WASCAL Master Research Program on Informatics for Climate Change

Spatial Data Analysis 2020 – 2021

May 2021

Class Project

Lecturer

Dr. Sarah Schönbrodt-Stitt

Department of Remote Sensing

University of Würzburg

Germany

sarah.schoenbrodt-stitt@uni-wuerzburg.de

Lecturer

Steven HILL

Department of Remote Sensing

University of Würzburg

Germany

Steven.hill@uni-wuerzburg.de

Student

Zourkalaini BOUBAKAR

WASCAL MRP Informatics for Climate Change

Postgraduate School of Applied Science

University of Prof. Joseph Ki-Zerbo

Ouagadougou, Burkina Faso

Tankostick@gmail.com

Table of Contents

I. Data and Methodology

For this work, a folder containing all needed datasets are provided by the Lectures. These are Nigeria fires datasets (csv file), Nigeria local government area datasets (shape files), the elevation datasets for Burkina Faso (shape file) and the SRTM Digital Elevation Model for Burkina Faso (raster file, in Tif format). All calculations and the plotting have been done with **Python programming language** (version 3) through **Jupyther Notebook** Integrated Development Environment, as a web-based interactive development environment of Python. As resources, lectures materials are used.

First, Nigeria fires datasets and Nigeria local government area datasets are analyzed to depict fires over Nigeria country through **Exercise 1**. Further in this work, results of the analysis are discussed. Lastly the elevation datasets for Burkina Faso is analyzed and results are discussed in **Exercise 2**.

II. Results and Discussion

1. Exercise1

1.1

How many fires were detected in total?

```
import pandas as pd
dataset_fires = pd.read_csv('../SpatialAnalysis2021/Data/non-spatial/
fire/fire.csv')
nb_fires = dataset_fires.shape[0]
print(f'Number of detected fires: {nb_fires}')
```

Number of detected fires: 21460

1.2

During which time period were the fires detected?

```
detected_date = (dataset_fires.ACQ_DATE).unique()
print(detected_date)
print(f'Fires were detected between {detected_date[0]} and
                                    {detected_date[-1]}')
```

```
['2019-01-01'  '2019-01-02'  '2019-01-03'  '2019-01-04'  '2019-01-05'
 '2019-01-06'  '2019-01-07'  '2019-01-08'  '2019-01-09'  '2019-01-10'
 '2019-01-11'  '2019-01-12'  '2019-01-13'  '2019-01-14'  '2019-01-15'
 '2019-01-16'  '2019-01-17'  '2019-01-18'  '2019-01-19'  '2019-01-20'
 '2019-01-21'  '2019-01-22'  '2019-01-23'  '2019-01-24'  '2019-01-25'
 '2019-01-26'  '2019-01-27'  '2019-01-28'  '2019-01-29'  '2019-01-30'
 '2019-01-31']
```

Fires were detected from 2019-01-01 to 2019-01-31

1.3

Identify the ten brightest fires during this period. Where are they located?

```
import geopatra
import geopandas as gpd
bright = dataset_fires.sort_values('BRIGHTNESS', ascending=False).head(10)
gdf_brigth = gpd.GeoDataFrame(bright, geometry=gpd.points_from_xy(bright.L
ONGITUDE, bright.LATITUDE))

gdf_bright
```

Numbers	Longitude	Latitude	Brightness
01	**9.676**	**10.613**	**422.9**
02	**10.5655**	**6.7566**	**421.8**
03	**8.1329**	**11.187**	**419.7**
04	**7.6403**	**8.1262**	**418**
05	**12.446**	**12.446**	**413.4**
06	**7.6324**	**8.3984**	**405.4**
07	**8.6089**	**10.527**	**399.9**
08	**9.6863**	**10.6208**	**397**
09	**11.3494**	**13.5145**	**396.7**
10	**10.0935**	**3.8743**	**995.2**

The figure below shows where these ten fires are located

Note: we can notice that 9 locations are showed by the map. Indeed, among the ten brightness fires, the first (**9.676, 10.613**) and eight (**9.6863, 10.6208**) fires are very closed to each other.

1.4

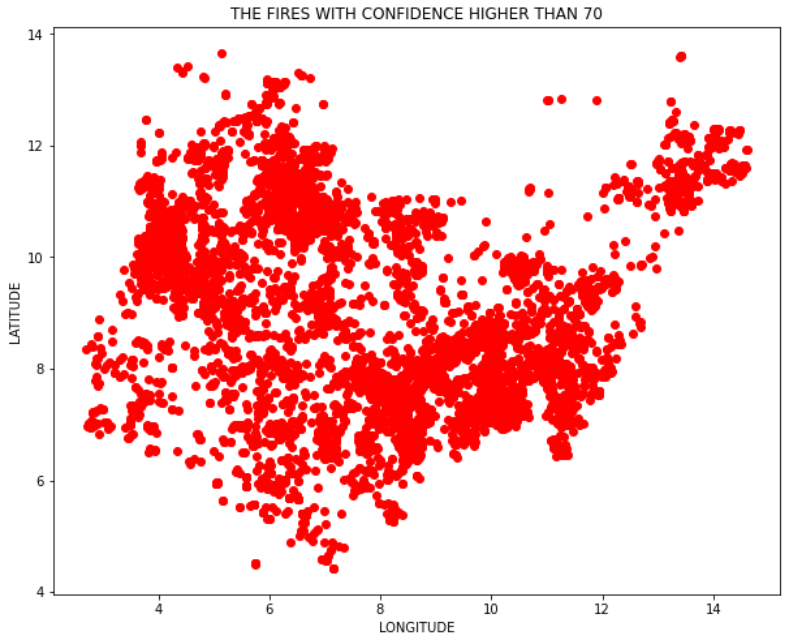
Extract all fires detected with a confidence higher than 70 percent. How much are these? Show in a plot

```
confid_fires = dataset_fires[dataset_fires['CONFIDENCE'] > 70]
confid_fires

print(f'Number of detected fires with confidence higher than 70:
      {confid_fires.shape[0]}')
```

Number of detected fires with confidence higher than 70: 7285

The map below shows these fires with confidence higher than 70.

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 8))
gdf_confid_fires = gpd.GeoDataFrame(confid_fires, geometry=gpd.points_fr
om_xy(confid_fires.LONGITUDE, confid_fires.LATITUDE))
gdf_confid_fires_plot = gdf_confid_fires.folium.plot()
bright_plot
#plt.scatter(dataset_fires.LONGITUDE, dataset_fires.LATITUDE, color='gra
y')
plt.scatter(confid_fires.LONGITUDE, confid_fires.LATITUDE, color='red')
plt.xlabel("LONGITUDE")
plt.ylabel("LATITUDE")
plt.title("THE FIRES WITH CONFIDENCE HIGHER THAN 70")
plt.show()
```

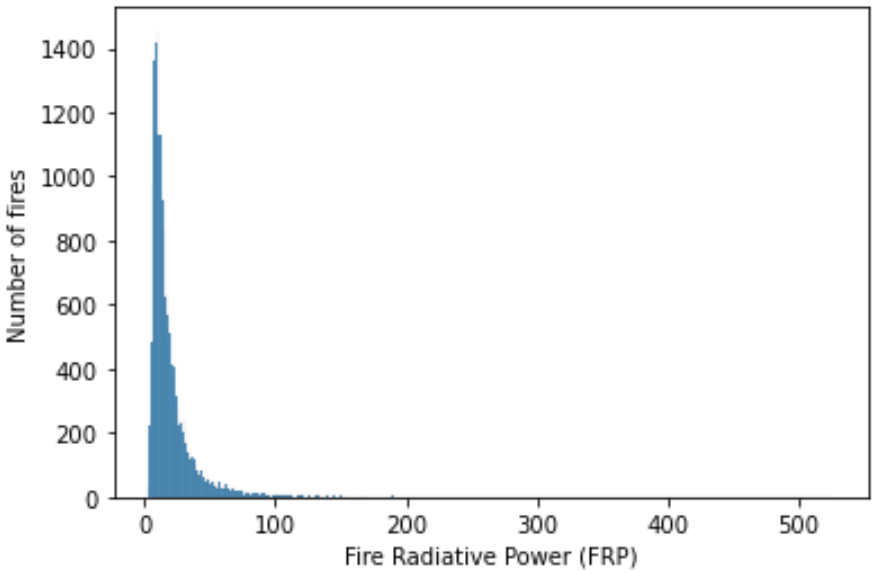


1.5

Create a histogram showing the distribution of the fire radiative power. Cleary indicate the description of the y-axis and x-axis.

The figure below shows the distribution of radiative power.

```
import seaborn as sns
ax = sns.histplot(dataset_fires['FRP'])
ax.set(xlabel='Fire Radiative Power (FRP)', ylabel='Number of fires')
```



The FRP ranges from 3 to 526 with an average of 20.

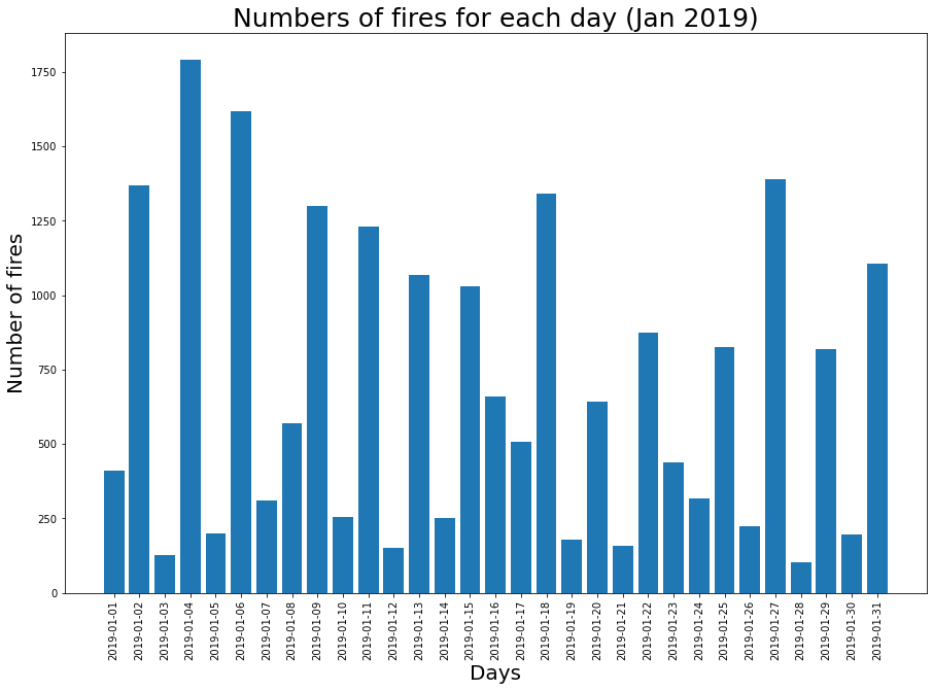
In this plot, we can notice that serious fires with FRP greater than 100 are represented by 1.22% of the total fires.

1.6

Create a plot showing the numbers of fires for each day. Convert the dataset into a spatial geopandas data frame

The plot below shows the number of fires for each day during January 2019.

