Leksioni 5

Endri Raco

05 May, 2024

Section 1

Të dhënat Raster (vazhdim)

Analiza e Pikave dhe Vlerësimi i Dendësisë së Kernelit (KDE)

```
import urllib.request

# Shkarkoni të dhënat e anijeve të mbytura historike
url_shipwrecks = 'https://github.com/endri81/instatgis/blob/master/data/gis4/darmc_historical_shipwrecks_500bce
file_name_shipwrecks = 'data/darmc_historical_shipwrecks_500bce_1500ce.geojson'
urllib.request.urlretrieve(url_shipwrecks, file_name_shipwrecks)

# Shkarkoni gjithashtu kufijtë e vendeve
url_boundaries = 'https://github.com/endri81/instatgis/blob/master/data/gis4/natural_earth_world_boundaries_50m
file_name_boundaries = 'data/natural_earth_world_boundaries_50m_2018.geojson'
urllib.request.urlretrieve(url_boundaries, file_name_boundaries)
```

Ngarkimi dhe Projekti i Dataset-eve

```
import geopandas as gpd

# Ngarkoni dataset-in dhe projektoni në 3035 (Lambert, i përshtatshëm për Evropën)
ship_df = gpd.read_file('data/darmc_historical_shipwrecks_500bce_1500ce.geojson').to_crs(3035)
countries_df = gpd.read_file('data/natural_earth_world_boundaries_50m_2018.geojson').to_crs(3035)
```

Shfaqni një shembull të të dhënave

```
ship_df.sample(5)
```

Shfaqni një shembull të të dhënave

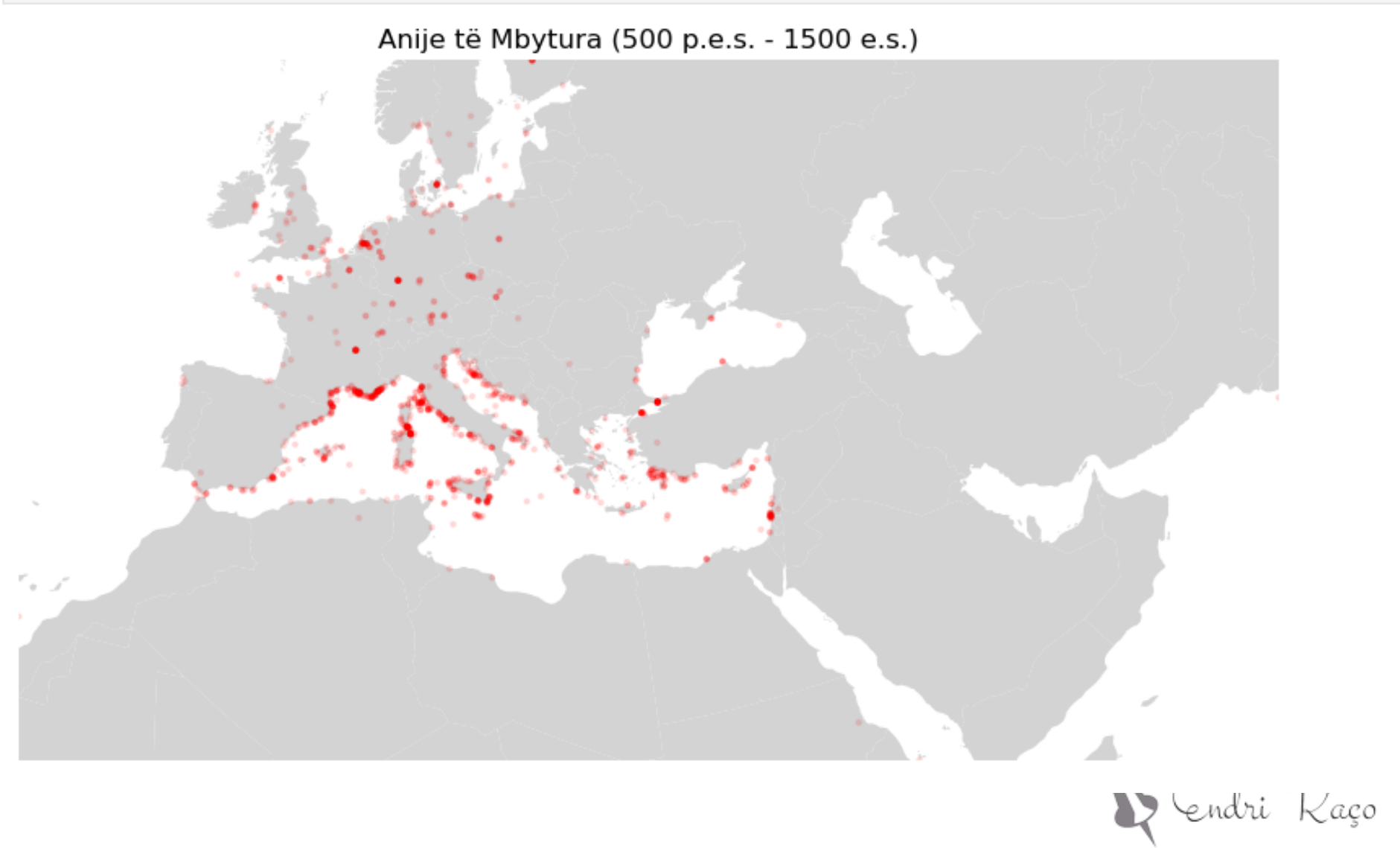
]:

	2010_wreck_id	name_1	name_2	start_date	end_date	year_found	depth	depth_q	length	width	cargo_1	type_1	cargo_2	type_2	cargo_3	type_3	other_ca
660	482.0	Macchia Tonda, La	None	50.0	100.0	None	12.0	None	None	NaN	amphoras	Dr14, pear-shaped; flat-bottomed amphora	None	None	None	None	N
276	702.0	Pomègues 1	None	200.0	300.0	None	7.0	None	None	NaN	amphoras	Almagro 50 and LaubenheimerG4	ceramic	pottery medallion	coins	sestertius of Antoninus Pius (145-161), middle...	la
147	369.0	Grazel 2	None	631.0	631.0	None	NaN	None	None	NaN	metal	bronze pots, box, strainer, lamp, fittings	coins	coins ending 630-1 AD	None	None	N
891	428.0	Kornat	None	1.0	100.0	None	NaN	None	None	NaN	amphoras	None	None	None	None	None	N
909	1009.0	Vis 7	None	1.0	500.0	None	NaN	None	None	NaN	amphoras	None	None	None	None	None	N

Vizualizimi i Pikave

```
import geoplot
import matplotlib.pyplot as plt

# Përkufizoni kanavacën
f, ax = plt.subplots(figsize=(10,7))

# Vizualizoni dy shtresa
countries_df.plot(ax=ax, color='lightgray', edgecolor="none", linewidth=.5)
geoplot.pointplot(ship_df, s=2, color='red', ax=ax, alpha=.1)

# Vendosni kufijtë e hartës
# Krijoni një buffer për të shtuar një margjinë
buff = ship_df.buffer(1)
xlim = ([buff.total_bounds[0], buff.total_bounds[2]])
ylim = ([buff.total_bounds[1], buff.total_bounds[3]])
ax.set_xlim(xlim)
ax.set_ylim(ylim)

# Vendosni titullin e hartës
ax.set_title('Anije të Mbytura (500 p.e.s. - 1500 e.s.)')

# Shfaqni rezultatet
plt.show()
```

Vizualizimi i Pikave



Histogram 2D

- Një mënyrë më e mirë për të përfaqësuar një densitet hapësinor është një histogram dy-dimensional (hist2d), i njohur gjithashtu si një grafik rrjeti.

Histogram 2D

- Vini re se një nga avantazhet e Python është mundësia e ndryshimit të parametrave të një funksioni përmes një cikli for (për shembull, numri i shtyllave në një histogram) dhe krahasimi i rezultateve.

Histogram 2D

```
# riprojektojmë në lon/lat për të pasur koordinata më të interpretueshme
ship_df = ship_df.to_crs(4326)

# le të luajmë me numrin e bins:
for bin_n in [10,20,30,40]:
    print("bin_n",bin_n)
    h = plt.hist2d(ship_df.geometry.x, ship_df.geometry.y, bins=bin_n, density=False)
    plt.colorbar(h[3])
    plt.title('2D histograma e anijeve (bins='+str(bin_n)+")")
    plt.show()
```

Histogram 2D

Këto grafikë tregojnë praninë e një zone me densitet jashtëzakonisht të lartë midis Francës, Korsikës dhe Italisë:

Grafiku KDE

- Një qasje më shkencore është vlerësimi i densitetit të bërthamës (KDE).
- **geoplot.kdeplot(...)** mund të vizatojë një KDE duke u nisur nga të dhënat e pikës.

