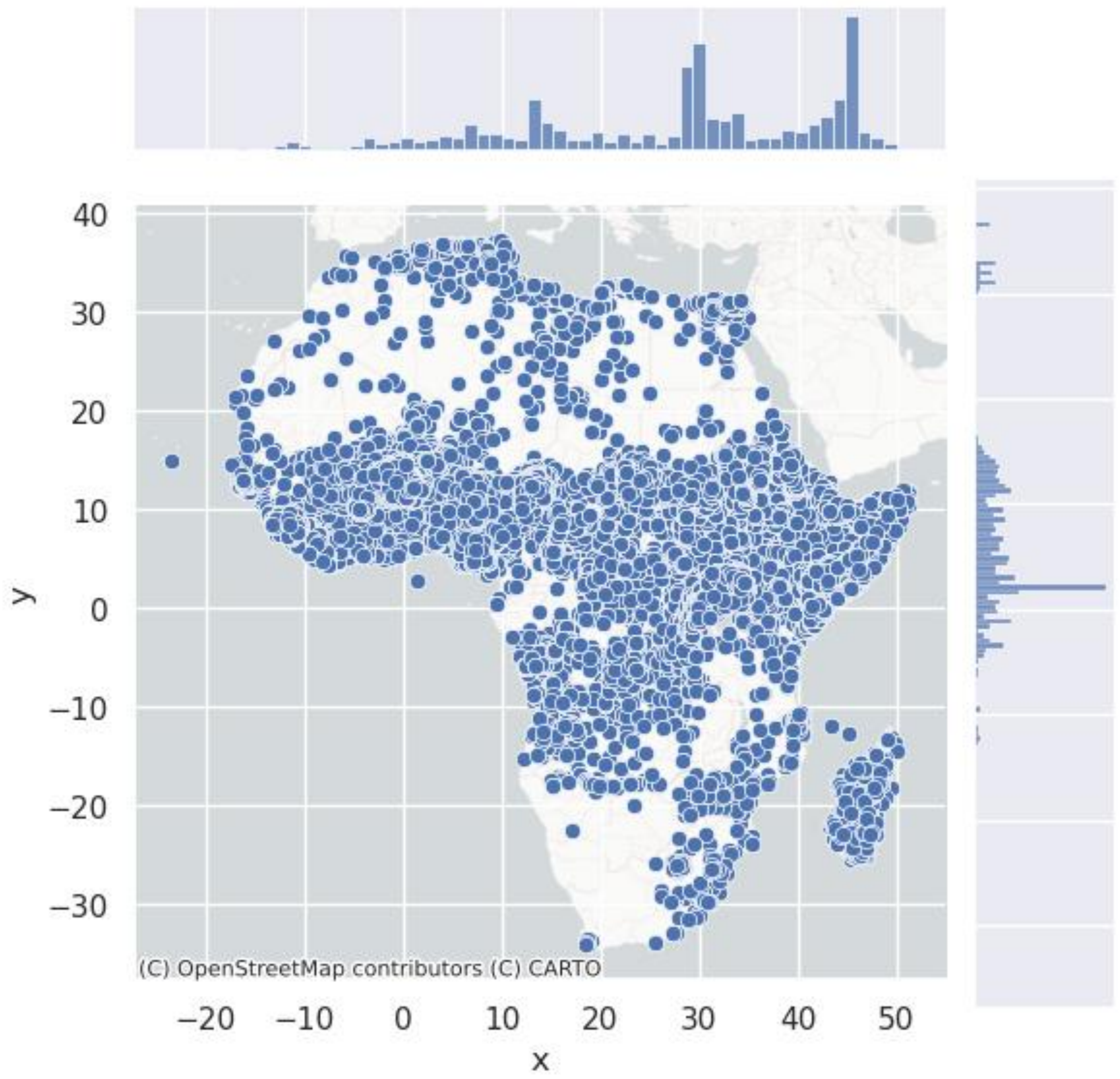
For the pattern point analysis, I tried to use different methods that can help me to identify the different clusters easier as soon as possible.

Therefore, I use pandas pattern point package and also the shapefile which contains the points. Also, I use the context package and also matplotlib package for the plots.