```
day_x = (dataset_fires.ACQ_DATE).unique()#days
#number of occurence of each day
y = []
for i in range(0, 31):
    y.append((list(dataset_fires.ACQ_DATE)).count(day_x[i]))
#plotting
plt.figure(figsize=(15, 10))
plt.bar(day_x, y)
plt.xticks(rotation=90)
plt.title("Numbers of fires for each day (Jan 2019)", fontsize=25)
plt.xlabel("Days", fontsize=20)
plt.ylabel("Number of fires", fontsize=20)
plt.show()
```



We can notice with this plot that avec a total record of **21460** fires during January 2009, an average of 692 fires occurred each day.

The following lines allow us to convert the dataset (pandas data frame) into geospatial data frame.

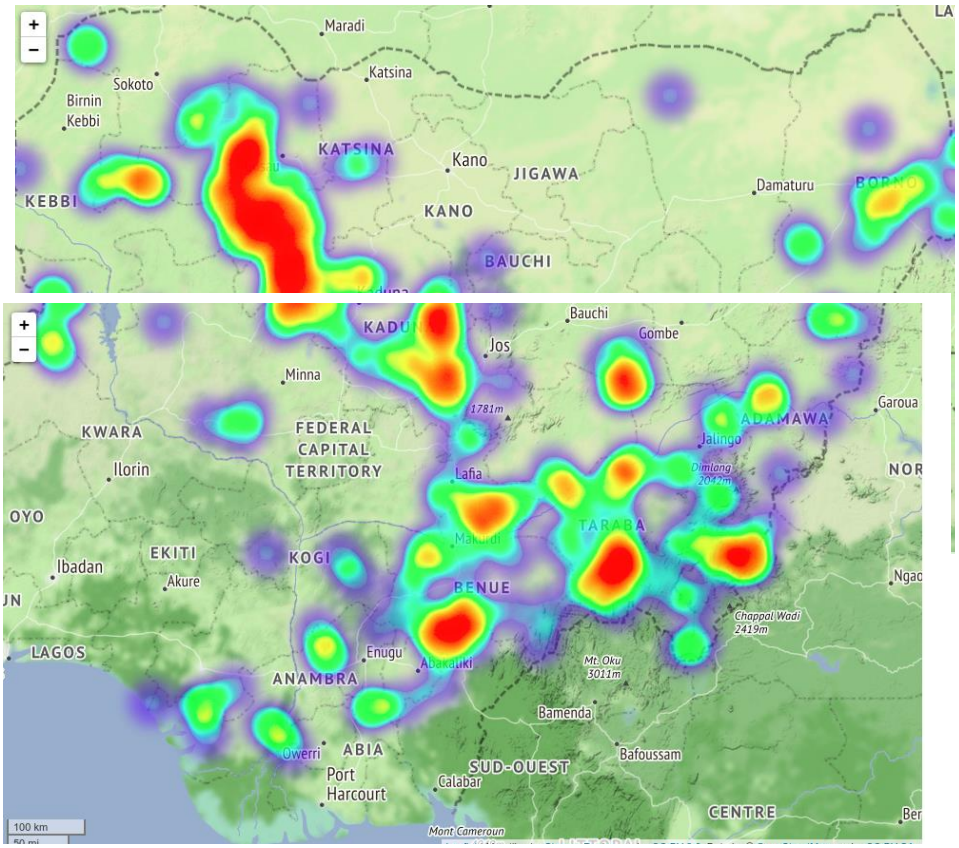
```
dataset_fires_gdf = gpd.GeoDataFrame(dataset_fires, geometry=gpd.points_
from_xy(dataset_fires.LONGITUDE, dataset_fires.LATITUDE))
```

1.7

Create a heat map showing the density of fires in Nigeria

```
import numpy as np
import folium
from folium.plugins import HeatMap

lat = np.array(dataset_fires['LATITUDE'])
lon = np.array(dataset_fires['LONGITUDE'])
js = np.array(y, dtype=float)
folium.Figure(width=1200, height=1200)
data1 = [[lat[i].mean(), lon[i].mean(), js[i]] for i in range (491)]
map_fire = folium.Map(location=[lon.max(), lat.max()], zoom_start=10, ti
les='Stamen Terrain', control_scale=True)
(HeatMap(data1).add_to(map_fire)).show
map_fire
```

The figure above is a heat map showing the density of fires in Nigeria.



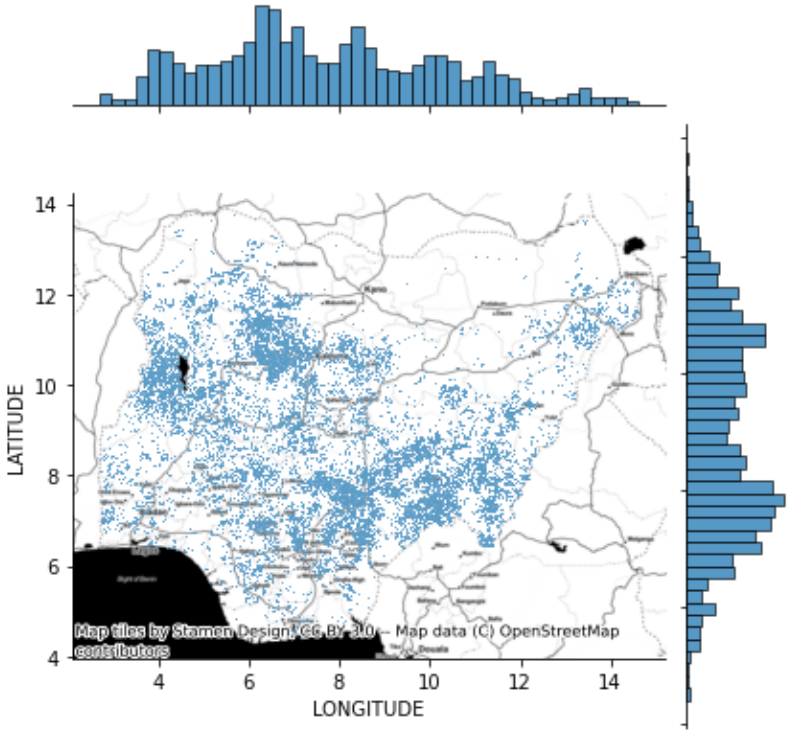
The monthly Fire Density (FD) was calculated by dividing the number of fires in the area by the surface (km2) of the vegetation and region considered

1.8

Investigate if the fires detected in Nigeria are distributed rather regular or if the fires appear rather clustered (if so try to identify the main clusters and explain). Import the local area government dataset (nigeria_lga.shp)

First, to get sense of what the spatial dimension of the dataset looks like is to plot it. At its most basic level, we can generate a scatter plot with seaborn, and provide additional context by overlaying a tile map from the internet, using contextily with jointplot.

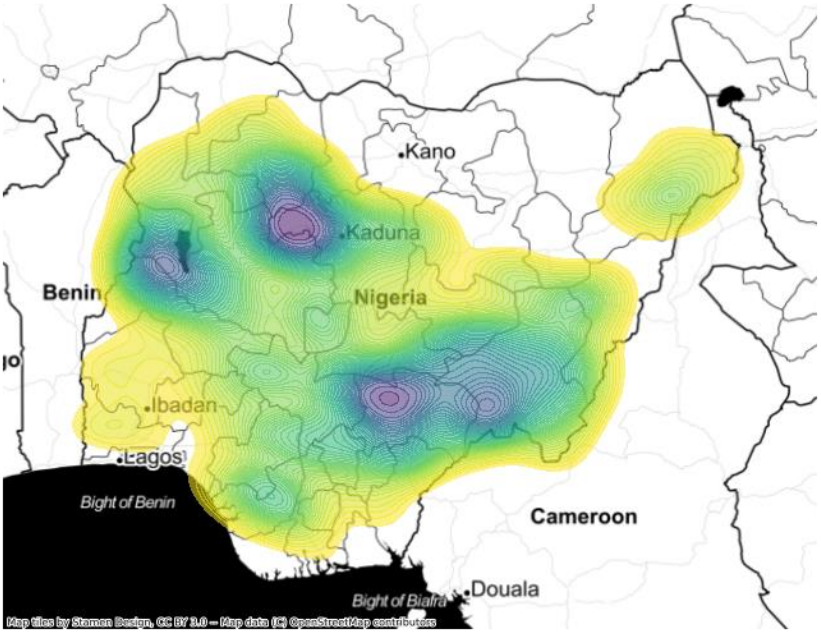
```
import contextily

joint_axes = sns.jointplot(x='LONGITUDE', y='LATITUDE', data=dataset_fir
es, s=1)

contextily.add_basemap(
    joint_axes.ax_joint,
    crs="EPSG:4326",
    source=contextily.providers.Stamen.Toner
)
```



We can see that dots tend to be concentrated in the center of the covered area. Furthermore, whiten the broad pattern, we can see there seems to **be more localized clusters**.

Despite there are localized clusters, there are areas where the density of points is so large that plotting opaque dots on top of one another. We can recast this approach as a spatial histogram using **Kernel Density Estimation** so that we can **identify the main clusters (KDE)**. KDE is an empirical approximation of the probability density function. It lays a grid of points over the space of interest on which it places kernel functions that count points around them with different weight based on the distance. These counts are then aggregated to generate a global surface with probability. From main clusters to less important ones, colors go from violet, then green up to yellow (reverse viridis colormap).

```
f, ax = plt.subplots(1, figsize=(12, 9))
sns.kdeplot(
    dataset_fires['LONGITUDE'],
    dataset_fires['LATITUDE'],
    n_levels=50,
    shade=True,
    alpha=0.55,
    cmap='viridis_r'
)
contextily.add_basemap(
    ax,
    source=contextily.providers.Stamen.Toner, crs='EPSG:4326'
)
ax.set_axis_off()
```



The following lines allow us to import Nigeria local government area dataset.