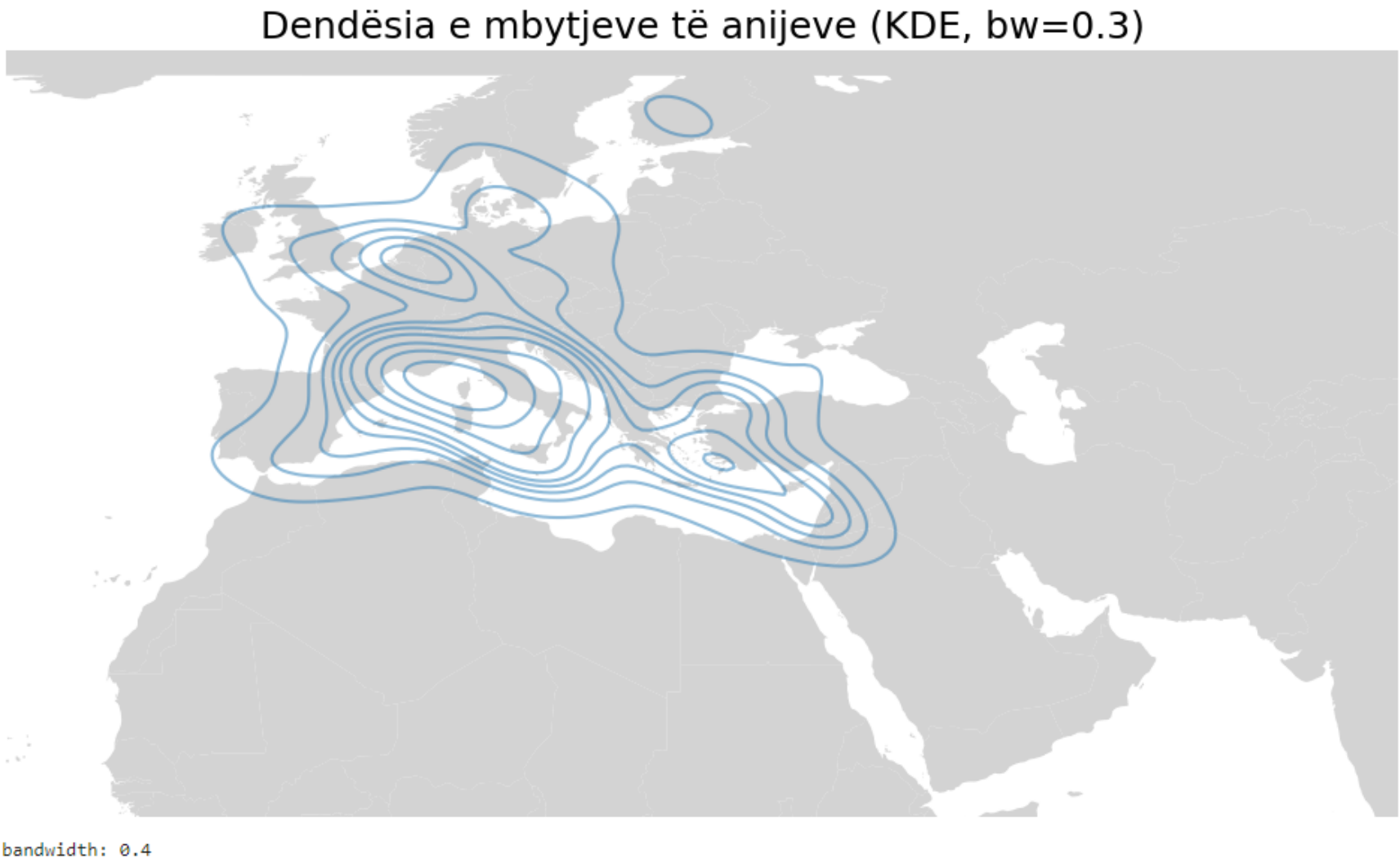
Grafiku KDE

- Një parametër vendimtar është **bandwidth** (bw), që është pragu i distancës që përdoret për të prodhuar sipërfaqen (distancat më të shkurtra rezultojnë në një sipërfaqe më të detajuar):

Grafiku KDE

```
# transformojmë në lon/lat
ship_df_ll = ship_df.to_crs(4326)

# gjenerojmë KDE me bandwidth të ndryshëm
for bandwidth in [.1, .2, .3, .4]:
    print("bandwidth:",bandwidth)
    # konturet e KDE
    ax = geoplot.kdeplot(ship_df_ll, shade=False, bw=bandwidth, figsize=(12, 12), alpha=.5)
    # shtojmë vijën bregdetare
    countries_df.to_crs(4326).plot(ax=ax, color='lightgray', edgecolor="none", linewidth=.5)
    # shtojmë titull
    plt.title('Dendësia e mbytjeve të anijeve (KDE, bw='+str(bandwidth)+")", fontsize=18)
    # figura
    plt.show()
```

Grafiku KDE



Endri Raço

Analiza e të dhënave

- Këta grafikë KDE tregojnë se dataset-i ka një përqendrim shumë të lartë të pikave në Detin Mesdhe, midis Francës Jugore, Korsikës dhe Bregut Perëndimor të Italisë.
- Në të gjitha grafikët, kjo qendër graviteti shfaqet qartë.

Analiza e të dhënave

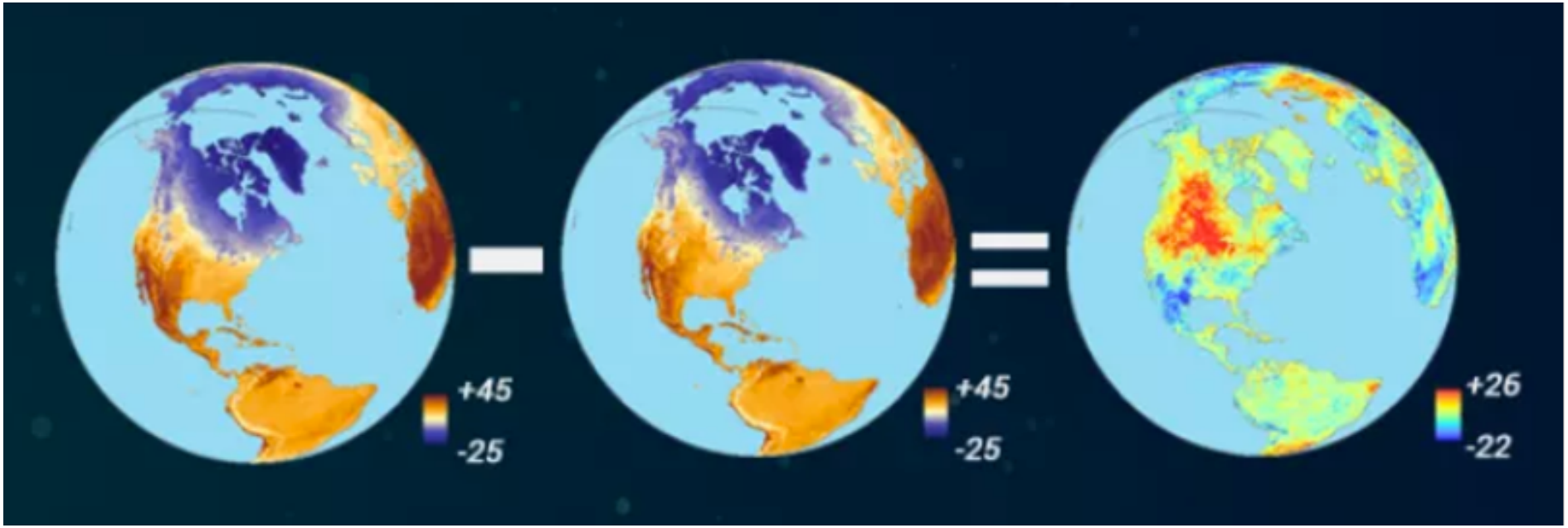
- Në aspektin shkencor, kjo mund të tregojë se kishte shumë më tepër mbytje anijesh aty se gjetkë, ose (më e mundshme) që të dhënat historike janë më të pasura dhe më të hollësishme për atë zonë.