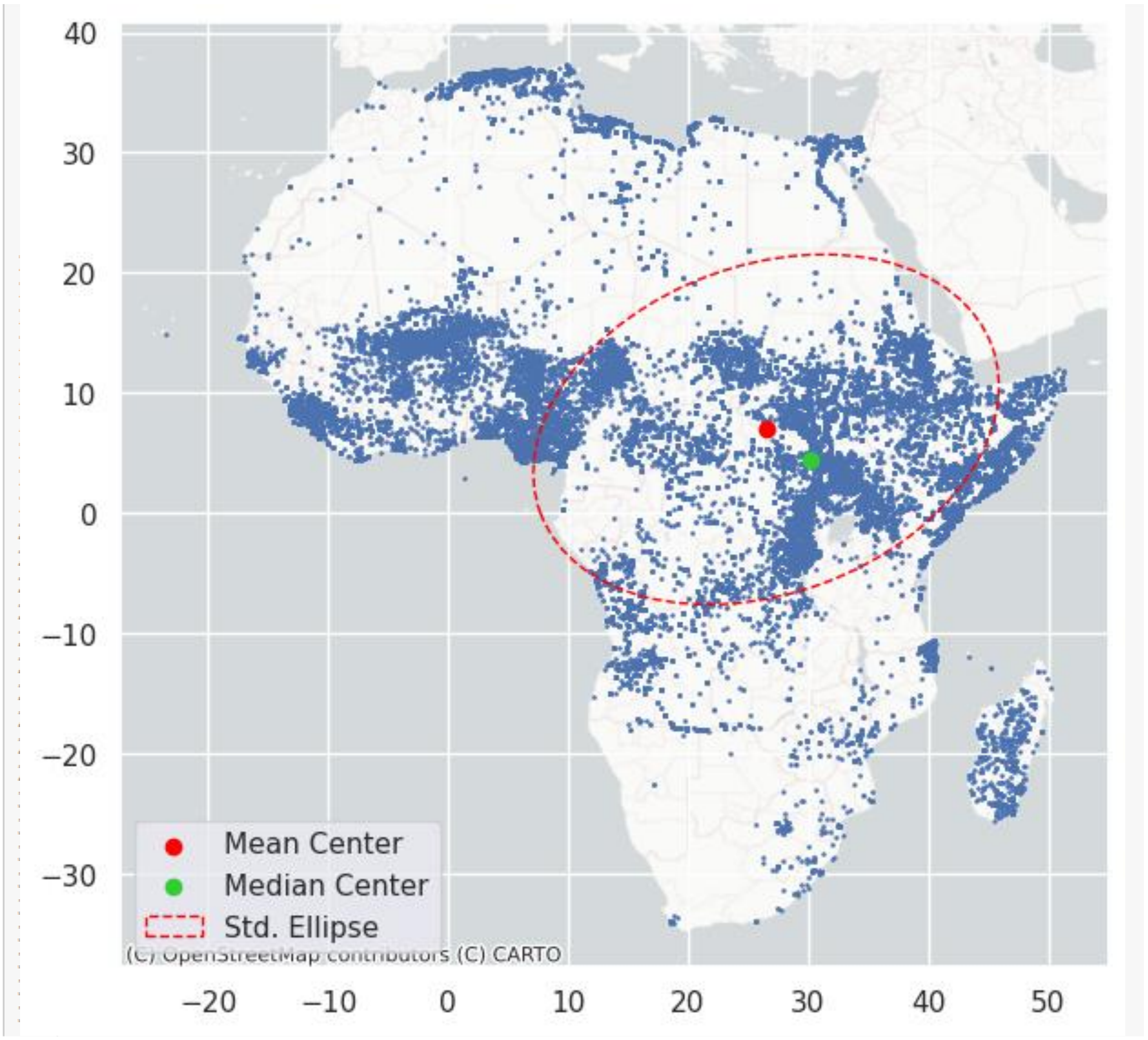
```python
import seaborn as sns
import contextily
import matplotlib.pyplot as plt
plot = sns.jointplot(x="x", y="y", data=pp.df)
contextily.add_basemap( plot.ax_joint, crs="EPSG:4326",
                       source=contextily.providers.CartoDB.PositronNoLabels)
plt.savefig("cluster.png")
```

The above map which goal is to identify the different clusters, by just seeing the histogram. Based on this map, we can see that there are many clusters in the middle of Africa and if we come now to the x-axis we can see that the middle and the east observation are higher than all the rest followed by the center.