```
nigeria_lga_gpdf = gpd.read_file('../SpatialAnalysis2021/Data/vector/fires/new_lga_nigeria_2003.shp')
```

1.9

Count the fires occurring in the different local government areas in Nigeria. How much are they?

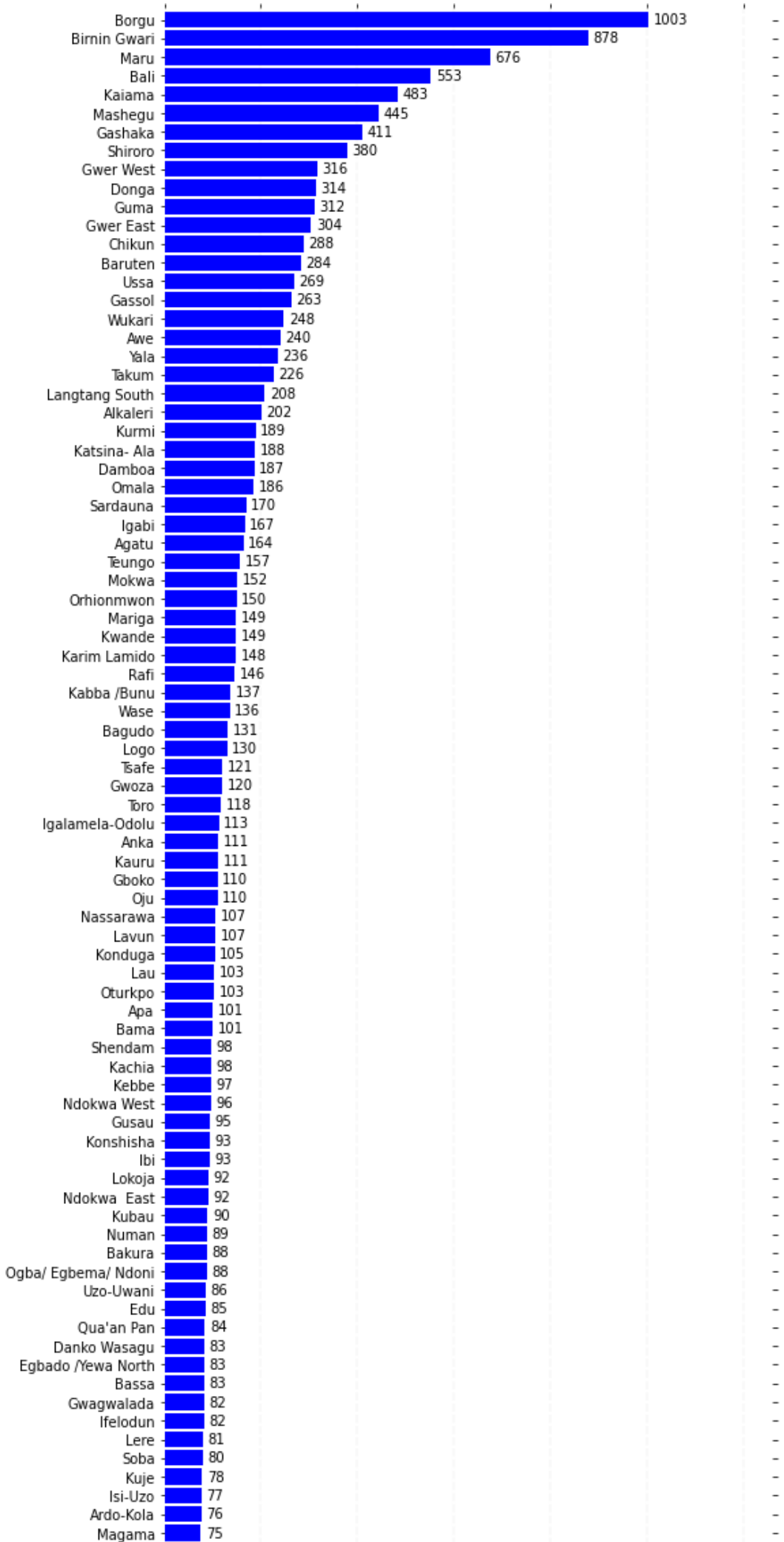
```
import numpy as np
link = gpd.sjoin(nigeria_lga_gpdf.to_crs('EPSG:4326'),dataset_fires_gdf)
y = link['LGA'].value_counts(ascending=True)

link = gpd.sjoin(nigeria_lga_gpdf.to_crs('EPSG:4326'),dataset_fires_gdf)

ax = y.plot(kind='barh', figsize=(8, 120), color='blue', zorder=2, width=0.85)
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.spines['left'].set_visible(False)
ax.spines['bottom'].set_visible(False)
```

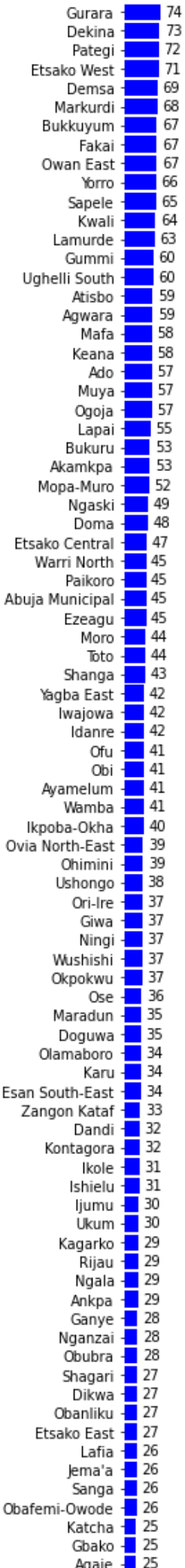
```
# Switch off ticks
ax.tick_params(axis="both", which="both", bottom="off", top="off", labelbottom="on", left="off", right="off", labelleft="on")
```

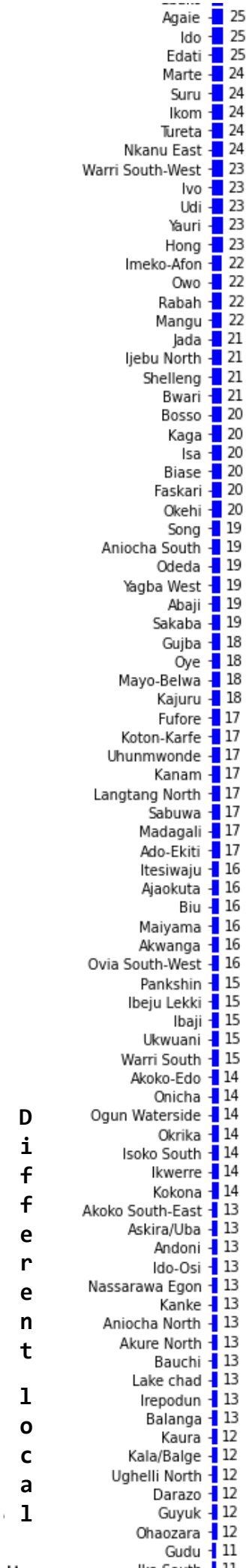
```
# Draw vertical axis lines
vals = ax.get_xticks()
for tick in vals:
    ax.axvline(x=tick, linestyle='dashed', alpha=0.4, color='#eeeeee', zorder=1)
```

```
# Set x-axis label
ax.set_xlabel("Fires occurring", labelpad=20, weight='bold', size=12)

