

Statistiques Exploratoire Spatiales (avec Python)

Réalisé par le Groupe 3

ENSAE 2024 : ISE1 CL

TP1 : Importation et Visualisation de données

DAILY AVERAGE TEMPERATURE PREDICTION

```
In [11]: # Définition de notre work directory
import os
directory='C:/Users/DELL/Documents/ISEP3_2025/Stats_spatiale/Statistique-Explora
os.chdir(directory)
```

LES PRÉLIMINAIRES...

Les fichiers utilisés

Les shape file...petit rappel sur la composition ; parler des niveaux 0 1 2 et 3

le Rateur Mali avec rappel dessus

Les bibliothèques utilisées

- La bibliothèque **rasterio** : elle...
- La bibliothèque **geopandas** : elle...
- La bibliothèque **matplotlib** : elle...
- La bibliothèque **folium** : elle...

```
In [12]: #importation des libraries

import rasterio
import geopandas as gpd
```

```
import matplotlib.pyplot as plt
import folium
```

VISUALISATION DES DONNÉES

Dans cette partie, le travail sera d'abord fait sur le fichier contenant les divisions administratives de niveau 2. On répliquera alors le processus sur les autres

Importation des fichiers

DÉCOUPAGE DE NIVEAU 2

Importation des fichiers

```
In [14]: # Importation du .tiff
tiff_file = '201501_Global_Travel_Time_to_Cities_MLI.tiff'

# Importatation du shapefile contenant les divisions administratives
shapefile = 'mli_admbnda_adm2_1m_gov_20211220.shp'
```

Ouverture du .tiff avec rasterio et visualisatio des données

```
In [15]: # Ouverture du .tiff avec rasterio
with rasterio.open(tiff_file) as src:
    # Lire la première couche du fichier .tiff
    tiff_data = src.read(1)

    # Affichage des métadonnées du .tiff
    print(src.meta)

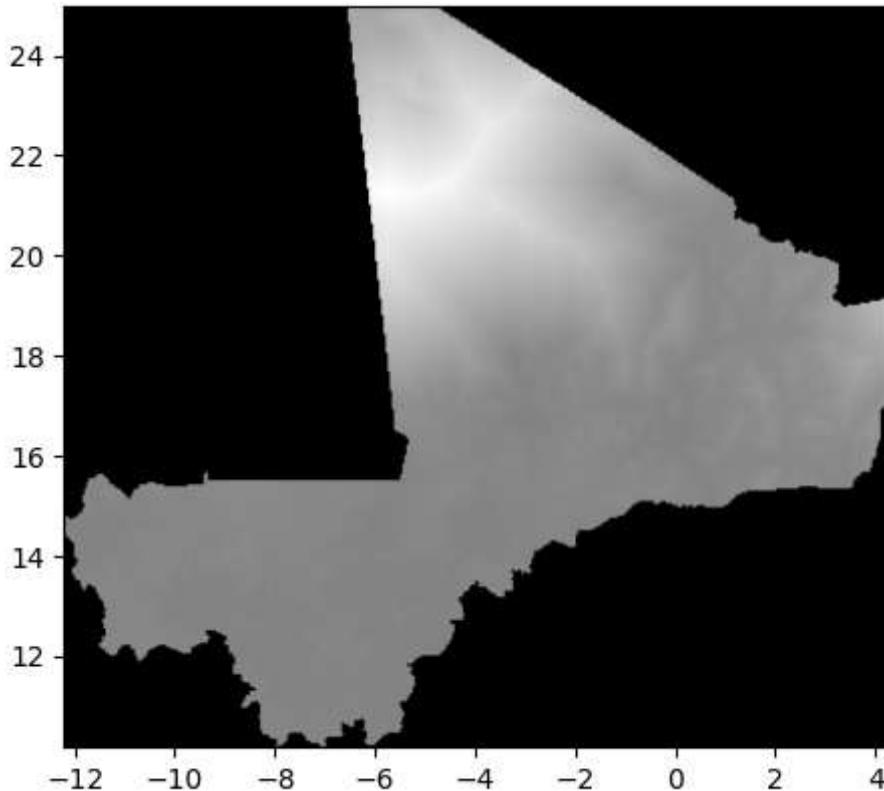
    # Lecture des informations de transformation pour projeter correctement les do
    transform = src.transform

{'driver': 'GTiff', 'dtype': 'int32', 'nodata': -9999.0, 'width': 1975, 'height': 1780, 'count': 1, 'crs': CRS.from_epsg(4326), 'transform': Affine(0.008333333333333333, 0.0, -12.208333333333332, 0.0, -0.008333333333333333, 25.0)}
```