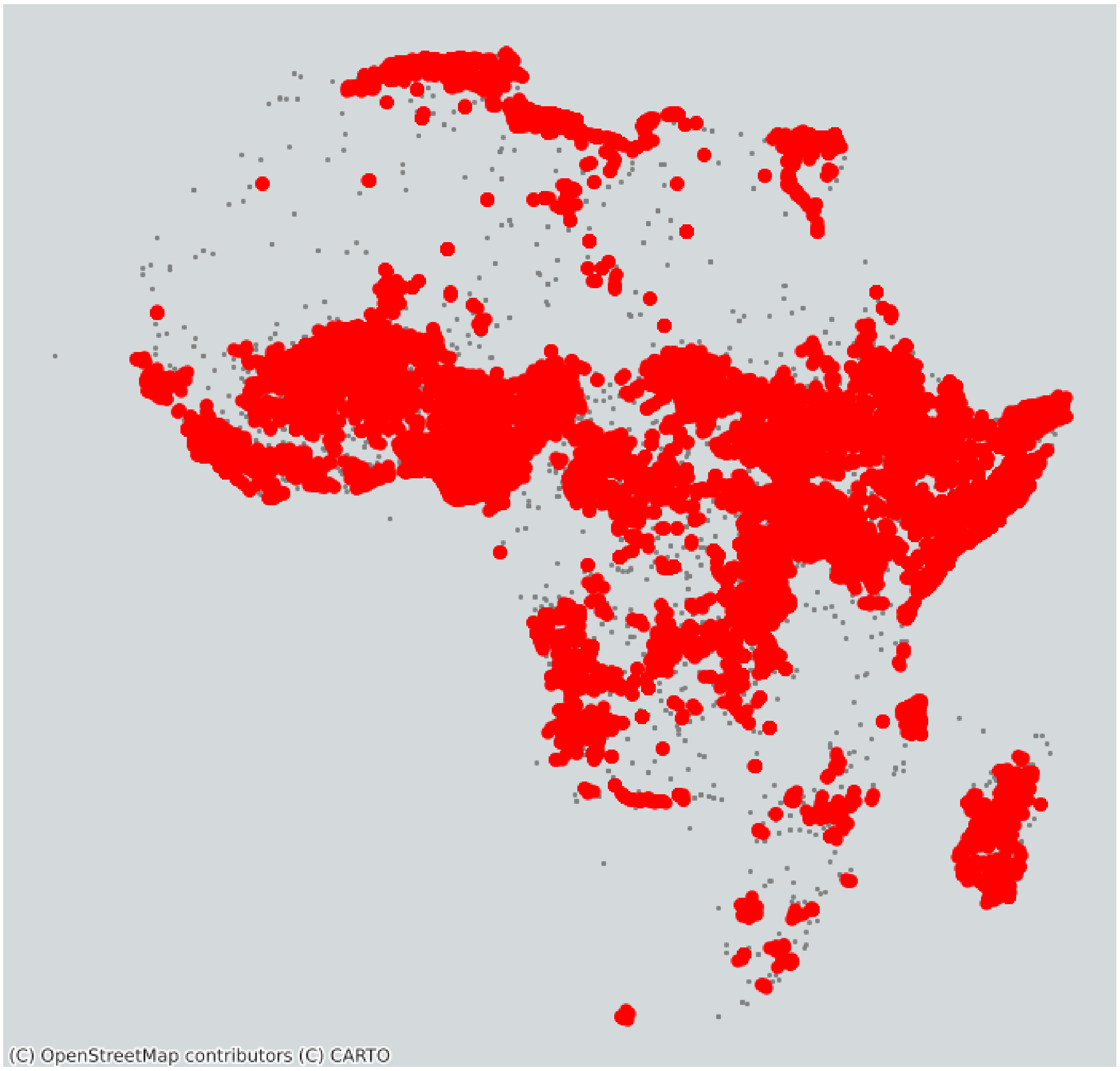


For the pattern point analysis, any method can be adopted but here in this part, I am using in this section the centrography method. This method is the one that helps us to show some basic statistics of the point data. In my work I focused on the mean center which is the computation of the average x and y coordinate value, I am interested on the median center which is a tool to measure the central tendency. And at the end of all, which is the separate distance for each axis.