# Set y-axis label
ax.set_ylabel("different local government areas", labelpad=20, weight='b
old', size=12)
for i, v in enumerate(y):
    ax.text(v + 10, i, str(v), color='black', fontsize=10, ha='left', va
='center')
```

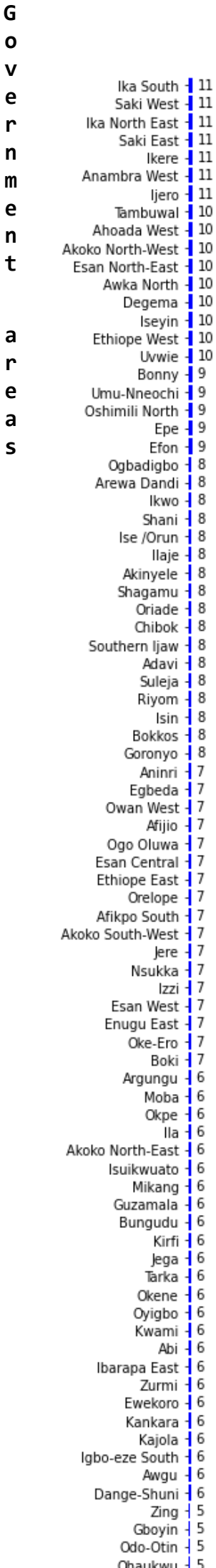
The figure bellow shows the number of occurring fires in Nigeria different local government area.


Borgu 1003
Birnin Gwari 878
Maru 676
Bali 553
Kaiama 483
Mashegu 445
Gashaka 411
Shiroro 380
Gwer West 316
Donga 314
Guma 312
Gwer East 304
Chikun 288
Baruten 284
Ussa 269
Gassol 263
Wukari 248
Awe 240
Yala 236
Takum 226
Langtang South 208
Alkaleri 202
Kurmi 189
Katsina- Ala 188
Damboa 187
Omala 186
Sardauna 170
Igabi 167
Agatu 164
Teungo 157
Mokwa 152
Orhionmwon 150
Mariga 149
Kwande 149
Karim Lamido 148
Rafi 146
Kabba /Bunu 137
Wase 136
Bagudo 131
Logo 130
Tsafe 121
Gwoza 120
Toro 118
Igalamela-Odolu 113
Anka 111
Kauru 111
Gboko 110
Oju 110
Nassarawa 107
Lavun 107
Konduga 105
Lau 103
Oturkpo 103
Apa 101
Bama 101
Shendam 98
Kachia 98
Kebbe 97
Ndokwa West 96
Gusau 95
Konshisha 93
Ibi 93
Lokoja 92
Ndokwa East 92
Kubau 90
Numan 89
Bakura 88
Ogba/ Egbema/ Ndoni 88
Uzo-Uwani 86
Edu 85
Qua'an Pan 84
Danko Wasagu 83
Egbado /Yewa North 83
Bassa 83
Gwagwalada 82
Ifelodun 82
Lere 81
Soba 80
Kuje 78
Isi-Uzo 77
Ardo-Kola 76
Magama 75

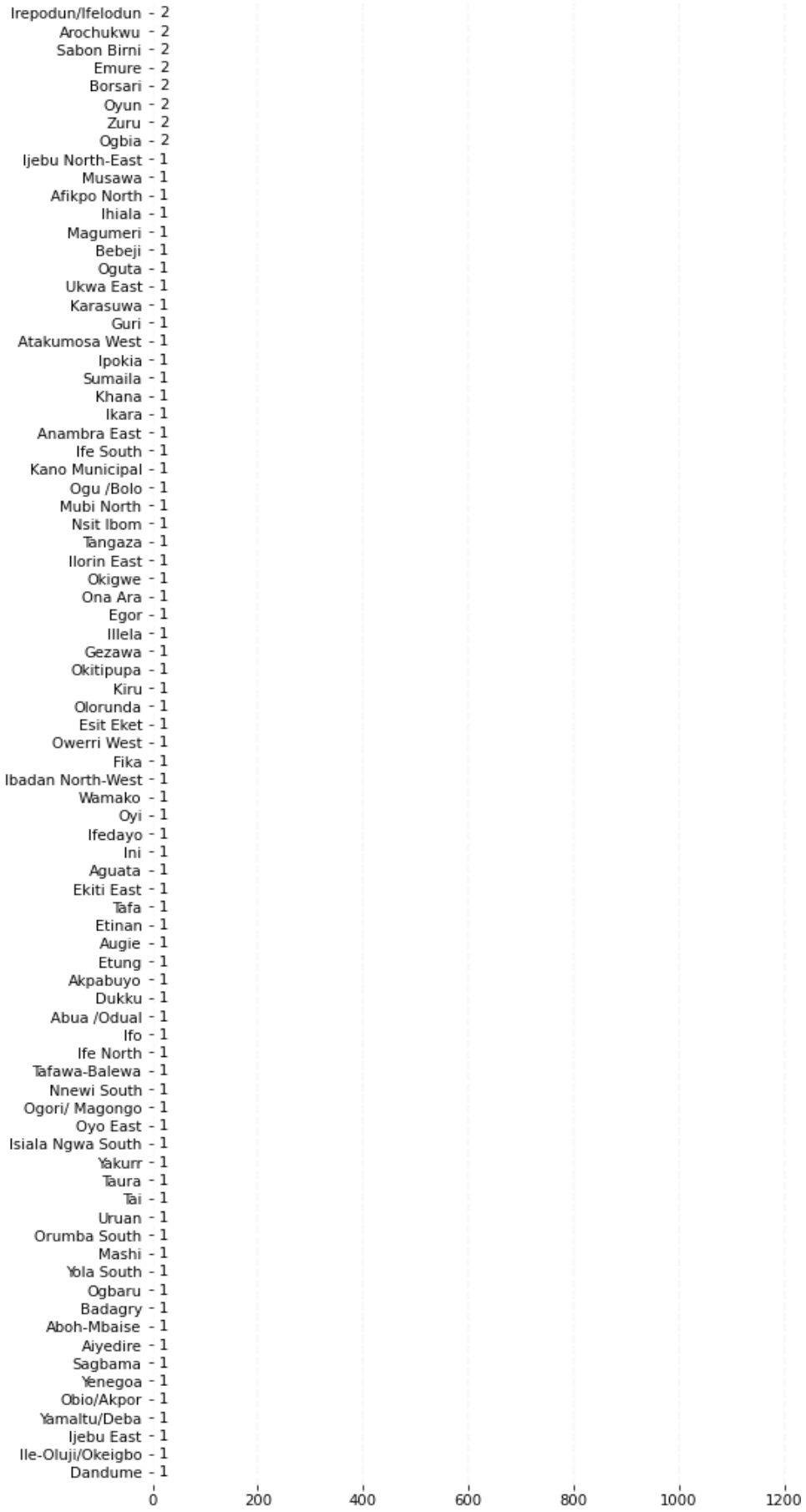

Gurara 74
Dekina 73
Pategi 72
Etsako West 71
Demsa 69
Markurdi 68
Bukkuyum 67
Fakai 67
Owan East 67
Yorro 66
Sapele 65
Kwali 64
Lamurde 63
Gummi 60
Ughelli South 60
Atisbo 59
Agwara 59
Mafa 58
Keana 58
Ado 57
Muya 57
Ogoja 57
Lapai 55
Bukuru 53
Akamkpa 53
Mopa-Muro 52
Ngaski 49
Doma 48
Etsako Central 47
Warri North 45
Paikoro 45
Abuja Municipal 45
Ezeagu 45
Moro 44
Toto 44
Shanga 43
Yagba East 42
Iwajowa 42
Idanre 42
Ofu 41
Obi 41
Ayamelum 41
Wamba 41
Ikpoba-Okha 40
Ovia North-East 39
Ohimini 39
Ushongo 38
Ori-Ire 37
Giwa 37
Ningi 37
Wushishi 37
Okpokwu 37
Ose 36
Maradun 35
Doguwa 35
Olamaboro 34
Karu 34
Esan South-East 34
Zangon Kataf 33
Dandi 32
Kontagora 32
Ikole 31
Ishielu 31
Ijumu 30
Ukum 30
Kagarko 29
Rijau 29
Ngala 29
Ankpa 29
Ganye 28
Nganzai 28
Obubra 28
Shagari 27
Dikwa 27
Obanliku 27
Etsako East 27
Lafia 26
Jema'a 26
Sanga 26
Obafemi-Owode 26
Katcha 25
Gbako 25
Agaie 25



Different local


Government areas
Ika South 11
Saki West 11
Ika North East 11
Saki East 11
Ikere 11
Anambra West 11
Ijero 11
Tambuwal 10
Ahoada West 10
Akoko North-West 10
Esan North-East 10
Awka North 10
Degema 10
Iseyin 10
Ethiope West 10
Uvwie 10
Bonny 9
Umu-Nneochi 9
Oshimili North 9
Epe 9
Efon 9
Ogbadigbo 8
Arewa Dandi 8
Ikwo 8
Shani 8
Ise /Orun 8
Ilaje 8
Akinyele 8
Shagamu 8
Oriade 8
Chibok 8
Southern Ijaw 8
Adavi 8
Suleja 8
Riyom 8
Isin 8
Bokkos 8
Goronyo 8
Aninri 7
Egbeda 7
Owan West 7
Afijio 7
Ogo Oluwa 7
Esan Central 7
Ethiope East 7
Orelope 7
Afikpo South 7
Akoko South-West 7
Jere 7
Nsukka 7
Izzi 7
Esan West 7
Enugu East 7
Oke-Ero 7
Boki 7
Argungu 6
Moba 6
Okpe 6
Ila 6
Akoko North-East 6
Isuikwuato 6
Mikang 6
Guzamala 6
Bungudu 6
Kirfi 6
Jega 6
Tarka 6
Okene 6
Oyigbo 6
Kwami 6
Abi 6
Ibarapa East 6
Zurmi 6
Ewekoro 6
Kankara 6
Kajola 6
Igbo-eze South 6
Awgu 6
Dange-Shuni 6
Zing 5
Gboyin 5
Odo-Otin 5
Obaukwu 5


Odo-Otin 5
Ohaukwu 5
Jos South 5
Billiri 5
Ibeno 5
Ezza South 5
Bunza 5
Birnin-Kebbi 5
Oluyole 5
Abakaliki 5
Ohaji /Egbema 5
Maiha 5
Barkin Ladi 5
Isoko North 5
Egbado /Yewa South 5
Ekiti 5
Irepo 5
Odigbo 5
Ezza North 5
Olorunsogo 5
Ahoada East 4
Igueben 4
Ekeremor 4
Asa 4
Odogbolu 4
Ifedore 4
Bodinga 4
Bogoro 4
Jos East 4
Oredo 4
Talata-Mafara 4
Abeokuta North 4
Patani 4
Aiyedaade 4
Vandeikya 4
Irele 4
Ilejemeje 4
Hawul 4
Obokun 3
Geidam 3
Kaura Namoda 3
Orumba North 3
Ganjuwa 3
Funtua 3
Tundun Wada 3
Gulani 3
Isokan 3
Boripe 3
Atiba 3
Bade 3
Nembe 3
Boluwaduro 3
Gwaram 3
Ngor-Okpala 3
Shomgom 3
Emuoha 3
Okobo 3
Ekiti West 3
Binji 3
Udenu 3
Igbo-eze North 3
Bende 3
Etche 3
Bunkure 3
Ekiti South-West 3
Eleme 3
Isiala Ngwa North 3
Obudu 3
Nafada 3
Koko/Besse 3
Shinkafi 2
Akko 2
Kalgo 2
Irewole 2
Bekwarra 2
Warji 2
Igbo-Etiti 2
Odukpani 2
Ede South 2
Oji-River 2
Idah 2
Nkanu West 2
Michika 2



Fires occurring

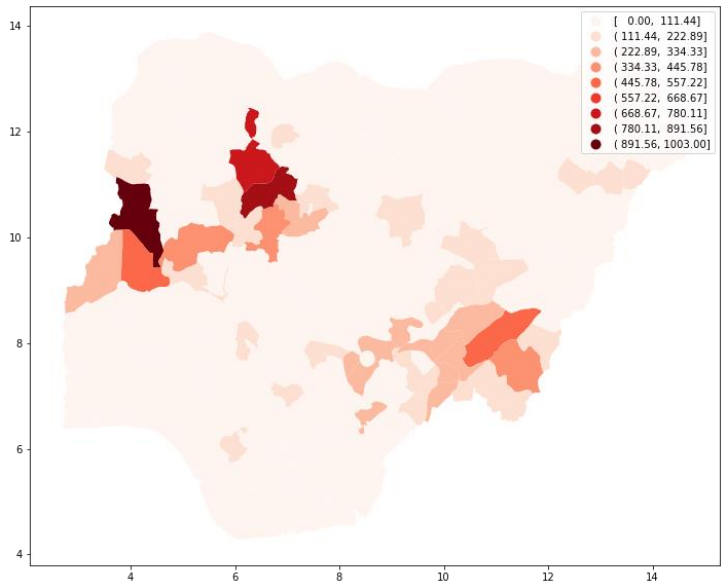
We can notice from the figure that at least one fire occurred in any local government area. The maximum of fire occurring is 1003 and it has occurred in Borgu local area.

1.10

Create a map displaying the number of fires for each local government area.

```
lga_index = nigeria_lga_gpdf.set_index('LGA')
lga_count  = link.groupby(by='LGA')[["LGA"]].count()
lga_count.rename(columns={'LGA':'nb_fires'},inplace=True)
lga_merge = lga_index.merge(lga_count,how='left', left_index=True, right
_index=True)
lga_merge = lga_merge.fillna(0.0)