Afichage de l'image rasteur

```
In [16]: # Affichage l'image .tiff  
plt.imshow(tiff_data, cmap='gray', extent=(transform[2], transform[2] + transform[5] + transform[4] * tiff_da
```

```
Out[16]: <matplotlib.image.AxesImage at 0x1c913f79e50>
```

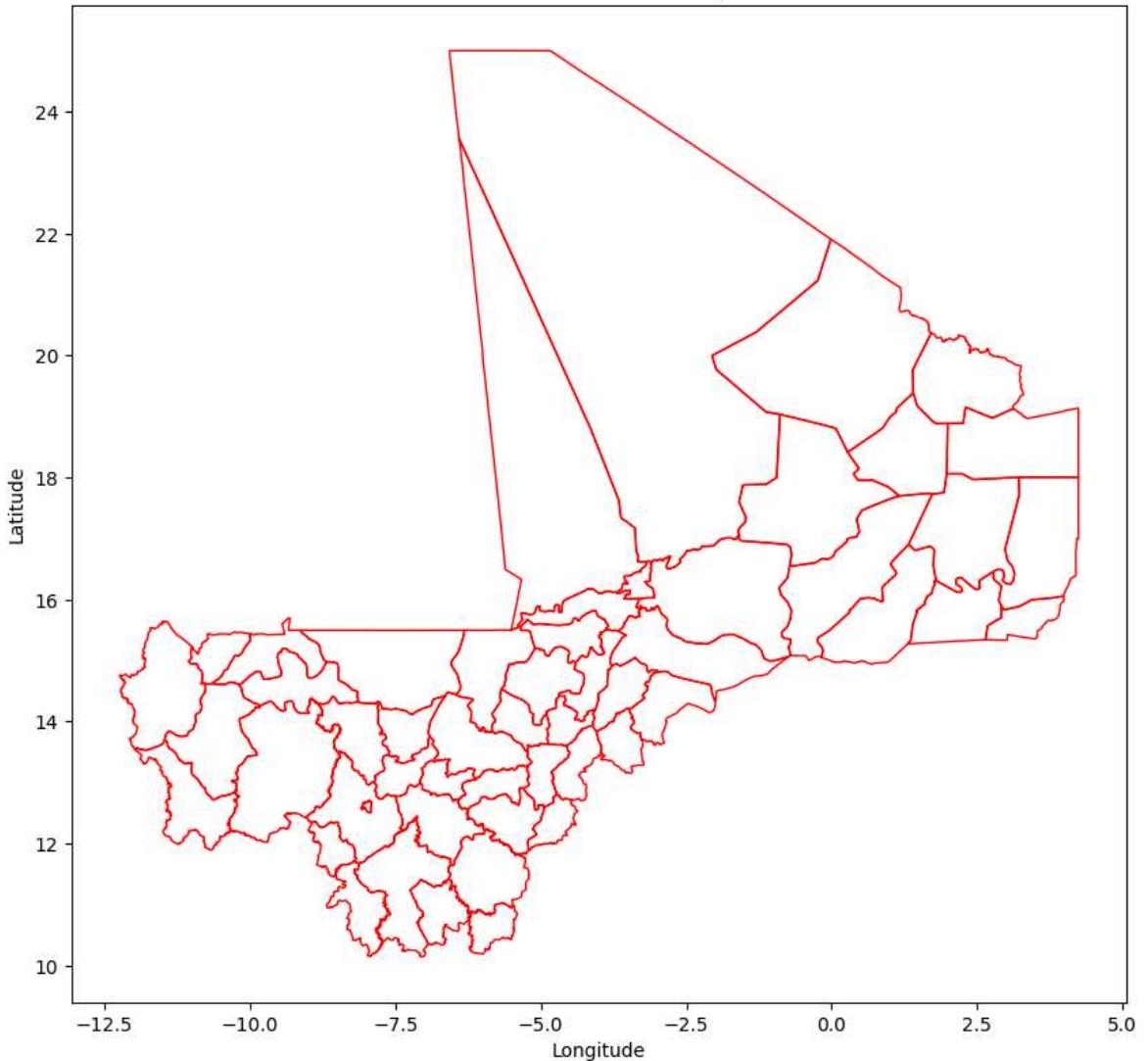


```
In [17]: # Chargement et visualisation du shapefile avec geopandas
```

```
In [18]: # Charger le shapefile avec geopandas  
gdf = gpd.read_file(shapefile)
```

```
In [19]: # Afficher la carte avec les données raster et les limites administratives  
fig, ax = plt.subplots(figsize=(10, 10))  
  
# Afficher les divisions administratives  
gdf.boundary.plot(ax=ax, edgecolor='red', linewidth=1)  
  
# Ajouter une légende et des axes  
plt.title('Divisions administratives uniquement')  
plt.xlabel('Longitude')  
plt.ylabel('Latitude')  
  
# Afficher la carte  
plt.show()
```