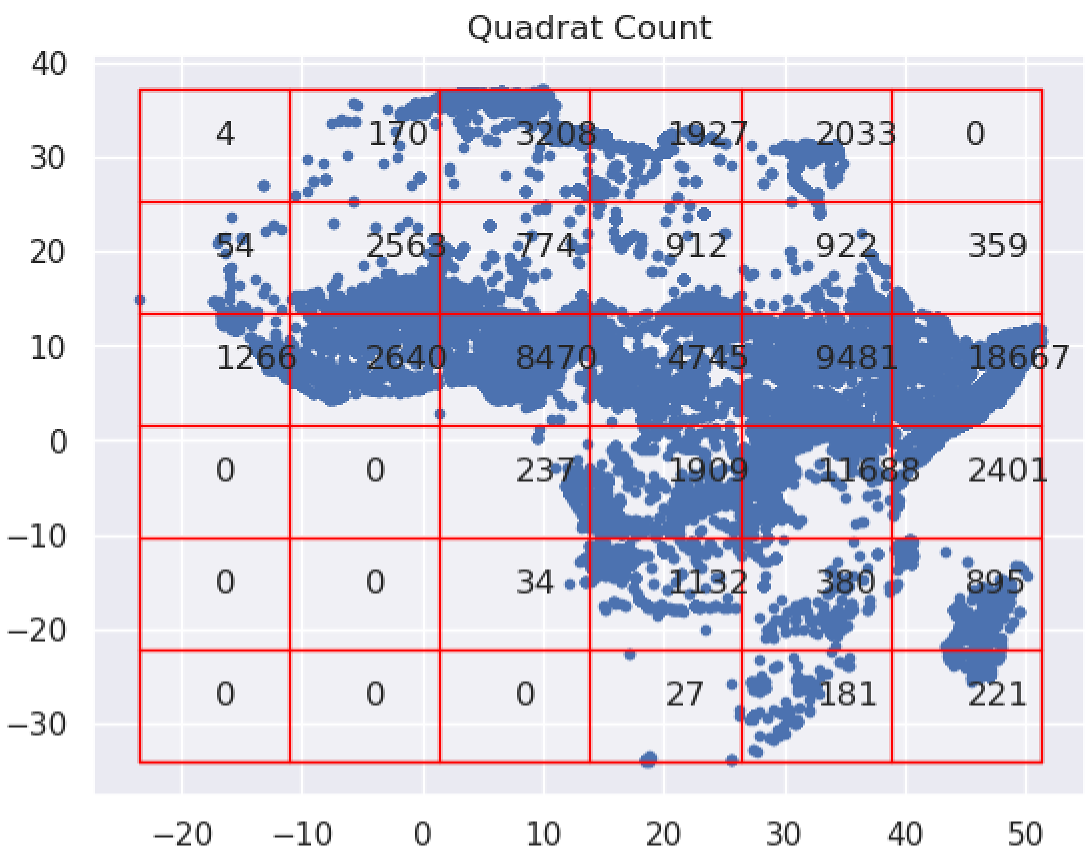
```python
from matplotlib.patches import Ellipse
import numpy
import contextily

# Set up figure and axis
f, ax = plt.subplots(1, figsize=(7, 7))
# Plot photograph points
ax.scatter(pp.df["x"], pp.df["y"], s=0.75)

contextily.add_basemap(ax, crs="EPSG:4326",
                       source=contextily.providers.CartoDB.PositronNoLabels)

ax.scatter(*mean_center, color="red", marker="o", label="Mean Center")
ax.scatter(*med_center, color="limegreen", marker="o", label="Median Center")
#ax.scatter(std_distance, color="yellow", label="standard distance")

# Construct the standard ellipse using matplotlib

ellipse = Ellipse(
    xy=mean_center,  # center the ellipse on our mean center
    width=major * 2,  # centrography.ellipse only gives half the axis
    height=minor * 2,
    angle=numpy.rad2deg(
        rotation
    ),  # Angles for this are in degrees, not radians
    facecolor="none",
    edgecolor="red",
    linestyle="--",
    label="Std. Ellipse",
)
ax.add_patch(ellipse)


ax.legend()

plt.savefig("ellipse.png")

plt.show()
```



Also, in this case, I use the DBSCAN can method to identify clusters. This help to identify in multiple dimension the clusters. So, after implementing the algorithm, we can see that the red color help to distinguish the clusters from the other observation which are not clusters.