Section 2

Algjebra e Hartave

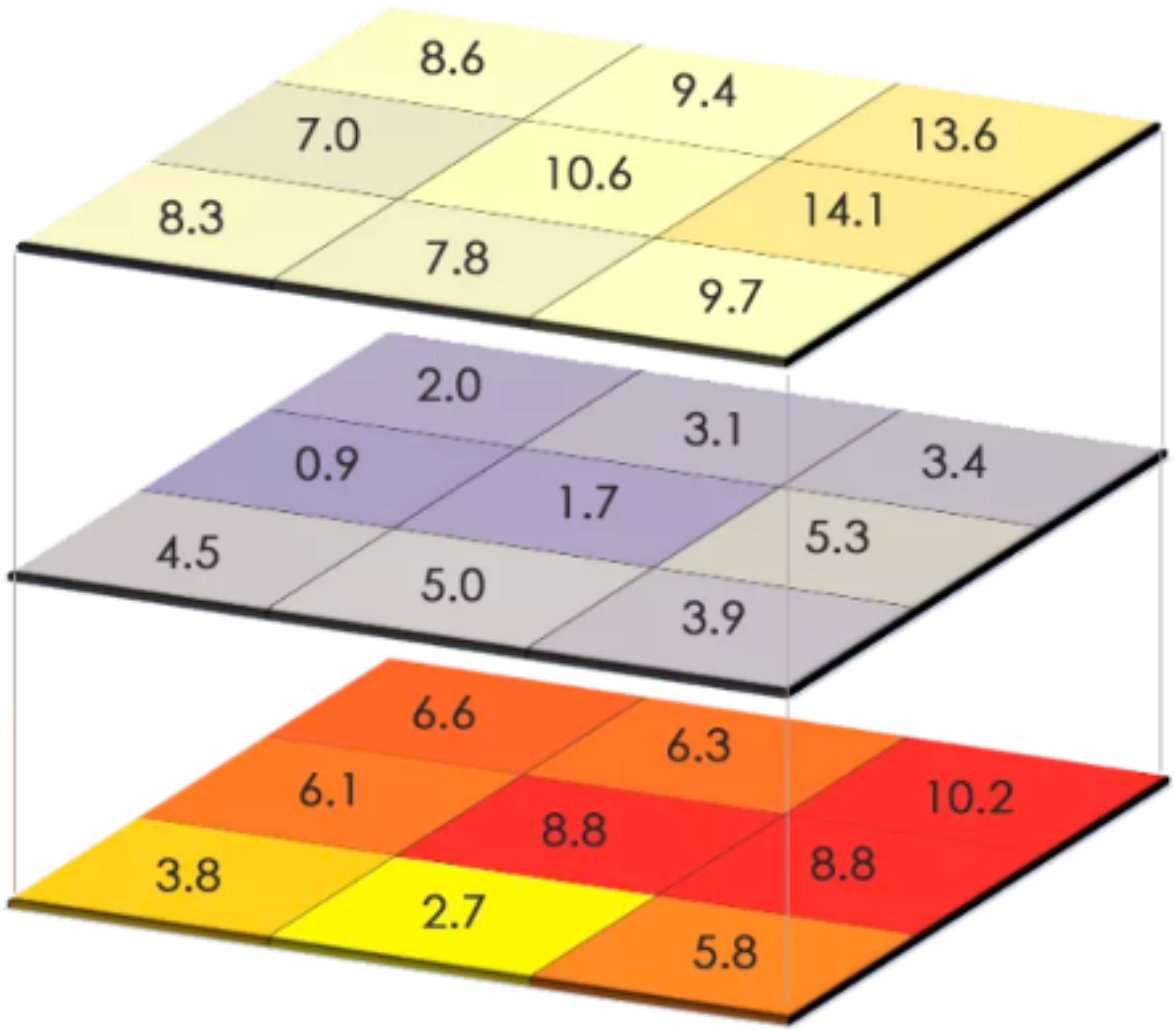
Algjebra e Hartave

- Termi “algjebra e hartave” i referohet idesë së aplikimit të operacioneve algjebrike në dataset-e raster.
- Për shembull, mund të dëshirojmë të zbresim nga njëri- tjetri dy rastera të temperaturës të kapur në kohë të ndryshme për të vëzhguar ndryshimin e temperaturës:


+45
-25
+45
-25
+26
-22

Algjebra e Hartave

Në praktikë, ky është një operacion aritmetik i aplikuar në çdo qelizë të të dy raster-ve:



Algjebra e hartave në Python

- Kur aksesojmë raster me **rasterio** ose **gdal**, ne mund të kryejmë çdo lloj llogaritjeje algjebrike lineare mbi të dhënat duke përdorur **numpy, scipy** dhe shumë paketa të tjera të fuqishme të Python.
- Statistikat zonale suportohen në librarinë **rasterstats**.

Algjebra e hartave në Python

- Kjo është arsyeja kryesore pse Python përdoret gjerësisht në komunitetet e remote sensing, machine learning, dhe AI.

Ngarkoni të dhënat e temperaturës

- Si një shembull, le të shkarkojmë dhe vizualizojmë dy datasete raster që përfaqësojnë temperaturën mesatare në vitin 2000 dhe 2017.

Ngarkoni të dhënat e temperaturës

```
import urllib.request

# Define new URLs and file names
url_2000 = "https://github.com/endri81/instatgis/blob/master/data/gis4/air_temp_2000-average.tif?raw=true"
url_2017 = "https://github.com/endri81/instatgis/blob/master/data/gis4/air_temp_2017-average.tif?raw=true"
file_name_2000 = 'data/air_temp_2000-average.tif'
file_name_2017 = 'data/air_temp_2017-average.tif'

# Download the files
urllib.request.urlretrieve(url_2000, file_name_2000)
urllib.request.urlretrieve(url_2017, file_name_2017)
```

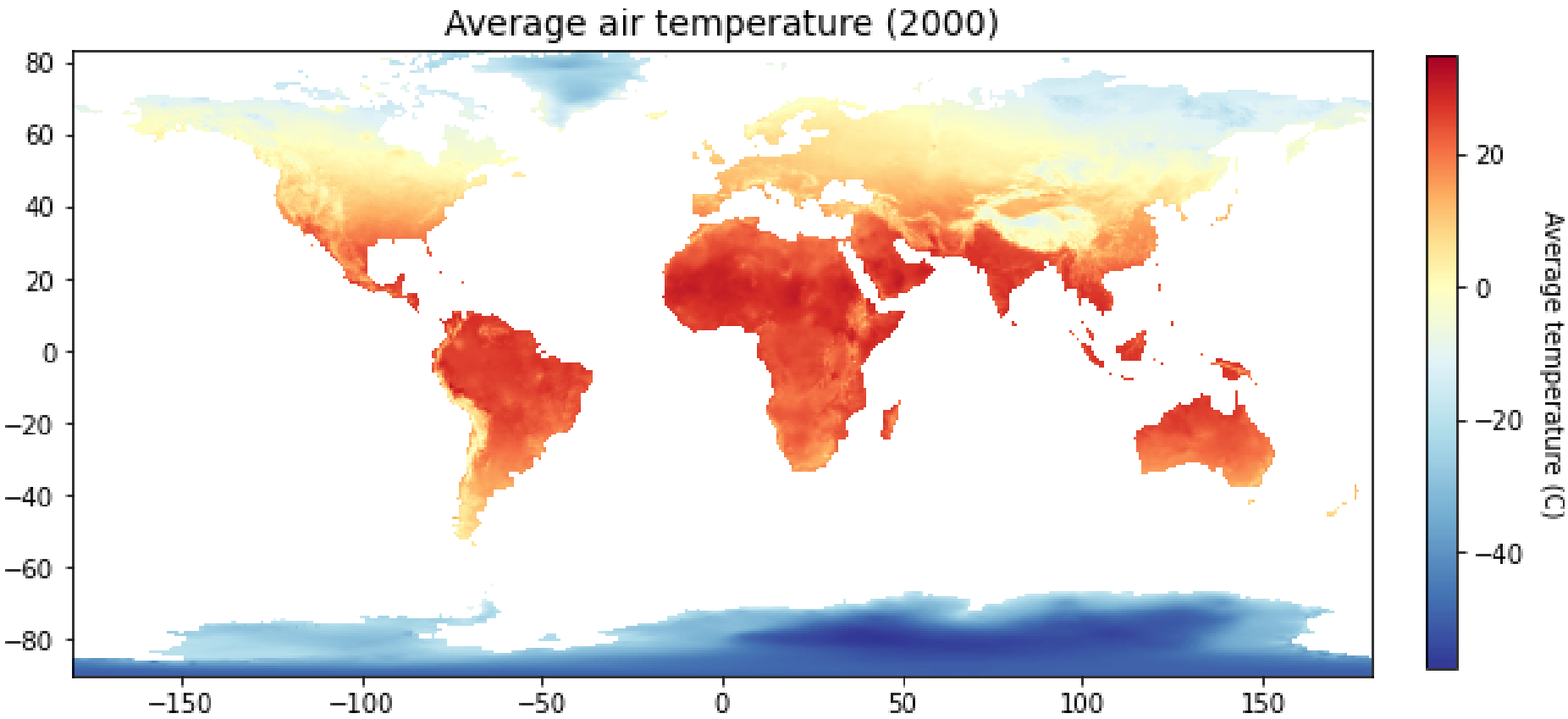
Ngarkoni të dhënat e temperaturës

```
temp00 = rasterio.open('data/air_temp_2000-average.tif')
print(temp00.meta)
temp17 = rasterio.open('data/air_temp_2017-average.tif')
print(temp17.meta)
```