Divisions administratives uniquement



Visualisation des deux en même temps

Nous avions visualisé les limites (boundaries) et le rasteur séparément. Maintenant, mettons les ensemble.

Rasteur et limites sur un même graphe

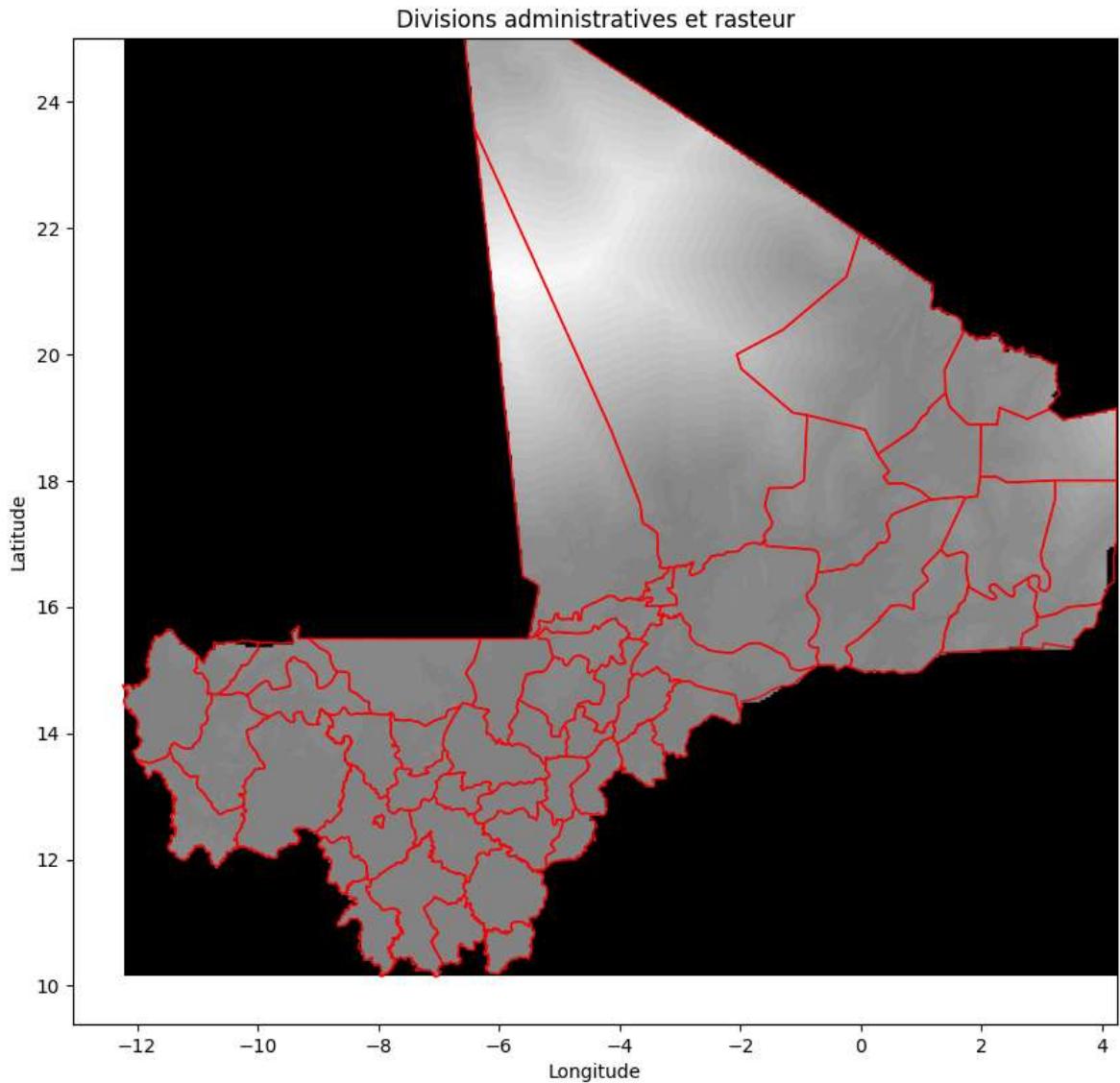
```
In [20]: # Mix des deux
# Afficher la carte avec les données raster et les limites administratives
fig, ax = plt.subplots(figsize=(10, 10))

# Afficher l'image .tiff
plt.imshow(tiff_data, cmap='gray', extent=(transform[2], transform[2] + transform[5] + transform[4] * tiff_da
# Afficher les divisions administratives
gdf.boundary.plot(ax=ax, edgecolor='red', linewidth=1)

# Ajouter une légende et des axes
plt.title('Divisions administratives et rasteur')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
```

```
# Afficher La carte
```

```
plt.show()
```