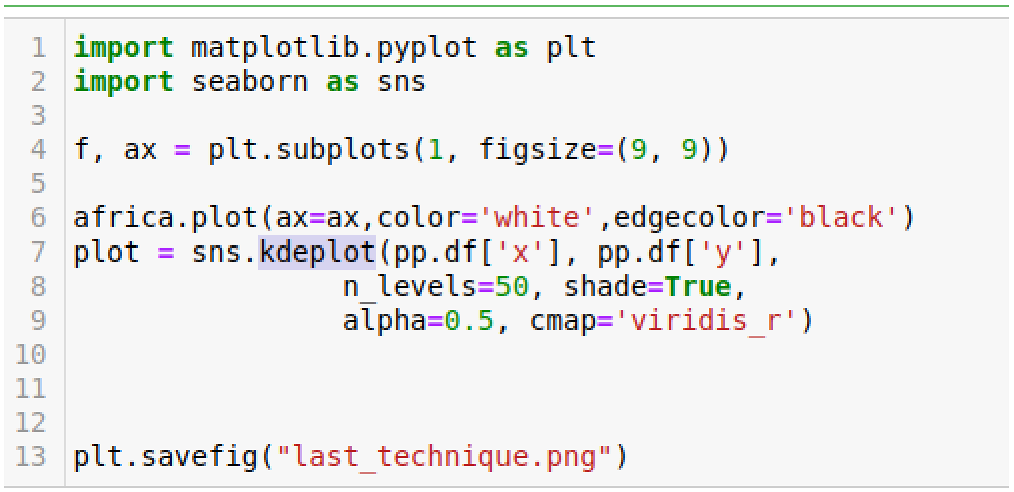


Also, to know more about the clusters, it’s better to do a quadrat plot which can allow us to know the number of observations in each quadrat. This method is better because for me, I can see among the clusters, the main clusters and identify them by the classification of the numbers.