lga_merge.plot(column='nb_fires', cmap='Reds', figsize=(16, 10), scheme=
'equal_interval', k=9, legend=True)
```

The map below shows the number of fires for each local government area.



In the map above, fires were classified into 9 group with approximately equal interval length. We can see that fires occurred mostly in the North-western part of the country followed by the South-Eastern part of the country. This confirm the previous density map of fires.

1.11

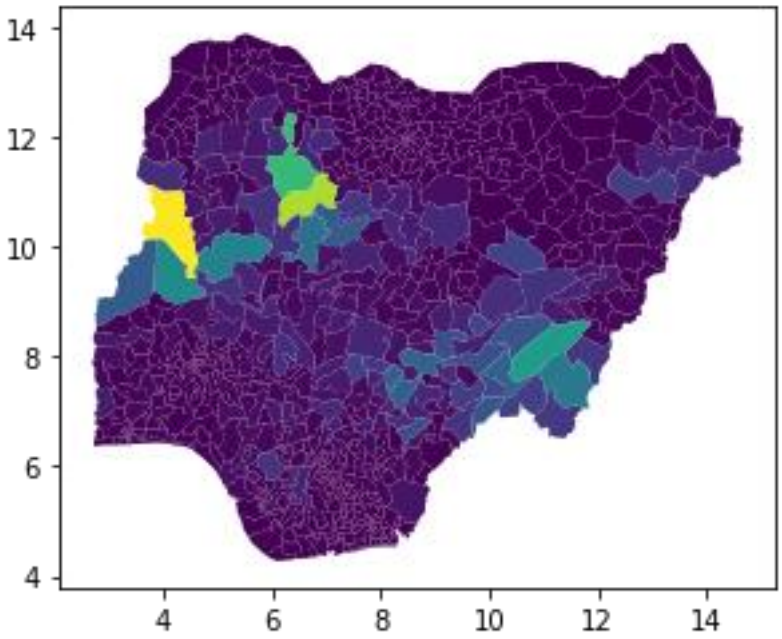
Investigate the spatial correlation between the numbers of fires in the different local government areas.

In order to understand better the notion of spatial autocorrelation, it is useful to begin by considering its absence. A key idea in this context is that of spatial randomness: a situation in which the location of an observation gives no information whatsoever about its value. In other words, a variable is spatially random if it is distributed following no discernible pattern over space. Spatial autocorrelation can thus be formally defined as the “absence of spatial randomness”. This definition renders spatial autocorrelation as a very encompassing and daunting concept. To better understand it, spatial autocorrelation is typically categorized along two main dimensions: sign and scale. Similar to the traditional, non-spatial case, spatial autocorrelation can adopt two main forms: positive and negative.

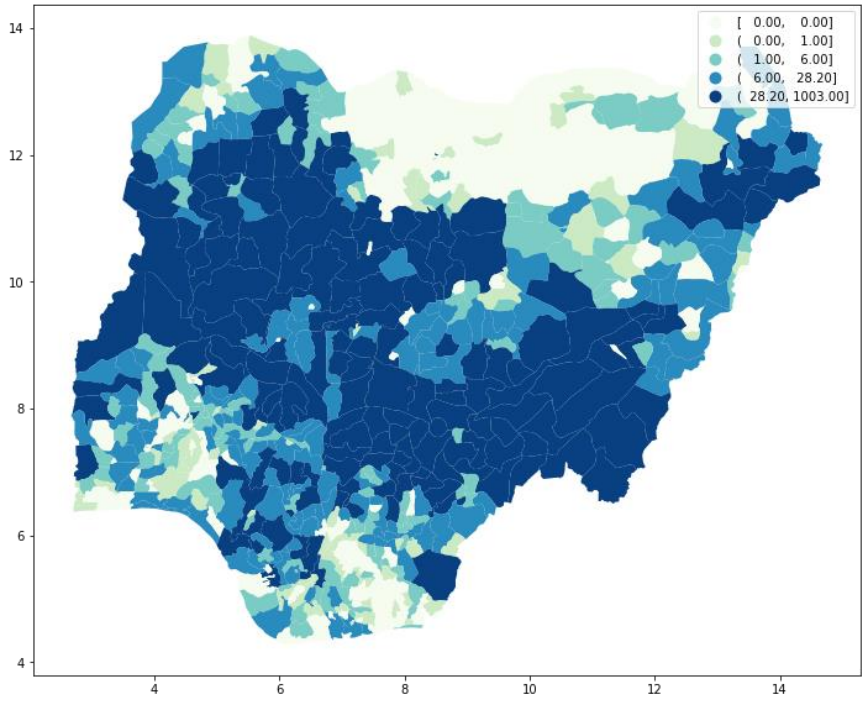
The former relates to a situation where similarity and geographical closeness go hand-in-hand. In other words, similar values are located near each other, while different values tend to be scattered and further away. It is important that the sign of these values is not relevant for the presence of spatial autocorrelation: it may be high values close to high values, or low values close to low values. The important bit in this context is the relationship between closeness and statistical similarity is positive (cited from https://geographicdata.science/book/notebooks/06_spatial_autocorrelation.html).

We can start by calculating **the spatial lag**.

```
import esda
from geopandas import GeoDataFrame
import libpysal as lps
from shapely.geometry import Point
%matplotlib inline

lga_merge.to_file('../Workspace/fires_by_lga.shp')
fires_by_lga = gpd.read_file('../Workspace/fires_by_lga.shp')

#median computation (grouped by lga)
median_fires_by_lga = fires_by_lga['nb_fires'].groupby([fires_by_lga['LGA']]).
mean()
```



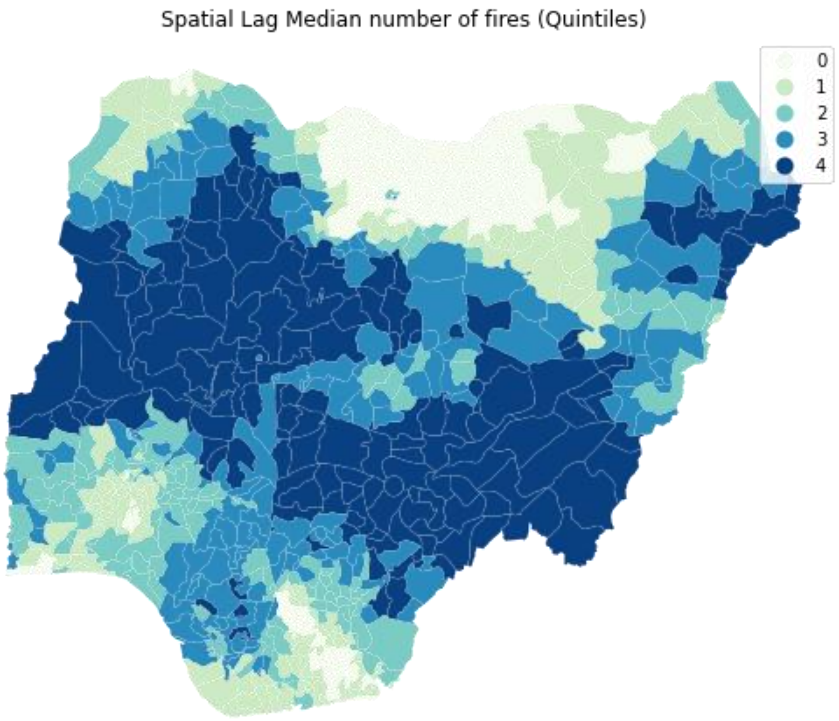

```
fig, ax = plt.subplots(figsize=(12,10), subplot_kw={'aspect':'equal'})
nigeria_median_fires_by_lga.plot(column='median_nb_fires', scheme='Quantiles',
 k=5, cmap='GnBu', legend=True, ax=ax)
```



Visual inspection of the map pattern for fires number allows us to search for spatial structure. If the spatial distribution of fires was random, then we should not see any clustering of similar values on the map. However, our visual system is drawn to the darker clusters in the center, and a concentration of the lighter hues (lower number of fire) in the north and south.

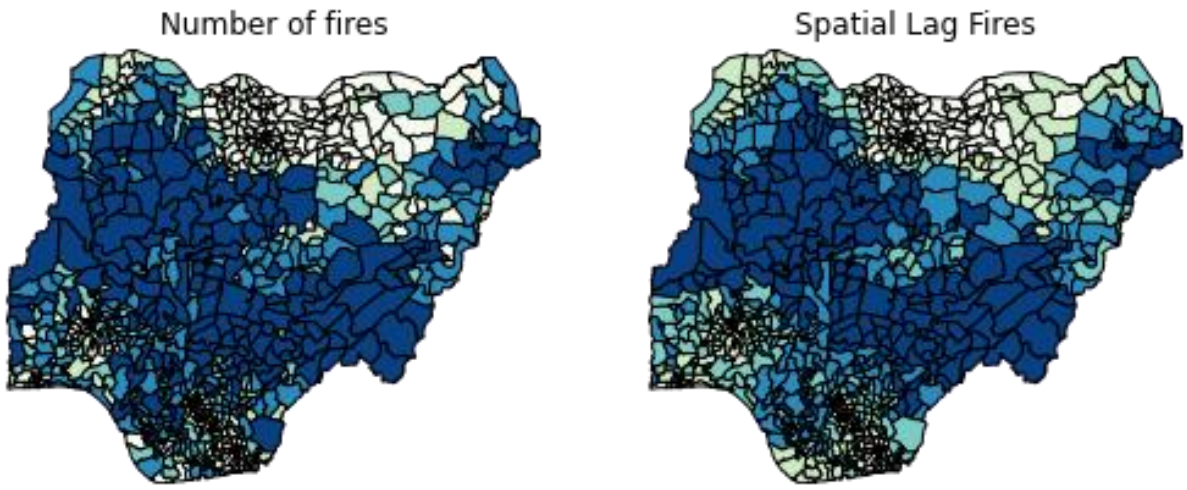
Now we are going to generate spatial similarity and attribute similarity. In spatial autocorrelation analysis, the spatial weights are used to formalize the notion of spatial similarity. here queen contiguity is used for that.

```
df = nigeria_median_fires_by_lga
wq = lps.weights.Queen.from_dataframe(df)
wq.transform = 'r'

y = df['median_nb_fires']
ylag = lps.weights.lag_spatial(wq, y)

#ylag

import mapclassify as mc
ylagq5 = mc.Quantiles(ylag, k=5)

f, ax = plt.subplots(1, figsize=(9, 9))
df.assign(cl=ylagq5.yb).plot(column='cl', categorical=True, \
        k=5, cmap='GnBu', linewidth=0.1, ax=ax, \
        edgecolor='white', legend=True)
ax.set_axis_off()
plt.title("Spatial Lag Median number of fires (Quintiles)")
plt.show()
```



Let's plot side by side the number of fires and the spatial lag of the number of fires.

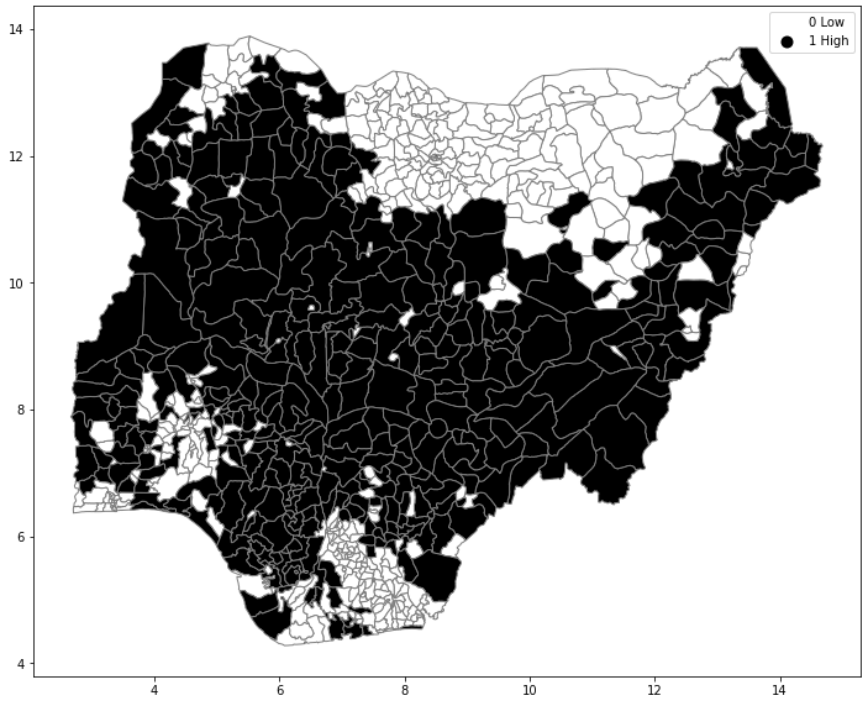
```
df['lag_median_nb_fires'] = ylag
f,ax = plt.subplots(1,2,figsize=(2.16*4,4))
df.plot(column='median_nb_fires', ax=ax[0], edgecolor='k',
        scheme="quantiles",  k=5, cmap='GnBu')
ax[0].axis(df.total_bounds[np.asarray([0,2,1,3])])
ax[0].set_title("Number of fires")
df.plot(column='lag_median_nb_fires', ax=ax[1], edgecolor='k',
        scheme='quantiles', cmap='GnBu', k=5)
ax[1].axis(df.total_bounds[np.asarray([0,2,1,3])])
ax[1].set_title("Spatial Lag Fires")
ax[0].axis('off')
ax[1].axis('off')
plt.show()
```



However, visually it is difficult to distinguish the two-above map. Thus, to complement the geo-visualization of these associations we can turn to formal statistical measures of spatial autocorrelation.

- Global Spatial Autocorrelation
  - Binary case (high and low autocorrelation)

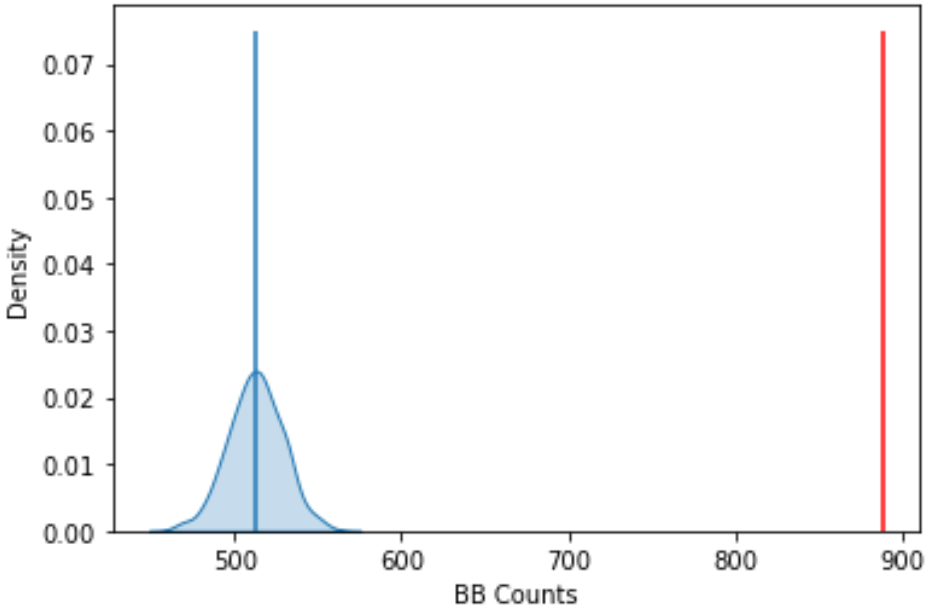
```
yb = y > y.median()
labels = ["0 Low", "1 High"]
yb = [labels[i] for i in 1*yb]
df['yb'] = yb