Ajout des labels

Les labels se trouvent dans le .dbf qui est chargé dans notre gdf. Regardons cette base de données

Visualisation de la base de données

In [32]:

```
# Visualisons un peu la base
```

```
# On s'intéresse au fichier dbf qui contient les données
```

```
gdf.info()
```

```
<class 'geopandas.geodataframe.GeoDataFrame'>
RangeIndex: 53 entries, 0 to 52
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Shape_Leng  53 non-null    float64
 1   Shape_Area  53 non-null    float64
 2   ADM2_FR     53 non-null    object  
 3   ADM2_PCODE  53 non-null    object  
 4   ADM2_REF    10 non-null    object  
 5   ADM2ALT1FR  0 non-null    object  
 6   ADM2ALT2FR  0 non-null    object  
 7   ADM1_FR     53 non-null    object  
 8   ADM1_PCODE  53 non-null    object  
 9   ADM0_FR     53 non-null    object  
 10  ADM0_PCODE  53 non-null    object  
 11  date        53 non-null    datetime64[ms]
 12  validOn    53 non-null    datetime64[ms]
 13  validTo    53 non-null    object  
 14  geometry    53 non-null    geometry
dtypes: datetime64[ms](2), float64(2), geometry(1), object(10)
memory usage: 6.3+ KB
```

In [30]: `gdf.head(5)`

Out[30]:

	Shape_Leng	Shape_Area	ADM2_FR	ADM2_PCODE	ADM2_REF	ADM2ALT1FR	ADM
--	------------	------------	---------	------------	----------	------------	-----

0	6.708207	1.654741	Bafoulabe		ML0101	None	None
---	----------	----------	-----------	--	--------	------	------

1	6.617321	1.075836	Diéma		ML0102	Diema	None
---	----------	----------	-------	--	--------	-------	------

2	7.243382	1.899436	Kayes		ML0103	None	None
---	----------	----------	-------	--	--------	------	------

3	8.276270	1.245417	Kéniéba		ML0104	Kenieba	None
---	----------	----------	---------	--	--------	---------	------

4	10.839833	2.981976	Kita		ML0105	None	None
---	-----------	----------	------	--	--------	------	------

In [31]: `# On va voir la colonne qui contient les données`

```
gdf.columns
```

```
Out[31]: Index(['Shape_Leng', 'Shape_Area', 'ADM2_FR', 'ADM2_PCODE', 'ADM2_REF',
   'ADM2ALT1FR', 'ADM2ALT2FR', 'ADM1_FR', 'ADM1_PCODE', 'ADM0_FR',
   'ADM0_PCODE', 'date', 'validOn', 'validTo', 'geometry'],
  dtype='object')
```