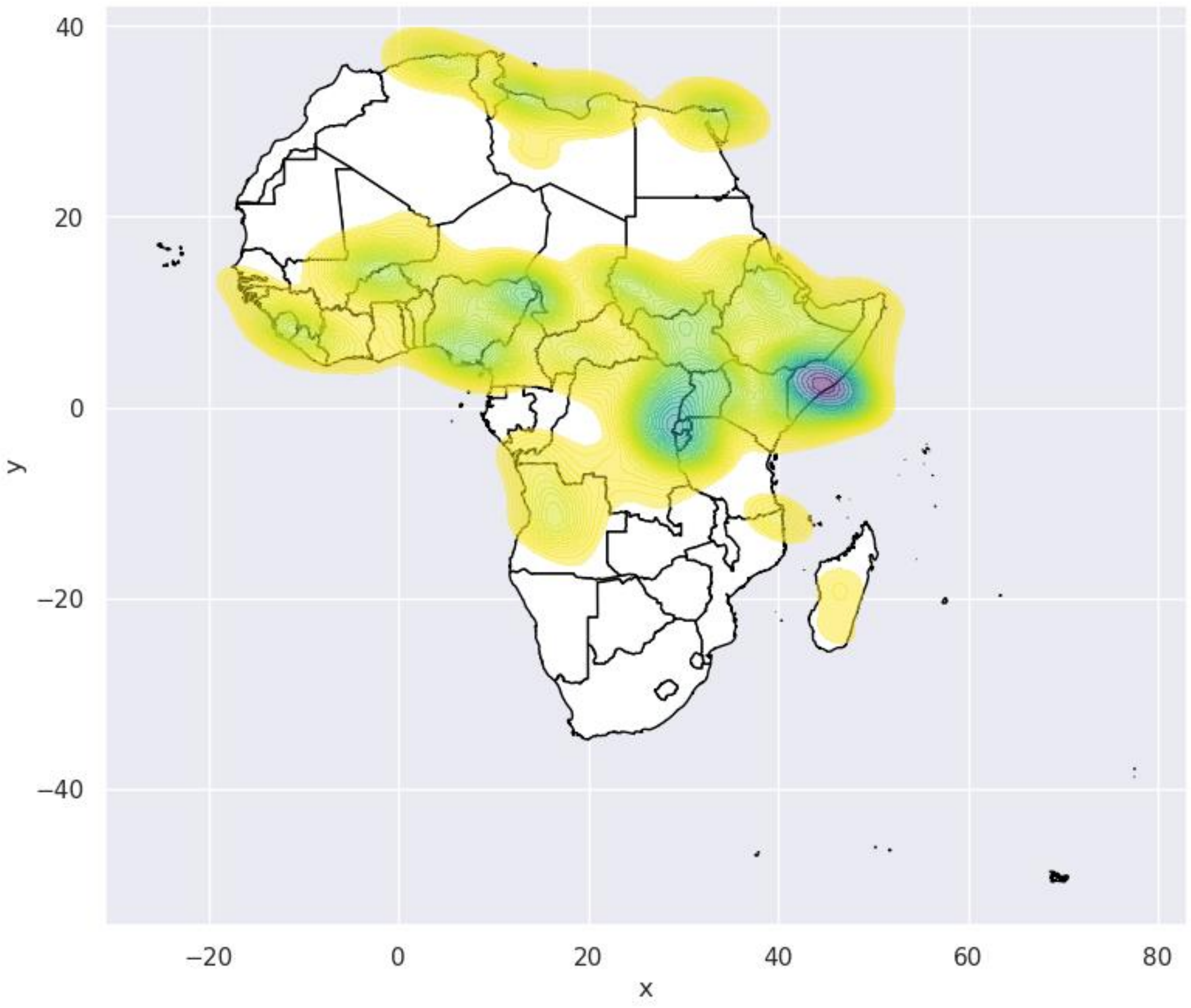


1.9:Create a map showing the main clusters of observations.

For this, A kernel density estimate (KDE) plot is a method for visualizing the distribution of observations in a dataset. So, I use it to show the main cluster of the observation.

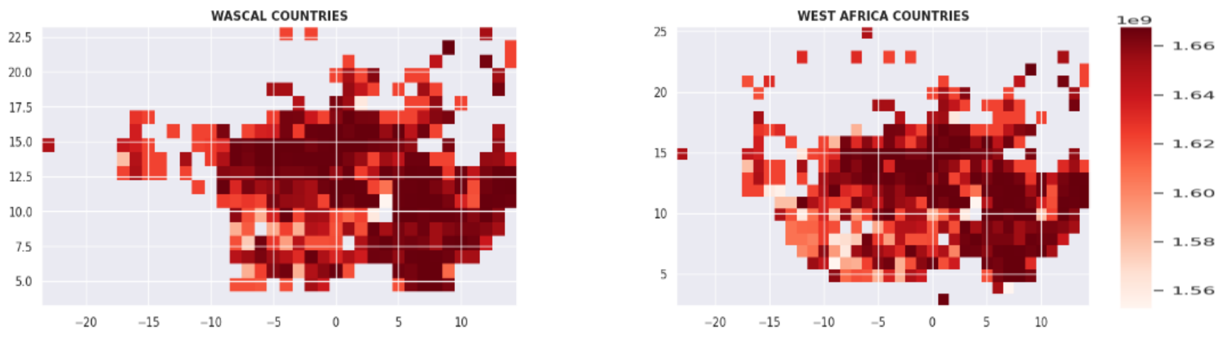


```
import matplotlib.pyplot as plt
import seaborn as sns

f, ax = plt.subplots(1, figsize=(9, 9))

africa.plot(ax=ax,color='white',edgecolor='black')
plot = sns.kdeplot(pp.df['x'], pp.df['y'],
                   n_levels=50, shade=True,
                   alpha=0.5, cmap='viridis_r')



plt.savefig("last_technique.png")
```



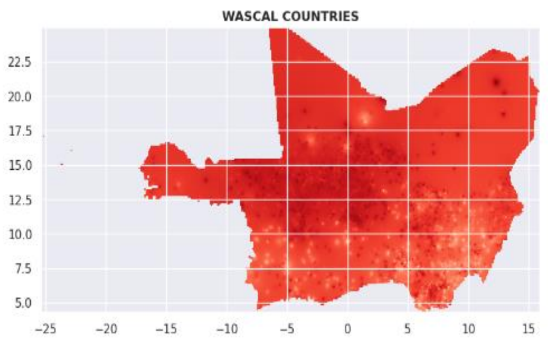
1.10.1 - Extract all those observations for (i) West African countries (16 countries!) and for (ii) the WASCAL member countries. Create a raster of the Battle for each of the two regions (i and ii) using spatial interpolation and validate your result.