fig, ax = plt.subplots(figsize=(12,10), subplot_kw={'aspect':'equal'})
df.plot(column='yb', cmap='binary', edgecolor='grey', legend=True, ax=ax)
```



Now let's look at the joint counts for each neighbor pair of observations, and the joins are reflected in our binary spatial weights object wq. If we pair each region with its neighbors we

can get three different types of joins for each pairing: Low Low (white white) High High (black black) High Low (black white)

```
np.random.seed(12345)
import esda

yb = 1 * (y > y.median()) # convert back to binary
wq =  lps.weights.Queen.from_dataframe(df)
wq.transform = 'b'
jc = esda.join_counts.Join_Counts(yb, wq)

jc.sim_bb = jc.sim_bb.astype('float64')

sns.kdeplot(jc.sim_bb, shade=True)
plt.vlines(jc.bb, 0, 0.075, color='r')
plt.vlines(jc.mean_bb, 0,0.075)
plt.xlabel('BB Counts')

Text(0.5, 0, 'BB Counts')
```



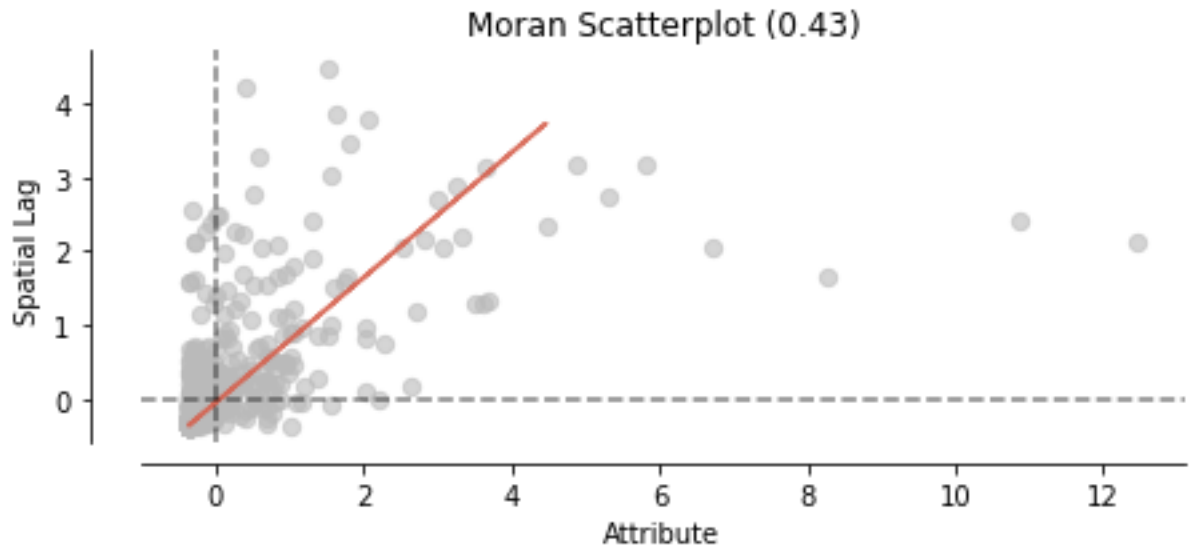
The density plot shows the distribution of the BB counts, with the blue vertical line indicating the mean BB count from the synthetic realizations and the red line the observed BB count. Clearly our observed value is extremely high. Since this is below conventional significance levels, we would reject the null of complete spatial randomness in favor of spatial autocorrelation.

- Moran's I

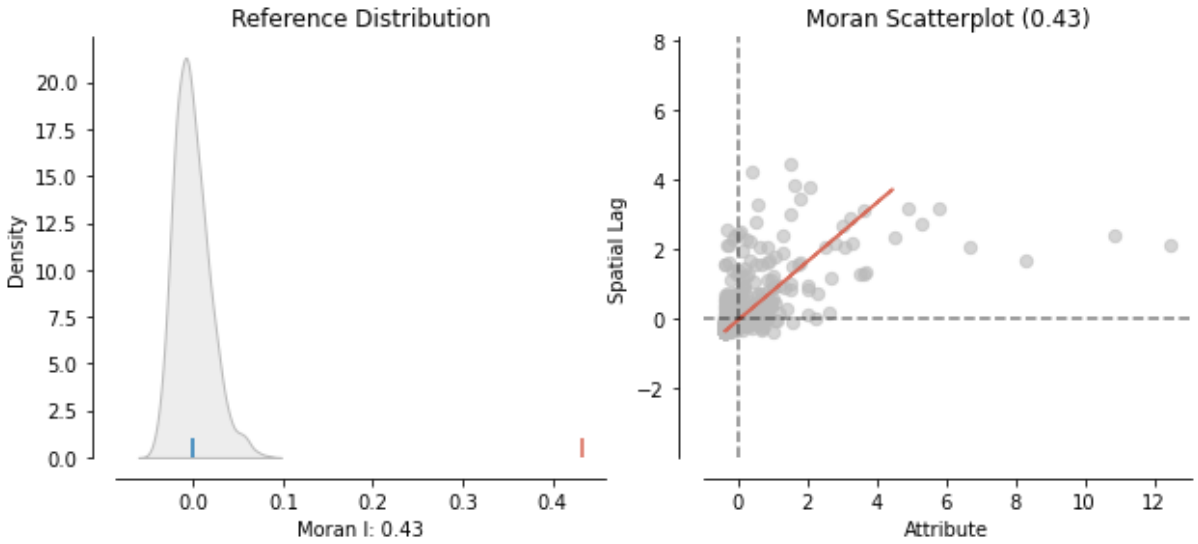
First, we transform our weights to be row-standardized, from the current binary state:

```
wq.transform = 'r'
y = df['median_nb_fires']

from esda.moran import Moran
moran = Moran(y, wq)
```

Now we proceed with Maran Scatterplot.



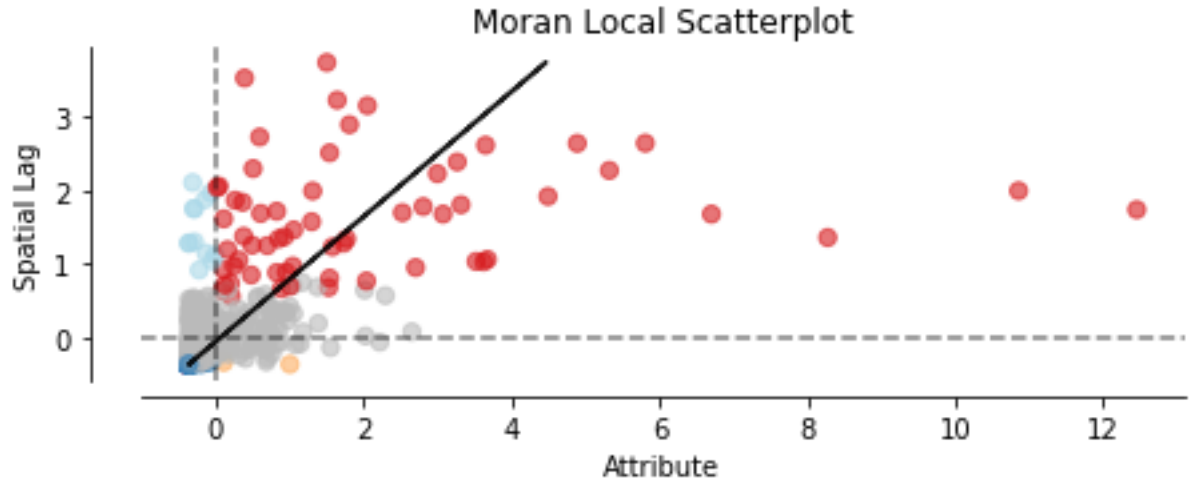
Let's have the reference distribution and Moran scatterplot side by side



- Local Autocorrelation

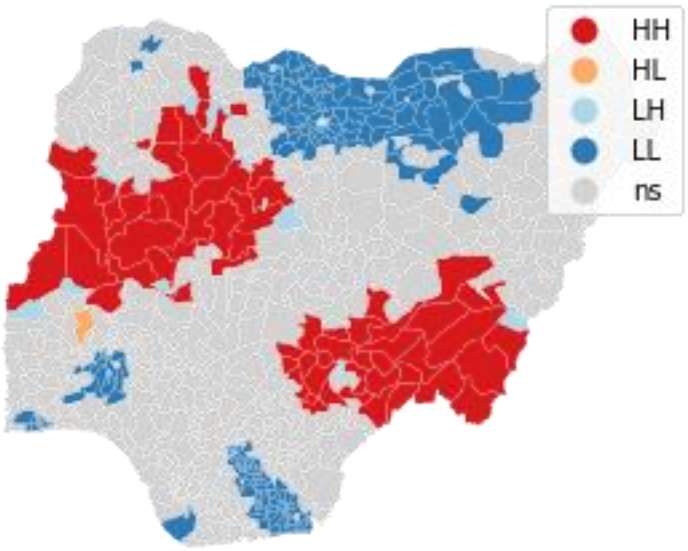
Here we are going to look at the local autocorrelation, which enables us to detect Hot Spots and Spatial Outliers.