-

Là, on choisit la colonne contenant les labels et on les affiche

Affichage des labels

```
In [33]: # Afficher la carte avec les données raster et les limites administratives
fig, ax = plt.subplots(figsize=(10, 10))

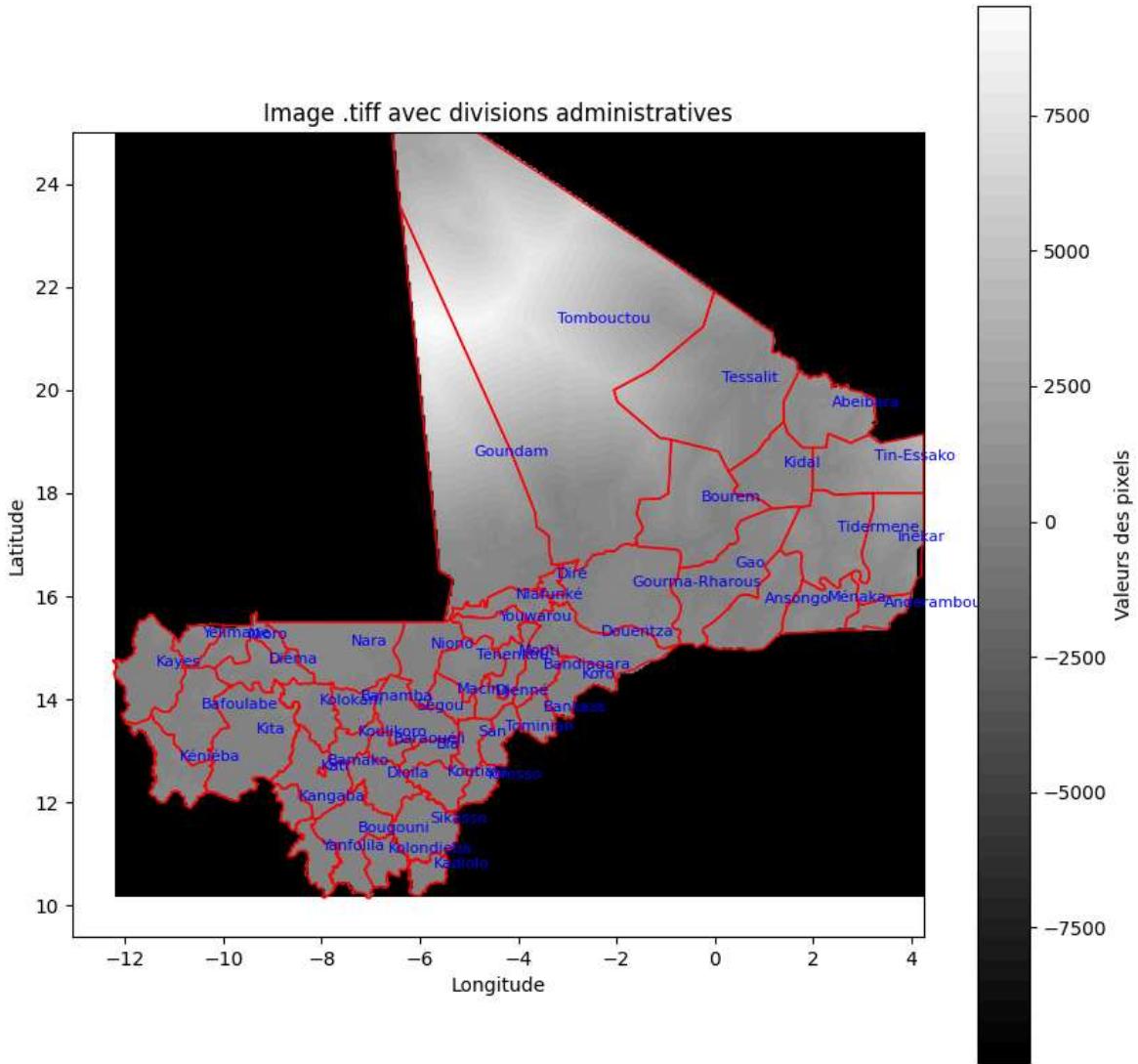
# Afficher l'image .tiff
plt.imshow(tiff_data, cmap='gray', extent=(transform[2], transform[2] + transform[5] + transform[4] * tiff_da

# Afficher les divisions administratives
gdf.boundary.plot(ax=ax, edgecolor='red', linewidth=1)

# Ajouter les labels
for idx, row in gdf.iterrows():
    # Utiliser le centreïde de chaque division pour positionner le label
    centroid = row.geometry.centroid
    plt.annotate(text=row['ADM2_FR'], xy=(centroid.x, centroid.y),
                 xytext=(3, 3), textcoords="offset points",
                 fontsize=8, color='blue')

# Ajouter une légende et des axes
plt.colorbar(label='Valeurs des pixels')
plt.title('Image .tiff avec divisions administratives')
plt.xlabel('Longitude')
plt.ylabel('Latitude')

# Afficher la carte
plt.show()
```