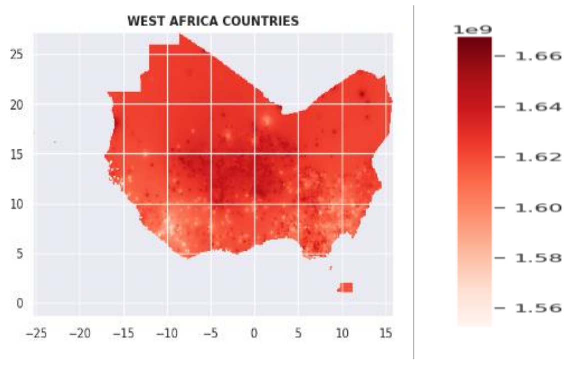
from my dataset, I select all the WASCAL countries in one file and also in the one another file I selected the West African countries. Then I use QGIS to do the raster using the spatial interpolation.



1.10.2 - Create two maps showing your results. Explain the results.

To do so, I sued the same method like the first part, Therefore, I use a loc function to select the different countries and created a new csv file based on it. I did the work in QGIS which I master a bit, just to find the simplest way to do my work, and after I imported my result in python where I save the result and displayed it. We can see that country like, Togo, Ghana, Benin, and Cabo Verde the battle is very low but in another West African country that is also WASCAL countries the parasite rate is higher.



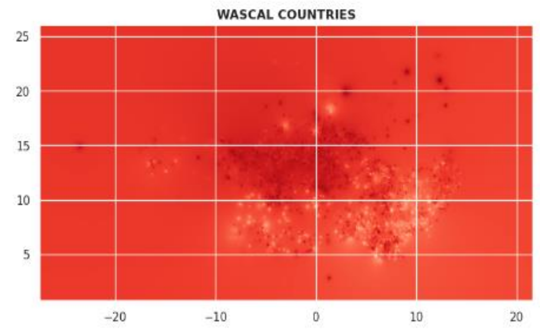


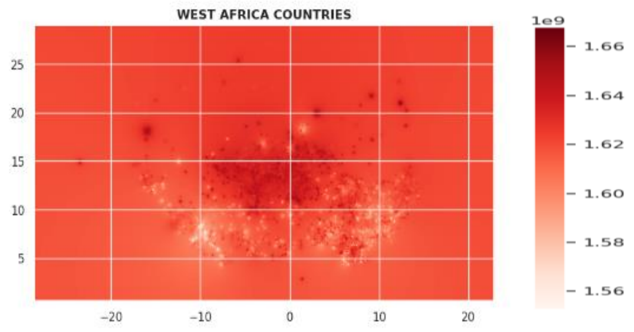
1.10.3: Discuss possible errors in your interpolation.

The error in this interpolation is that, as we can see just the highest value is given to the inner countries, but the coastal countries, the estimation by interpolation shows that the battle is not high in this area but when we come to the data itself which is the plot at the down (bar plot for visualization), countries like Senegal, Nigeria, Ivory coast have a big value in term of battle.

1.10.4 - Export your results as spatial rasters.

This is a raster I made, for west African countries and also, for WASCAL countries. The rasters are a grid of regularly sized pixels, they are showing continually varying information.