```
fig, ax = moran_scatterplot(moran_loc, p=0.05)
ax.set_ylabel('Spatial Lag')
plt.show()
```



We can now distinguish between different types of autocorrelation These types of local spatial autocorrelation describe similarities or dissimilarities between a specific polygon with its neighboring polygons.

```
from splot.esda import lisa_cluster
lisa_cluster(moran_loc, nigeria_lga_gpdf, p=0.05)
plt.show()
```



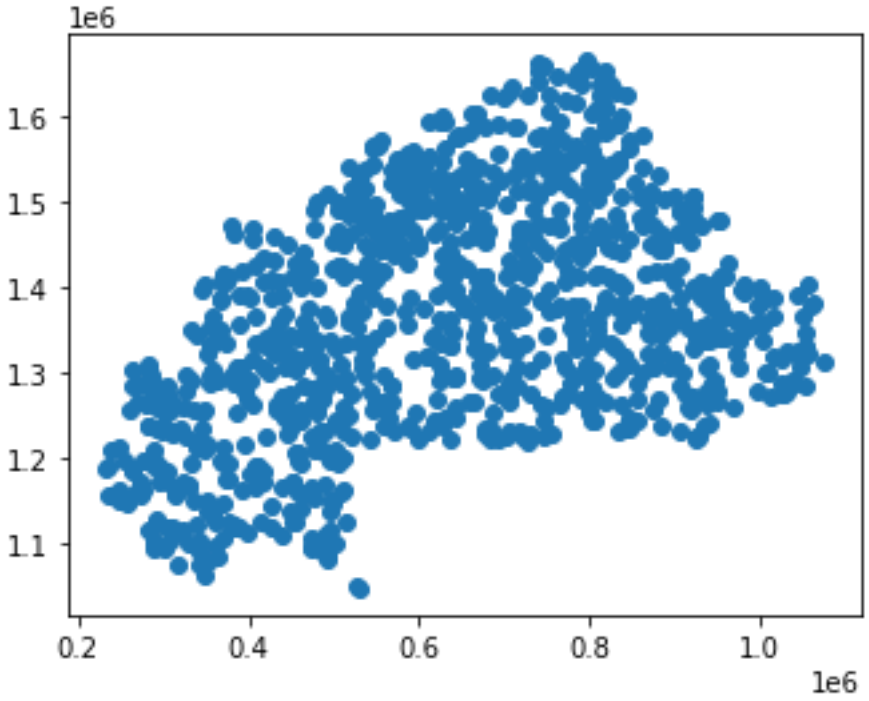
- Conclusion

The investigation of the spatial correlation between the numbers of fires in the different government areas results that there is exist an autocorrelation. This confirms the first low of geography: Everything is related to everything else, but near things are more related than distant things." Waldo R. Tobler (Tobler 1970)

2. Exercise 2

Import the elevation dataset for Burkina Faso (bfa_elevation.shp)

```
import geopandas as gpd
import matplotlib.pyplot as plt

bfa_elevation_gpdf = gpd.read_file('../project/bfa_elevation.shp')

bfa_elevation_gpdf.plot()
```



2.1

Choose a spatial interpolation (deterministic or stochastic spatial interpolation technique) method and create a continuous elevation raster in a spatial resolution (cell size) of 20 m. Explain why you decided for a certain spatial interpolation technique. Import the SRTM for Burkina Faso compare it to your results (bfa_srtm.tif).

- Spatial interpolation technique: **Kriging, stochastic technique**
- Reason:

Firstly, stochastic interpolation technique because of its advantage over the deterministic approaches is the consideration of the spatial variances of values by means of semi-variograms. For the predicted values at unknown locations, the weight of the sampled known values will be computed so that the estimation error of the variance is lowest as possible. Lastly, Kriging among various stochastic techniques because of its advantages:

*Directional influences can be accounted

*Exceeds the minimum and maximum point values.

*Derived a best linear unbiased estimator

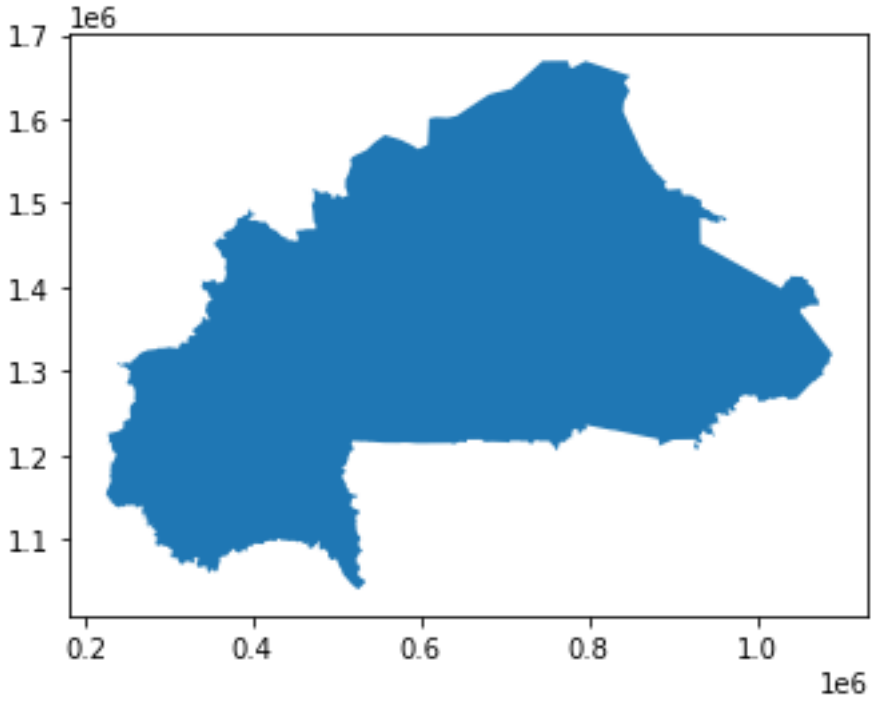
*It produces good data representation for dataset between 250 and 1000 observation.

*The produced result is raster data.

The number of observations of our dataset is 1000.

To start, let's import the region over which we wished to interpolation our dataset.

```
region_BF = gpd.read_file('/home/mrpinformcc/Documents/Limite BF.shp')
region_BF.plot()
```



Now we are going the to do the interpolation of our dataset using **Ordinary Kriging.**

```
import numpy as np
import pandas as pd
import glob
from pykrige.ok import OrdinaryKriging
import pykrige.kriging_tools as kt
from pykrige.kriging_tools import write_asc_grid
from matplotlib.colors import LinearSegmentedColormap
from matplotlib.patches import Path, PathPatch
```

In these following lines we are extracting the boundaries of interpolation region.

```
xmin = bfa_elevation_gpdf.geometry.x.min()
ymin = bfa_elevation_gpdf.geometry.y.min()
xmax = bfa_elevation_gpdf.geometry.x.max()
ymax = bfa_elevation_gpdf.geometry.y.max()
```

The we create range of values for x and y of the two-dimension interpolation area.

```
grid_lon = np.linspace(xmin, xmax, 100)
grid_lat = np.linspace(ymin, ymax, 100)
```

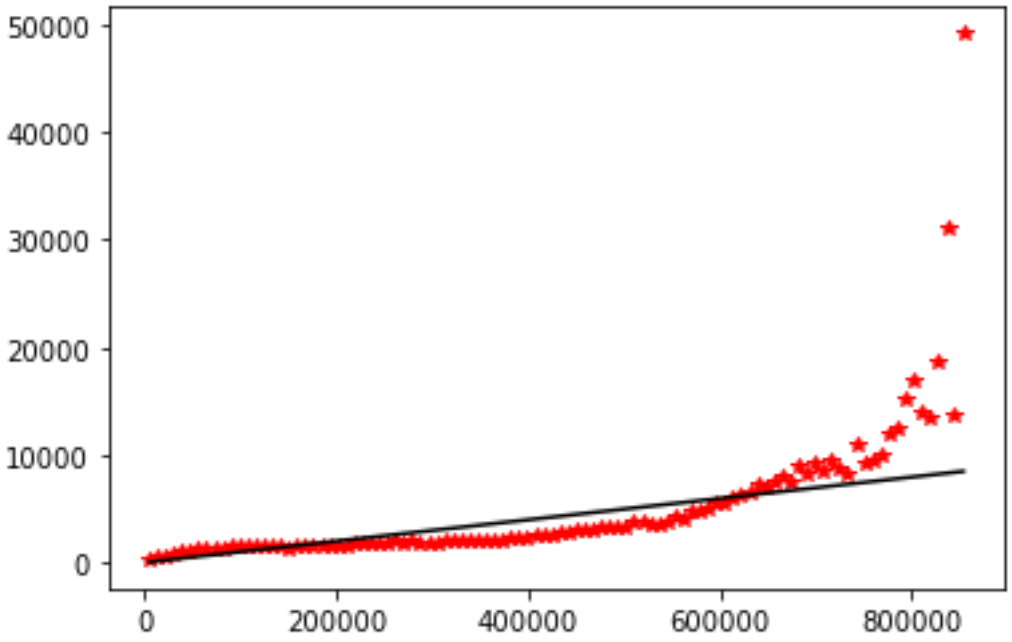
Here we add x and y columns to our datasets.

```
bfa_elevation_gpdf['x'] = bfa_elevation_gpdf['geometry'].x
bfa_elevation_gpdf['y'] = bfa_elevation_gpdf['geometry'].y
```

In the call of ordinary kriging method, we used linear for variogram_model because of the continuous distribution of our observation data.

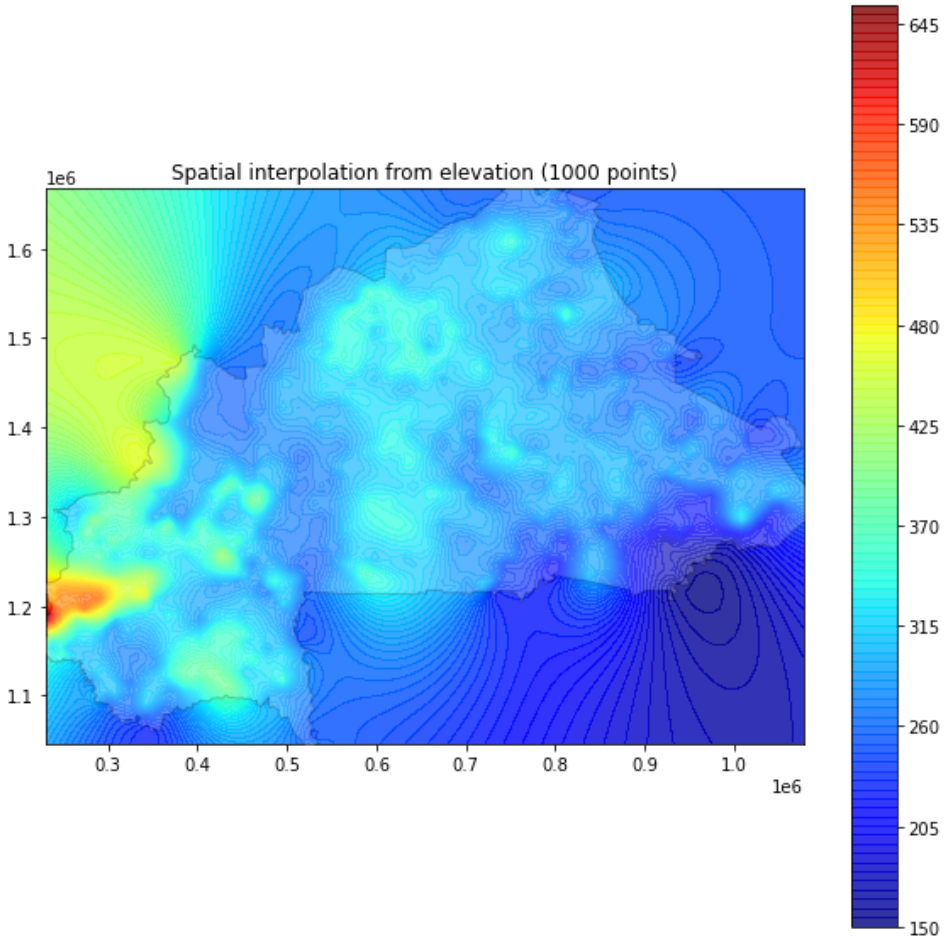
```
OK = OrdinaryKriging(bfa_elevation_gpdf.x, bfa_elevation_gpdf.y, bfa_elevation
_gpdf.SRTM30mBur, variogram_model='linear',verbose=True, enable_plotting=True,
 nlags=100)
z, ss = OK.execute('grid', grid_lon, grid_lat)
```

Using 'linear' Variogram Model
Slope: 0.009926049250352011
Nugget: 0.004315681047255765



The following map shows the result of the interpolation of elevation using Kriging.

```
xintrp, yintrp = np.meshgrid(grid_lon, grid_lat)

fig, ax = plt.subplots(figsize=(10,10))

contour = plt.contourf(xintrp, yintrp, z,len(z),cmap=plt.cm.jet,alpha = 0.8)
plt.colorbar(contour)

region_BF.plot(ax=ax, color='white', alpha = 0.2, edgecolor='black', zorder =
5)
npts = len(bfa_elevation_gpdf.x)


plt.xlim(xmin,xmax)
plt.ylim(ymin,ymax)

plt.xticks()
plt.yticks()


plt.title('Spatial interpolation from elevation (%d points)' % npts)
plt.show()
```



- **Creation of a continuous elevation raster in a spatial resolution of 20 mm**

```
rRes = 20
#definition of the raster transformation array
from rasterio.transform import Affine
transform = Affine.translation(xintrp[0][0] -rRes/2, yintrp[0][0] -rRes/2)*Aff
ine.scale(rRes,rRes)
transform
```

```
Affine(20.0, 0.0, 230966.40281275153,
       0.0, 20.0, 1046709.9229832509)
```

```
#get crs as wkt
from rasterio.crs import CRS
rasterCrs = CRS.from_epsg(32630)
rasterCrs.data
```

```
{'init': 'epsg:32630'}
```

```
#definition, register and close of interpolated raster

import rasterio
interpRaster = rasterio.open('../Workspace/interpRaster.tif',
                             'w',
                             driver='GTiff',
                             height=1000,
                             width=1500,
                             count=1,
                             dtype=z.dtype,
                             crs=rasterCrs,
                             transform=transform,
                            )

interpRaster.write(z, 1)
interpRaster.close()
```

- **Importation the SRTM for Burkina Faso**

```
import rasterio
bfa_srtm = rasterio.open('../project/bfa_srtm-002.tif')
```

Here we are creating a function which could be used to display information about the raster data.

```
def displayInfo(raster):
    # what is the name of this image
    img_name = raster.name
    print('Image filename: {n}'.format(n=img_name))
    # How many bands does this image have?
    num_bands = raster.count
    print('Number of bands in image: {n}'.format(n=num_bands))
    # How many rows and columns?
    rows, cols = raster.shape
    print('Image size is: {r} rows x {c} columns'.format(r=rows, c=cols))
    # Does the raster have a description or metadata?
    desc = raster.descriptions
```

```
    metadata = raster.meta
    print('Raster description: {desc}'.format(desc=desc))
    # What driver was used to open the raster?
    driver = raster.driver
    print('Raster driver: {d}'.format(d=driver))
    # What is the raster's projection?
    proj = raster.crs
    print('Image projection:')
    print(proj, '\n')
    # What is the raster's "geo-transform"
    gt = raster.transform
    print('Image geo-transform:\n{gt}\n'.format(gt=gt))
    print('\n')

    print('All raster metadata:')
    print(metadata
```

Now let's call our function on our imported raster data.

```
displayInfo(bfa_srtm)
```

Image filename: ../project/bfa_srtm-002.tif
Number of bands in image: 1
Image size is: 20452 rows x 28526 columns
Raster description: (None,)
Raster driver: GTiff
Image projection:
EPSG:4326

Image geo-transform:
| 0.00, 0.00,-5.52|
| 0.00,-0.00, 15.08|
| 0.00, 0.00, 1.00|

All raster metadata:
{'driver': 'GTiff', 'dtype': 'float32', 'nodata': None, 'width': 28526, 'height': 20452, 'count': 1, 'crs': CRS.from_epsg(4326), 'transform': Affine(0.00027777777778167283, 0.0, -5.51875,
0.0, -0.00027777777777234506, 15.082361111)}

- **Comparison with the interpolated raster**

Features	STRM DEM raster	Interpolated raster
Number of bands	1	1
Image size	20452 rows x 28526 columns	1000 rows x 1500 columns
Raster driver	GTiff	GTiff
Image projection	EPSG:4326	EPSG: 322630
Resolution	0.00027	20

2.2

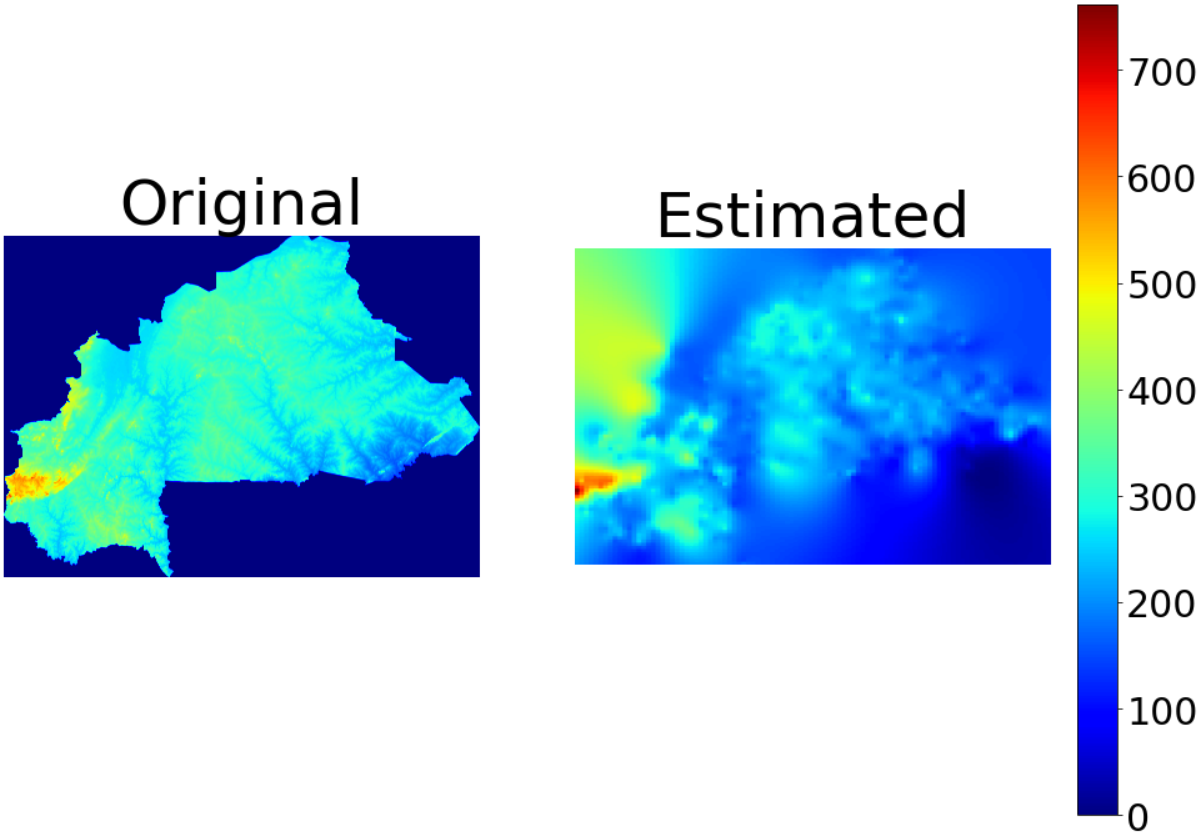
Show maps for both raster and compare your created continuous elevation raster with the SRTM Digital Elevation Model for Burkina Faso. What do you observe? Where are the differences?

Firs we import both raster data

```
original_data = rasterio.open('../project/bfa_srtm-002.tif').read(1)[1:-1,1:-1]

estimated_data = rasterio.open('../Workspace/interpRaster.tif').read(1)[1:-1,1:-1]
```

Now, let's plot them.

```
estimated_data = estimated_data[::-1,:]

fig = plt.figure(figsize=(15,15))
plt.subplot(121)
im = plt.imshow(original_data,cmap=plt.cm.jet)
plt.axis('off')
plt.title('Original',fontsize=50)
cb_ax = fig.add_axes([0.92, 0.2, 0.03, 0.6])
cb = fig.colorbar(im, cax=cb_ax)
cb.ax.tick_params(labelsize=30)
plt.subplot(122)
im = plt.imshow(estimated_data,cmap=plt.cm.jet)
plt.axis('off')
plt.title('Estimated',fontsize=50)
plt.show()
```



The maps above show both rasters whiten a scale of 0-750 m of elevation.

We do observe the rasters data do not have the same range of pixel values. On the other hand, we can say that the interpolation is approximately accurate. The highest values can be observed at the north-southern part of Burkina Faso and lowest values around the northeastern part of the country.