Affichage de la carte complète avec les autres niveaux

```
In [38]: # Chemin vers ton fichier .tiff
tiff_file = '201501_Global_Travel_Time_to_Cities_MLI.tiff'

# Chemin vers ton fichier shapefile contenant les divisions administratives
shapefile = 'mli_admbnda_adm0_1m_gov_20211220.shp'

# Ouvrir le fichier .tiff avec rasterio
with rasterio.open(tiff_file) as src:
    # Lire la première couche du fichier .tiff
    tiff_data = src.read(1)

    # Lire les informations de transformation pour projeter correctement les données
    transform = src.transform

# Charger le shapefile avec geopandas
gdf = gpd.read_file(shapefile)
```

```
In [39]: gdf.head(5)
```

Out[39]:

	Shape_Leng	Shape_Area	ADM0_FR	ADM0_PCODE	ADM0_REF	ADM0ALT1FR	ADM
--	------------	------------	---------	------------	----------	------------	-----

0	71.768739	106.823823	Mali (le)	ML	None	None	
---	-----------	------------	-----------	----	------	------	--

In [40]:

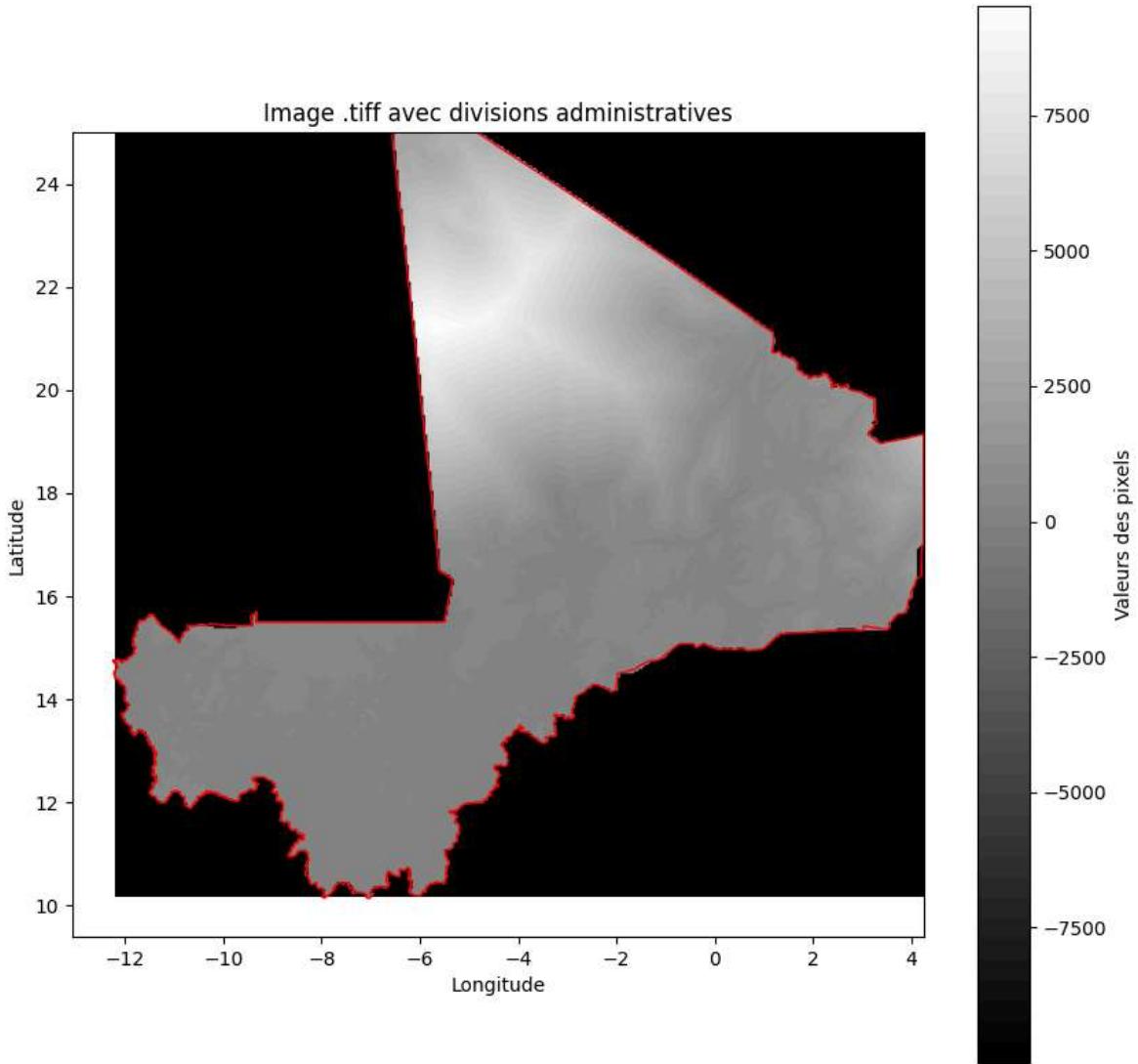
```
# Afficher la carte avec les données raster et les limites administratives
fig, ax = plt.subplots(figsize=(10, 10))

# Afficher l'image .tiff
plt.imshow(tiff_data, cmap='gray', extent=(transform[2], transform[2] + transform[3],
                                             transform[5] + transform[4] * tiff_da
```

```
# Afficher les divisions administratives
gdf.boundary.plot(ax=ax, edgecolor='red', linewidth=1)
```

```
# Ajouter une Légende et des axes
plt.colorbar(label='Valeurs des pixels')
plt.title('Image .tiff avec divisions administratives')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
```

```
# Afficher la carte
plt.show()
```