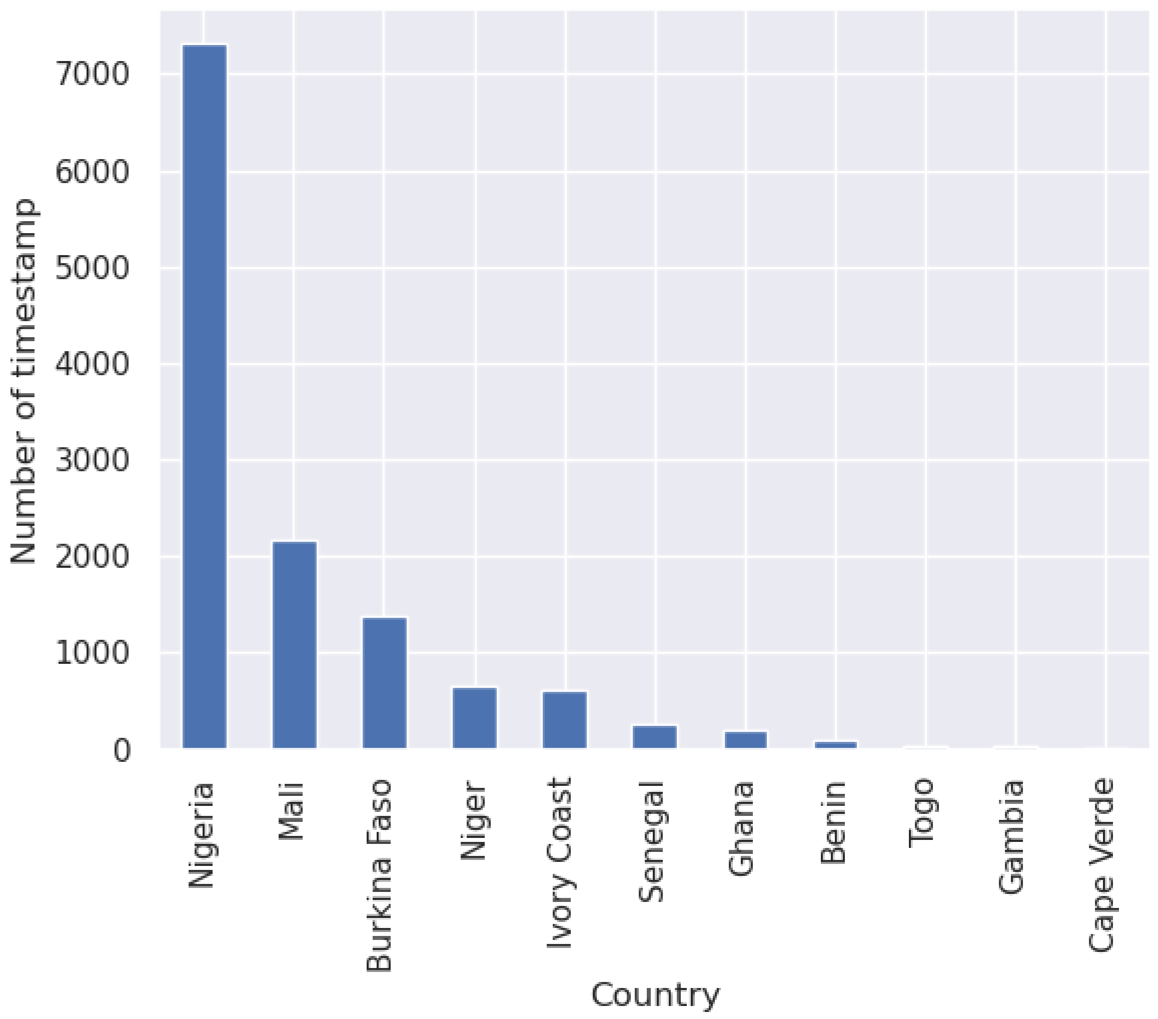
1.11: Extract the average battle for each WASCAL member country.

For this question, the plot below is showing the result. The mean average cannot be possible but we can count the number of battle incidents for each country and rank them.

1.12 - Rank these countries by battle and visualize this ranking in a plot.

Regarding the dataset I am using which is a conflict dataset gathered from the ACLED database, we extract the west African countries, therefore we group the country by the same event which is Battle. We can clearly see hear that countries like Nigeria had the biggest battle event in the wascal's countries and followed by Mali and Also Burkina Faso. In some countries like Cabo Verde and the Gambia, the Battle event are almost null.

There for the largest number of Nigeria and Mali ahead Burkina Faso should not be something that should surprise us. Because those countries have known battle many time before the recent terrorism situation in Burkina Faso



Number of timestamp
7000
6000
5000
4000
3000
2000
1000
0
Nigeria
Mali
Burkina Faso
Niger
Ivory Coast
Senegal
Ghana
Benin
Togo
Gambia
Cape Verde
Country