Ngarkoni të dhënat e temperaturës

```
import matplotlib.pyplot as plt
import rasterio
from matplotlib.colors import TwoSlopeNorm
# Vini re diverge_zero: kjo përdoret sepse temperatura në Celsius mund të vizualizohet si diverguese nga zero
plot_raster(temp00, temp00.read(1, masked=True), 'Temperatura mesatare e ajrit (2000)', 'Temperatura mesatare (
    'RdYlBu_r', diverge_zero=True)
plot_raster(temp17, temp17.read(1, masked=True), 'Temperatura mesatare e ajrit (2017)', 'Temperatura mesatare (
    'RdYlBu_r', diverge_zero=True)
```

Ngarkoni të dhënat e temperaturës



Krahasimi i të dhënave raster

- Vizualisht, nuk është e mundur të dallohen ndryshimet midis të dhënave të vitit 2000 dhe atyre të vitit 2017.
- Prandaj, do të zbresim dy rasterat e temperaturës, duke përdorur Algjebrën e Hartave.

Krahasimi i të dhënave raster

- Në praktikë, Python lejon të bëhet kjo në mënyrë intuitive si **raster_vals2 - raster_vals1**.
- Këto janë operacione algjebrike lineare të aplikuara në çdo qelizë të matricave.

Krahasimi i të dhënave raster

- Pastaj do të ndërtojmë një histogram të vlerave dhe raster-it, duke treguar se temperaturat mesatare janë më të larta me 0.5 gradë, me disa raste ekstreme pozitive dhe negative që mund të shkaktohen nga gabimet e sensorëve.

Krahasimi i të dhënave raster

Rezultati do të ruhet në një skedar të ri raster, duke ripërdorur metadatat nga raster-at hyrës.

Krahasimi i të dhënave raster

```
vals17 = temp17.read(1, masked=True)
vals00 = temp00.read(1, masked=True)
```

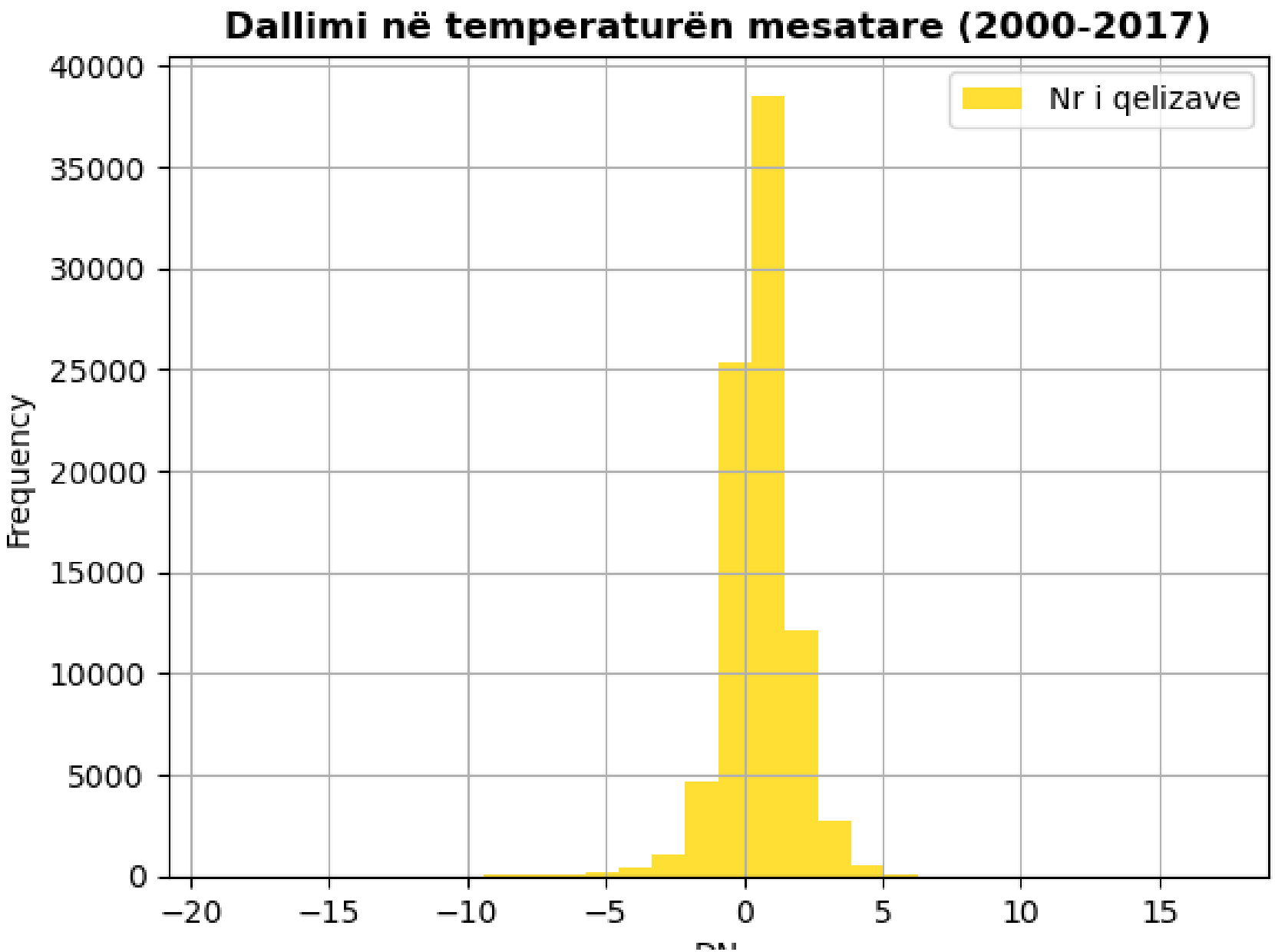
Krahasimi i të dhënave raster

```
# zbresim të dy raster-at
vals_diff = vals17 - vals00
print("Statistikat e Diferencës:", vals_diff.min(), round(vals_diff.mean(), 2), vals_diff.max())
print("Diferenca midis mesatareve:", round(vals17.mean()-vals00.mean(), 3))
```

Krahasimi i të dhënave raster

```
# vizatoni histogramin
show_hist(vals_diff, bins=30, lw=0.2, stacked=False, alpha=0.8, label='Nr i qelizave',
    histtype='stepfilled', title="Dallimi në temperaturën mesatare (2000-2017)")
```

Krahasimi i të dhënave raster



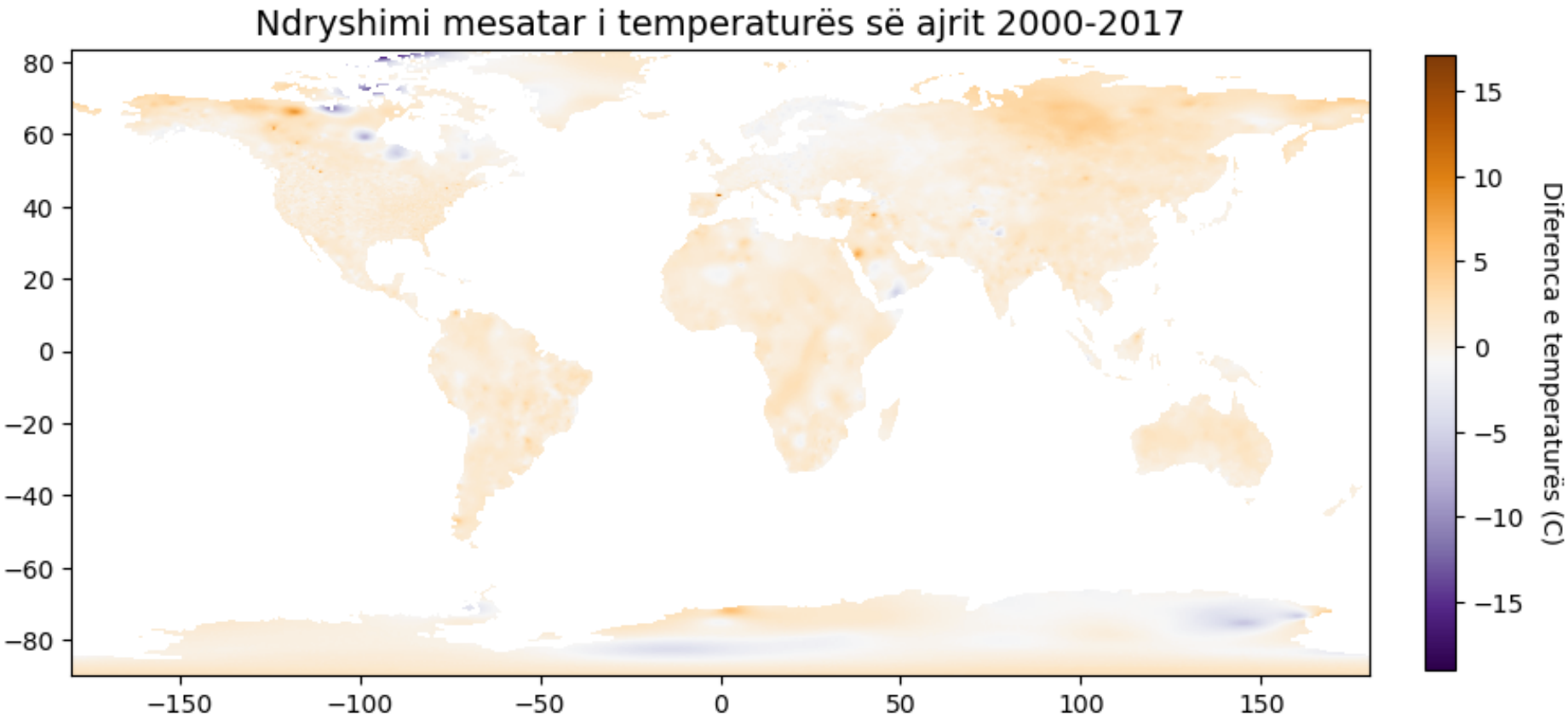
Ndërtojmë rasterin

- Cmap (Purple - White - Orange) thekson vlerat ekstreme, duke fshehur zonat ku vlerat nuk divergojnë.

Ndërtojmë rasterin

```
plot_raster(temp17, vals_diff, 'Ndryshimi mesatar i temperaturës së ajrit 2000-2017',
            'Diferenca e temperaturës (C)', 'PuOr_r')
```

Ndërtojmë rasterin



Ruani rezultatin në një skedar raster

- Është e rëndësishme të specifikohen metadatat nga skedarët hyrës, përfshirë CRS, vlerën NODATA dhe transformimin e koordinatave gjeografike:

Ruani rezultatin në një skedar raster

```
fout = 'tmp/air_temp_diff_2000_2017.tif'
ds = rasterio.open(fout, 'w',
    driver='GTiff', # formati i skedarit të daljes
    height=vals_diff.shape[0], # madhësia e matricës
    width=vals_diff.shape[1], # madhësia e matricës
    count=1, # numri i bandave
    dtype=vals_diff.dtype, # lloji i të dhënave (p.sh., pikë lundruese)
    crs=temp17.crs, # CRS (p.sh., Lambert, WGS84, UTM, etj.)
    nodata=temp17.nodata, # vlera e përdorur për të përfaqësuar NO DATA
    transform=temp17.transform # transformimi i koordinatave gjeografike
)

ds.write(vals_diff, 1)
ds.close()
print("Raster u ruajt te", fout, '.')
```

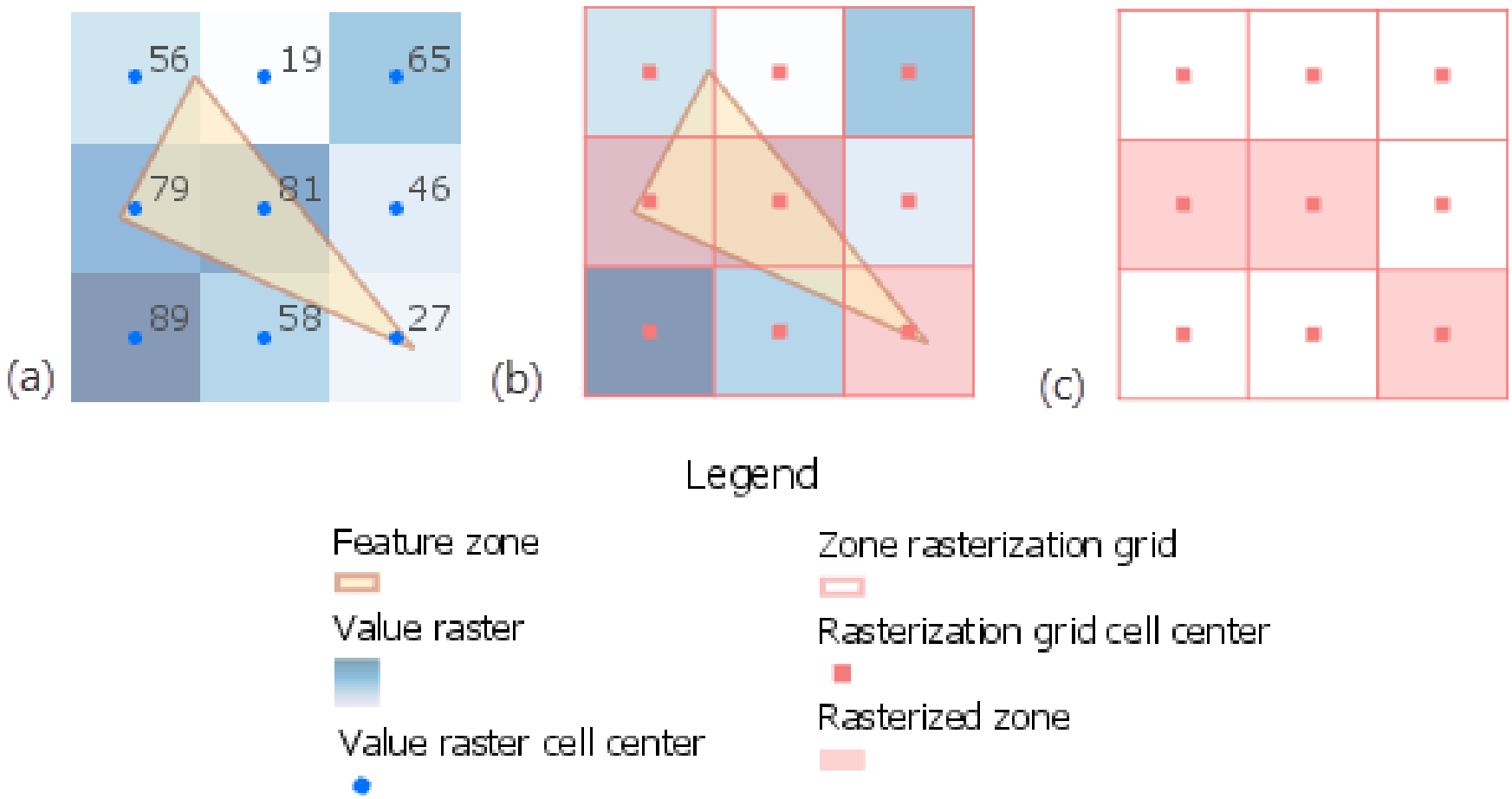
Statistikat zonale

- Kur dëshirojmë të llogarisim statistikat raster bazuar në një zonë gjeografike, na duhen **statistikat zonale**.
- Për shembull, mund të dëshirojmë të llogarisim lartësinë mesatare (vlerat) e çdo rrethi (zonave) në Angli.

Statistikat zonale

- Si skedar **input**, statistikat zonale kanë nevojë për një raster që përfaqëson vlerat dhe një grup tjetër të dhënash që përfaqëson zonat për të cilat duam të llogarisim statistikat:

Statistikat zonale



Statistikat zonale

- Në këtë shembull, ne do të përdorim të dhënat evropiane të NOx të përdorura më sipër si vlera dhe zonat statistikore evropiane (NUTS)

Shkarkojmë datat

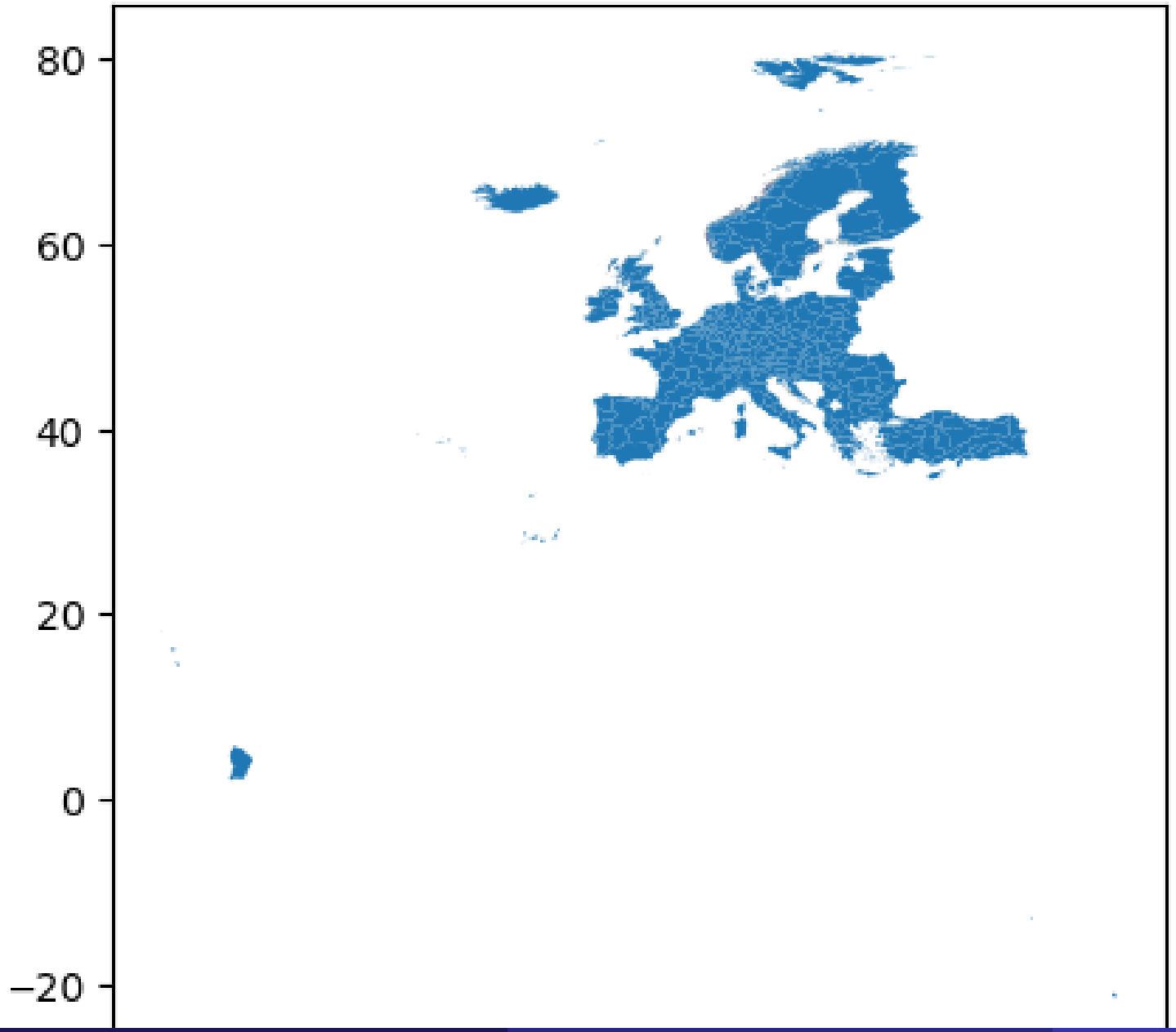
```
# shkarkoni kufijtë rajonalë të BE-së (niveli NUTS 2, 2021)
nuts2_file = 'data/NUTS_RG_01M_2021_4326_LEVL_2.geojson.gz'
url = 'https://raw.githubusercontent.com/endri81/instatgis/master/data/gis4/NUTS_RG_01M_2021_4326_LEVL_2.geojso
urllib.request.urlretrieve(url, nuts2_file)
```

Shkarkojmë datat

Skedari është gzip dhe mund ta hapim direkt me **gzip.open(...)**

```
import geopandas as gpd
import gzip
nuts2_df = gpd.read_file(gzip.open(nuts2_file))
nuts2_df.plot()
```

Shkarkojmë datat



dri Raço

Kontrollojmë

```
nuts2_df.sample(5)
```

Përgatitja e të Dhënave

- Duke qenë se nuk kemi të dhëna për Guyana Franceze dhe territore të tjera më të vogla, mund t'i heqim ato nga dataset-i.

Në dataframe pandas, mund të shkruajmë kushte në mënyra të ndryshme:

- `column.str.contains(string)` kryen një përputhje të pjesshme në një kolonë me format tekst
- `column.isin(list)` kryen një përputhje të saktë në përmbajtjen e një kolone ndaj një liste
- `~` do të thotë "jo" (vetëm në kontekstin e pandas)

Përgatitja e të Dhënave

- Do projektojmë kufijtë dhe ti ruajmë ato në një GeoPackage.
- Kini parasysh se GeoJSON lejon vetëm gjeometri në WGS84 (4326).
- Kur keni një CRS tjetër, duhet të përdorni një GeoPackage.

Përgatitja e të Dhënave

```
# Kjo shprehje do të thotë:
# zgjidhni rreshta ku NUTS_ID nuk përmban 'FRY'
nuts2_df = nuts2_df[~nuts2_df['NUTS_ID'].str.contains("FRY")]

# hiqni rreshtat me kode që korrespondojnë me ishujt për të cilët nuk kemi të dhëna:
nuts2_df = nuts2_df[~nuts2_df['NUTS_ID'].isin(['PT20','PT30','ES70','NOOB'])]

# projektoni në Lambert (i përshtatshëm për Evropën)
nuts2_df = nuts2_df.to_crs(3035)
nuts2_df.info()

# Ky është një rregullim: dataframe gjeo kanë nevojë që FID të jetë i tipit integer
nuts2_df['FID'] = nuts2_df.index
```

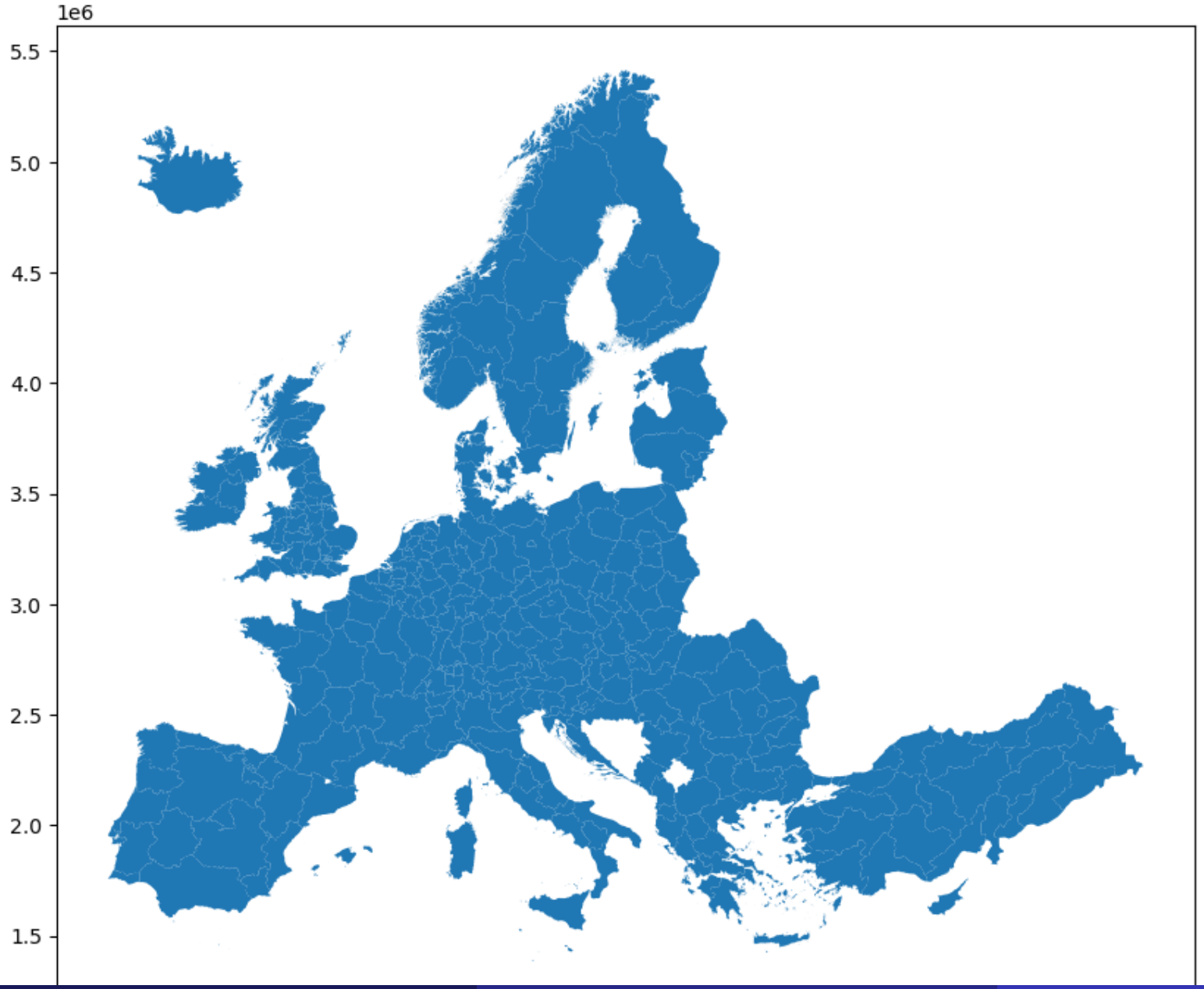
Ruani këtë dataset në një skedar

```
nuts2_clean_file = "tmp/nuts2_boundaries.gpkg"
nuts2_df.to_file(nuts2_clean_file, driver="GPKG")
```

Vizatoni gjeometrinë

```
nuts2_df.plot(figsize=(10,10))
```

Vizatoni gjeometrinë



Endri Raço

Statistikat Zonale në Python

- Statistikat zonale mund të llogariten me funksionin **rasterstats.zonal_stats**
- Parametri `stats` tregon cilat statistika dëshirojmë të llogariten në çdo zonë.

Statistikat Zonale në Python

- Do të llogarisim disa statistika zonale dhe do ta ruajmë rezultatin në një GeoPackage dhe një skedar CSV:

Statistikat Zonale në Python

```
from rasterstats import zonal_stats

print("Duke llogaritur statistikat zonale midis", nuts2_clean_file, 'dhe data/eu-2016-nox_avg.tif ...')

zon_stats = zonal_stats(nuts2_clean_file, 'data/eu-2016-nox_avg.tif',
                        stats="count min mean max median", geojson_out=True)
print('Përfunduar.')
```

Statistikat Zonale në Python

- Rezultati është një listë e fjalorëve që përmban statistikat për çdo rresht të skedarit hyrës të vektorit
- Këto rezultate mund të konvertohen në një kornizë të të dhënave gjeo kështu:

Statistikat Zonale në Python

```
import geopandas as gpd
stats_df = gpd.GeoDataFrame.from_features(zon_stats)

# riemërtoni kolonat për të qenë më kuptimplota
stats_df = stats_df.rename(columns={"min": "nox_min",
                                    "max": "nox_max",
                                    "count": "nox_count",
                                    "mean": "nox_mean",
                                    "median": "nox_median"})
stats_df.sample(4)
```

Statistikat Zonale në Python

	geometry	id	COAST_TYPE	MOUNT_TYPE	NAME_LATN	CNTR_CODE	NUTS_ID	NUTS_NAME	LEVL_CODE	URBN_TYPE	n
56	MULTIPOLYGON (((3735722.439 1899591.154, 37359...	ES53	NaN	0	Illes Balears	ES	ES53	Illes Balears	2	NaN	
90	POLYGON ((5080508.322 3065872.586, 5080260.235...	PL21	NaN	0	Małopolskie	PL	PL21	Małopolskie	2	NaN	
111	POLYGON ((4384447.442 3167714.424, 4384669.555...	DEG0	NaN	0	Thüringen	DE	DEG0	Thüringen	2	NaN	
185	MULTIPOLYGON (((4545500.129 2265401.004, 45461...	ITI2	NaN	0	Umbria	IT	ITI2	Umbria	2	NaN	

Statistikat Zonale në Python

```
# ruajmë rezultatin në një geopackage
stats_df.to_file('tmp/eu_nox_2016_nuts2.gpkg', driver="GPKG")

# për lehtësi ruajmë tabelën e atributeve si CSV
stats_df.drop(columns=['geometry']).to_csv('tmp/eu_nox_2016_nuts2.csv', index=False)
print("results saved.")
```

Renditja dhe Vizualizimi i Rezultateve

- Tani, ne mund të eksplorojmë dhe vizualizojmë rezultatet.
- Funksioni **.rank()** i pandas na lejon të renditim vlerat.

Renditja dhe Vizualizimi i Rezultateve

- Shpesh është një ide e mirë të ndajmë në qeliza të ndryshme llogaritjet dhe vizualizimet e gjata.
- Në këtë rast, nëse dëshirojmë të ekzekutojmë vizualizime të ndryshme, nuk kemi nevojë të rikthejmë llogaritjet zonale në qelizën e mëparshme.

Renditja dhe Vizualizimi i Rezultateve

```
nox_nuts2_df = gpd.read_file('tmp/eu_nox_2016_nuts2.gpkg')
print(nox_nuts2_df.describe())
print(nox_nuts2_df.columns)
```

Renditja dhe Vizualizimi i Rezultateve

	COAST_TYPE	MOUNT_TYPE	LEVL_CODE	URBN_TYPE	nox_min	nox_max
count	6.0	325.0	325.0	6.0	325.000000	325.000000
mean	0.0	0.0	2.0	0.0	4.336815	29.984501
std	0.0	0.0	0.0	0.0	5.983261	18.589846
min	0.0	0.0	2.0	0.0	0.050000	1.061000
25%	0.0	0.0	2.0	0.0	0.050000	18.127001
50%	0.0	0.0	2.0	0.0	1.395000	23.889000
75%	0.0	0.0	2.0	0.0	7.528000	34.648998
max	0.0	0.0	2.0	0.0	47.188000	118.250999

	nox_mean	nox_count	nox_median
count	325.000000	325.000000	325.000000
mean	11.772977	4428.744615	11.407328
std	7.030471	5540.615930	6.949615
min	0.083497	3.000000	0.050000
25%	7.030450	1301.000000	6.784000
50%	10.620482	2840.000000	10.453000
75%	13.782574	5964.000000	13.590500
max	54.289036	56773.000000	52.843498

Renditja dhe Vizualizimi i Rezultateve

```
# Rendisni rajonet: 1 = vlera më e lartë
nox_nuts2_df['nox_mean_rank'] = nox_nuts2_df['nox_mean'].rank(ascending=False)
nox_nuts2_df['nox_max_rank'] = nox_nuts2_df['nox_max'].rank(ascending=False)
```

Renditja dhe Vizualizimi i Rezultateve

```
# E vërtetë nëse vlera > 40
nox_nuts2_df['nox_max_high'] = nox_nuts2_df['nox_max'] > 40
```

Renditja dhe Vizualizimi i Rezultateve

```
# Shikoni rajonet më të ndotura NUTS, duke përzgjedhur vetëm kolonat përkatëse
sel_df = nox_nuts2_df[['NUTS_ID','NUTS_NAME','nox_mean','nox_max',
                       'nox_mean_rank','nox_max_rank','nox_max_high']]
sel_df.sample(5)

# Rendisni rajonet sipas nox_mean
sel_df.sort_values('nox_mean',ascending=False).head(20)
```

Renditja dhe Vizualizimi i Rezultateve

	NUTS_ID	NUTS_NAME	nox_mean	nox_max	nox_mean_rank	nox_max_rank	nox_max_high
234	UKI3	Inner London — West	54.289036	60.915001	1.0	23.0	True
235	UKI4	Inner London — East	48.673785	60.771000	2.0	24.0	True
233	UKI7	Outer London — West and North West	42.045606	60.487999	3.0	25.0	True
41	BE10	Région de Bruxelles-Capitale/ Brussels Hoofdst...	40.851095	49.681000	4.0	40.0	True
236	UKI5	Outer London — East and North East	33.673233	53.467999	5.0	34.0	True
42	BE21	Prov. Antwerpen	30.655168	86.754997	6.0	6.0	True
171	NL33	Zuid-Holland	29.971545	79.961998	7.0	13.0	True
286	UKI6	Outer London — South	29.414680	52.473000	8.0	36.0	True
138	DEA1	Düsseldorf	29.366611	53.619999	9.0	33.0	True
232	UKG3	West Midlands	28.163673	35.469002	10.0	76.0	False
177	ITC4	Lombardia	26.701113	84.431000	11.0	8.0	True
230	UKD7	Merseyside	26.619785	59.910000	12.0	27.0	True
102	ES63	Ciudad de Ceuta	26.568334	33.310001	13.0	93.0	False
169	NL31	Utrecht	26.447465	37.550999	14.0	67.0	False
322	TR10	İstanbul	25.629883	118.250999	15.0	1.0	True
242	UKD3	Greater Manchester	25.620574	42.200001	16.0	57.0	True
217	NL41	Noord-Brabant	24.854877	81.477997	17.0	12.0	True
153	HU11	Budapest	24.001561	30.010000	18.0	113.0	False
218	NL42	Limburg (NL)	23.951939	30.733000	19.0	107.0	False
45	AT13	Wien	23.914827	34.138000	20.0	86.0	False

Renditja dhe Vizualizimi i Rezultateve

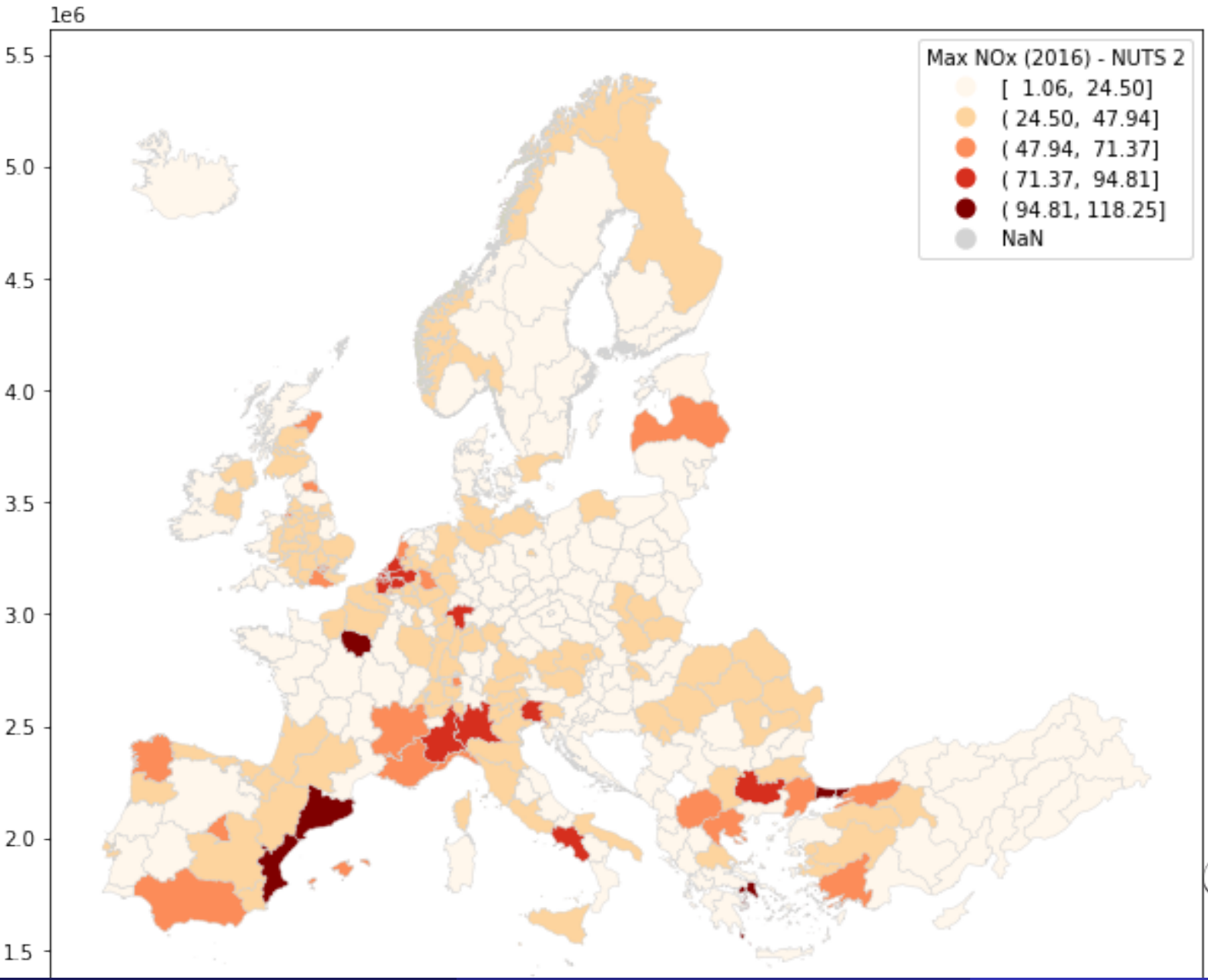
- Vini re se si renditja për **nox_max** dhe **nox_mean** mund të ndryshojë:
- Për shembull, Inner London - West ("UKI3") ka nivelin më të lartë të NOx në mesatare, por është vetëm i 23-ti për sa i përket vlerës maksimale.

Renditja dhe Vizualizimi i Rezultateve

- Mund të vizatojmë vlerat e grumbulluara me një choropleth:

```
nox_nuts2_df.plot(column='nox_max', figsize=(12,9), scheme='equalinterval', cmap='OrRd', k=5,
    edgecolor="lightgrey", linewidth=0.4,
    legend=True, legend_kwds={'loc': 'upper right', 'title': 'NOx Maksimale (2016) - NUTS 2'},
    missing_kwds={'color': "lightgrey"})
```

Renditja dhe Vizualizimi i Rezultateve



Endri Raço

Renditja dhe Vizualizimi i Rezultateve

- Shpërndarja e NOx është shumë heterogjene hapësinore (dmth., ndryshon shumë në çdo vend).
- Ne mund të përdorim **groupby** për të gjetur njësinë kryesore të NUTS për çdo vend për sa i përket NOx maksimale:

Renditja dhe Vizualizimi i Rezultateve

```
nox_nuts2_df['nox_max_country_rank'] = nox_nuts2_df.groupby('CNTR_CODE')['nox_max'].rank(ascending=False)

# për çdo vend, njësitë renditen në mënyrë të brendshme
sel_df = nox_nuts2_df[['NUTS_ID','NUTS_NAME','nox_max','nox_max_country_rank']]
sel_df
```

Renditja dhe Vizualizimi i Rezultateve

```
# Zgjidhni vetëm njësinë kryesore për çdo vend (rank==1) dhe rendisni ato sipas NOx maksimale:
top_df = sel_df[sel_df['nox_max_country_rank']==1]
top_df.sort_values('nox_max', ascending=False)
```

Renditja dhe Vizualizimi i Rezultateve

	NUTS_ID	NUTS_NAME	nox_max	nox_max_country_rank
322	TR10	İstanbul	118.250999	1.0
14	ES51	Cataluña	108.094002	1.0
74	EL30	Αττική	106.764000	1.0
136	FR10	Ile-de-France	99.392998	1.0
42	BE21	Prov. Antwerpen	86.754997	1.0
177	ITC4	Lombardia	84.431000	1.0
43	BG42	Южен централен	83.987000	1.0
204	NL34	Zeeland	81.850998	1.0
85	DE71	Darmstadt	73.099998	1.0
234	UKI3	Inner London — West	60.915001	1.0
26	CH04	Zürich	56.997002	1.0
213	MK00	Северна Македонија	49.800999	1.0