Voyons avec les autres (modifications en instantané)

```
In [43]: # fichier .tiff
tiff_file = '201501_Global_Travel_Time_to_Cities_MLI.tiff'

# fichier shapefile contenant les divisions administratives
shapefile = 'mli_admbnda_adm1_1m_gov_20211220.shp'

# Ouvrir le fichier .tiff avec rasterio
with rasterio.open(tiff_file) as src:
    # Lire la première couche du fichier .tiff
    tiff_data = src.read(1)

    # Lire les informations de transformation pour projeter correctement les don
    transform = src.transform

# Charger le shapefile avec geopandas
gdf = gpd.read_file(shapefile)

In [44]: gdf.head(5)
```

Out[44]:

	Shape_Leng	Shape_Area	ADM1_FR	ADM1_PCODE	ADM1_REF	ADM1ALT1FR	ADM
0	20.449044	10.250256	Kayes	ML01	None	None	
1	21.764713	7.563718	Koulikoro	ML02	None	None	
2	20.958431	5.965027	Sikasso	ML03	None	None	
3	14.430528	5.158060	Ségou	ML04	Segou	None	
4	16.266328	6.649725	Mopti	ML05	None	None	

In [46]:

```
# Afficher la carte avec les données raster et les limites administratives
fig, ax = plt.subplots(figsize=(10, 10))

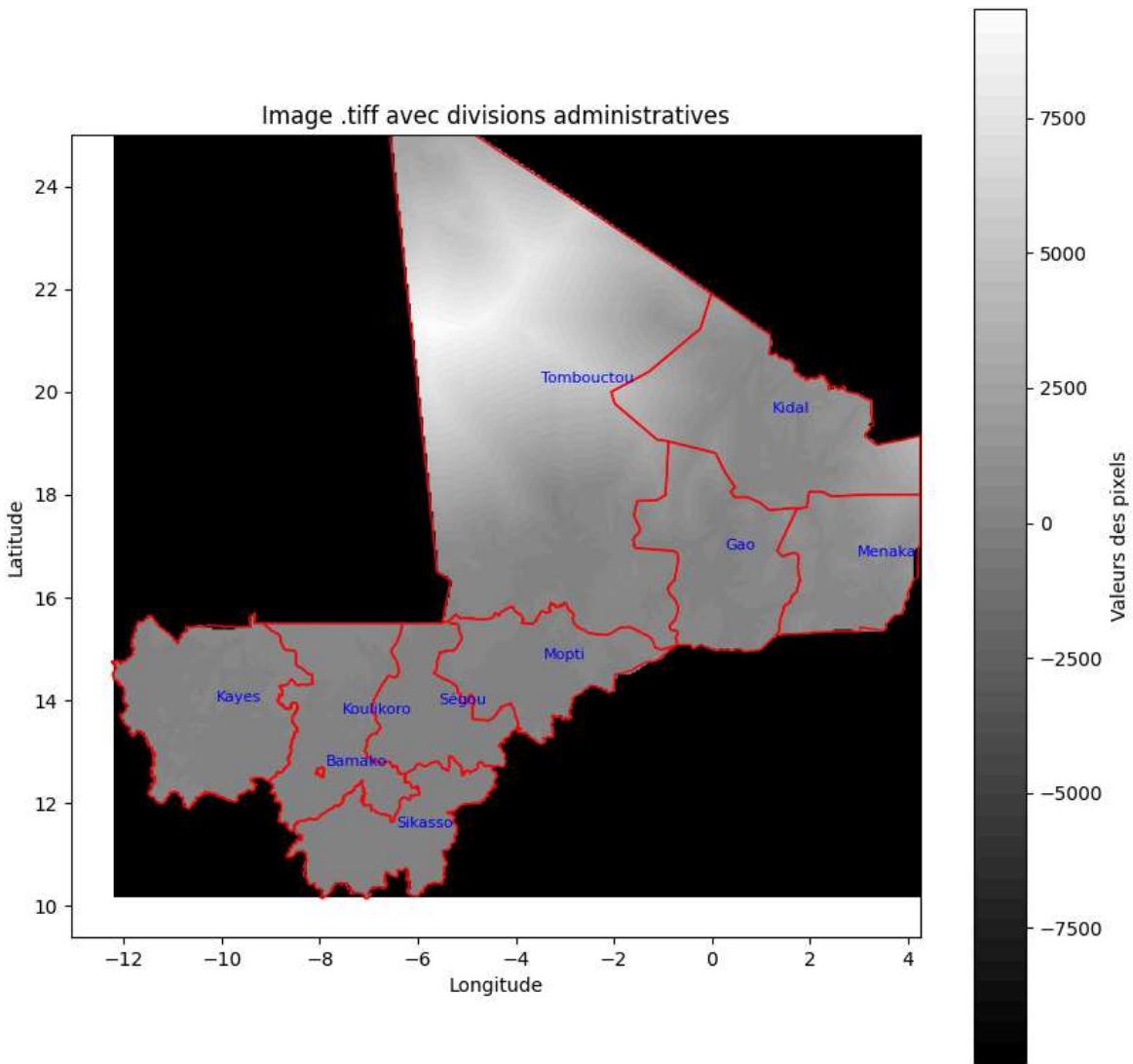
# Afficher l'image .tiff
plt.imshow(tiff_data, cmap='gray', extent=(transform[2], transform[2] + transform[5], transform[5] + transform[4] * tiff_da

# Afficher les divisions administratives
gdf.boundary.plot(ax=ax, edgecolor='red', linewidth=1)

# Ajouter les labels
for idx, row in gdf.iterrows():
    # Utiliser le centroïde de chaque division pour positionner le label
    centroid = row.geometry.centroid
    plt.annotate(text=row['ADM1_FR'], xy=(centroid.x, centroid.y),
                xytext=(3, 3), textcoords="offset points",
                fontsize=8, color='blue')

# Ajouter une légende et des axes
plt.colorbar(label='Valeurs des pixels')
plt.title('Image .tiff avec divisions administratives')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
```

```
# Afficher la carte  
plt.show()
```



Visualisation sur une carte

Nous utilisons ici la bibliothèque folio

Créons une simple carte

```
In [49]: # Create a map centered on a specific location  
m = folium.Map(location=[12.0, -8.0], zoom_start=6)  
  
# Nous ajoutons un marker  
folium.Marker([12.0, -8.0], popup='Point').add_to(m)  
  
m
```

Out[49]: Make this Notebook Trusted to load map: File -> Trust Notebook

Ajout du shapefile sur la carte avec les labels en popup

```
In [50]: # fichier shapefile contenant les divisions administratives
shapefile = 'mli_admbnda_adm2_1m_gov_20211220.shp'
gdf = gpd.read_file(shapefile)

# La carte Folium centrée sur une localisation générale
m = folium.Map(location=[12.0, -8.0], zoom_start=6)

# L'ajout des labels
for _, r in gdf.iterrows():
    sim_geo = gpd.GeoSeries(r["geometry"]).simplify(tolerance=0.001)
    geo_j = sim_geo.to_json()
    geo_j = folium.GeoJson(data=geo_j, style_function=lambda x: {"fillColor": "o
    folium.Popup(r["ADM2_FR"]).add_to(geo_j)
    geo_j.add_to(m)
```

In [51]: m

Out[51]: Make this Notebook Trusted to load map: File -> Trust Notebook

**MERCI DE VOTRE ATTENTION
DES QUESTIONS ?**

In []: