

Leksioni 8

Endri Raco

05 May, 2024



1 Të dhënat Raster (vazhdim)

2 Algjebra e Hartave

3 Praktikë



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_1500ce.geojson'
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_2018.geojson'
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...	lan
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()
```



Anije të Mbytura (500 p.e.s. - 1500 e.s.)



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



- 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()
```



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



- 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.



- 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):

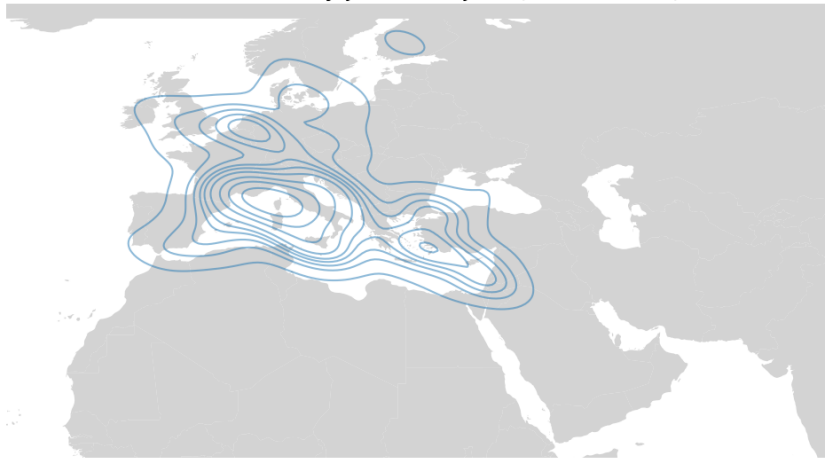


```
# transformojmë në lon/lat
ship_df_ll = ship_df.to_crs(4326)

# 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()
```



Dendësia e mbytjeve të anijeve (KDE, bw=0.3)



bandwidth: 0.4

- 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.



- 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.



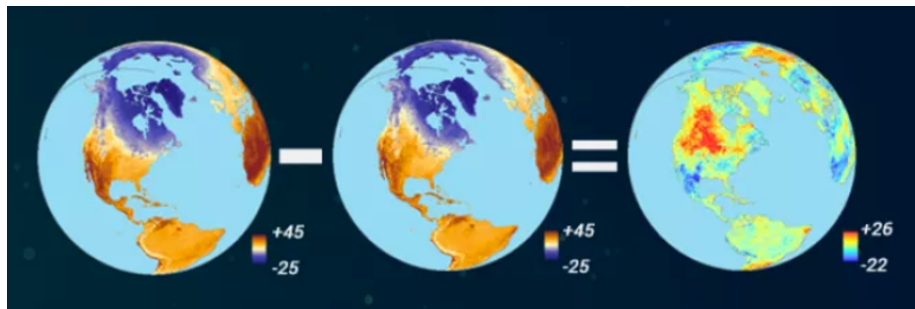
Section 2

Algjebra e Hartave



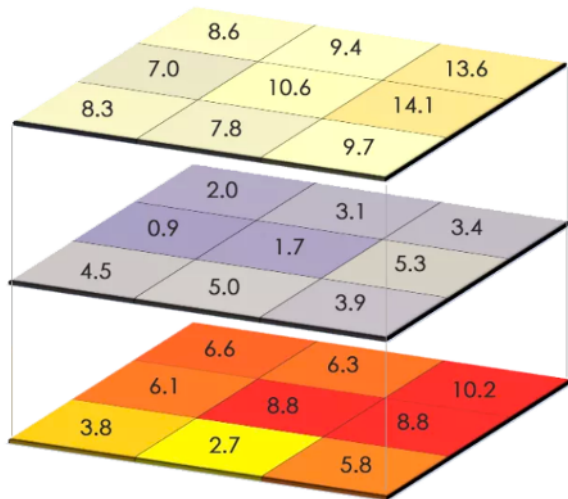
- Termi “algjebra e hartave” i referohet idesë së aplikimit të operacioneve algjebrike në dataset-e raster.
- Për shembull, mund të dëshirojmë të zbresim nga njëri- tjetri dy rastera të temperaturës të kapur në kohë të ndryshme për të vëzhguar ndryshimin e temperaturës:





Algjebra e Hartave

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



- 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**.



- 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)
```

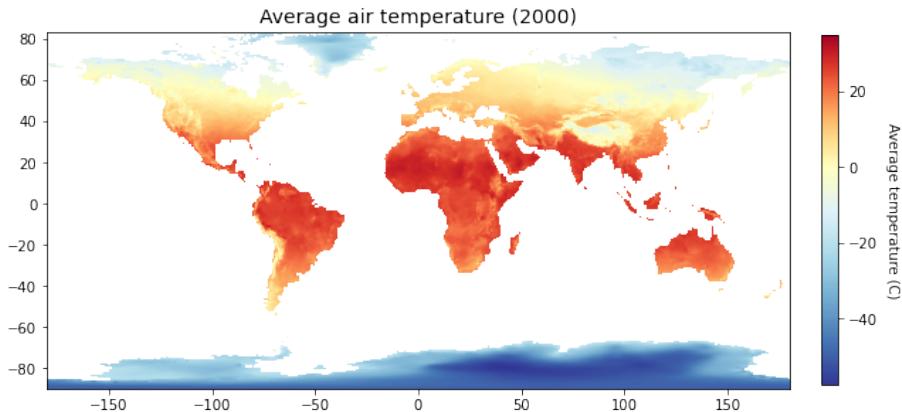


Ngarkoni të dhënat e temperaturës

```
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 (2000)',
            'RdYlBu_r', diverge_zero=True)
plot_raster(temp17, temp17.read(1, masked=True), 'Temperatura mesatare e ajrit (2017)', 'Temperatura mesatare (2017)',
            'RdYlBu_r', diverge_zero=True)
```



Ngarkoni të dhënat e temperaturës



- 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.



- 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.



- 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.



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))
```



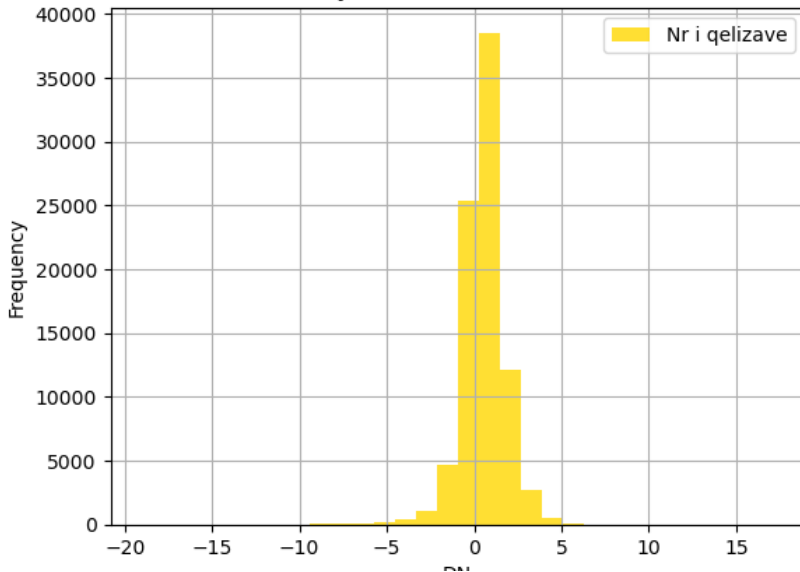
Krahasimi i të dhënave raster

```
# 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

Dallimi në temperaturën mesatare (2000-2017)



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

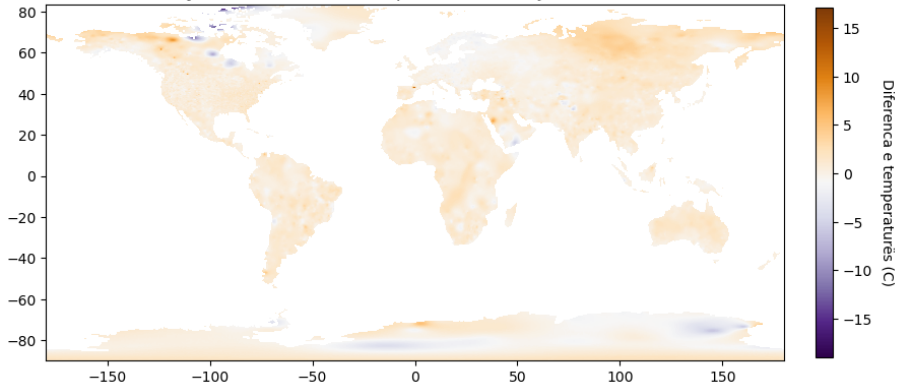


Ndërtojmë rasterin

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



Ndryshimi mesatar i temperaturës së ajrit 2000-2017



- 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, '.')
```



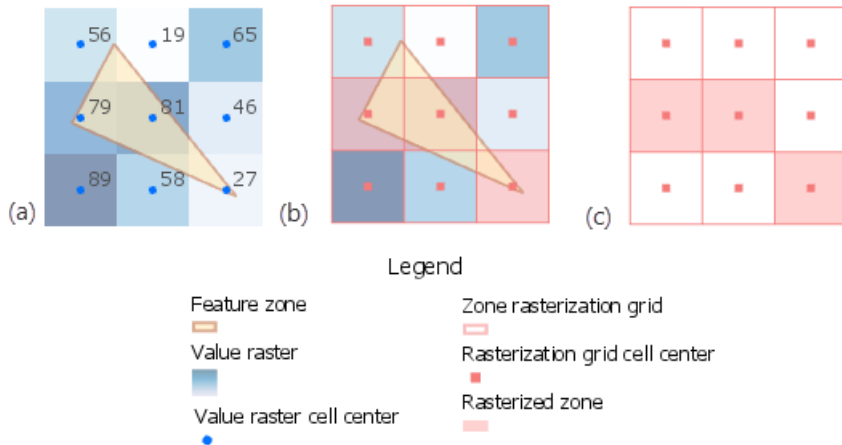
- Kur dëshirojmë të llogarisim statistikat raster bazuar në një zonë gjeografike, na duhen **statistikat zonale**.
- Për shembull, mund të dëshirojmë të llogarisim lartësinë mesatare (vlerat) e çdo rrethi (zonave) në Angli.



- 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 NO_x të përdorura më sipër si vlera dhe zonat statistikore evropiane (NUTS)

```
# 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.geojson.gz'
urllib.request.urlretrieve(url, nuts2_file)
```

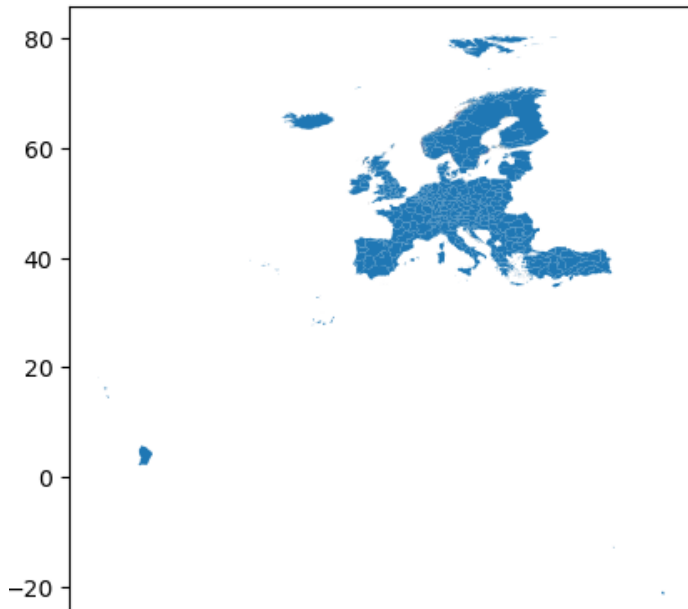


Skedari është gzip dhe mund ta hapim direkt me **gzip.open(...)**

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



Shkarkojmë datat



dri Raço

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

- 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)



- 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', 'NO0B'])]  
  
# projektioni 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")
```



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


Vizatoni gjeometrinë



Endri Raco

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



- 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.')
```



- Rezultati është një listë e fjalorëve që përmban statistikën 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)

# riemërtori 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 attributeve si CSV
stats_df.drop(columns=['geometry']).to_csv('tmp/eu_nox_2016_nuts2.csv', index=False)
print("results saved.")
```



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



- 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

- 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.

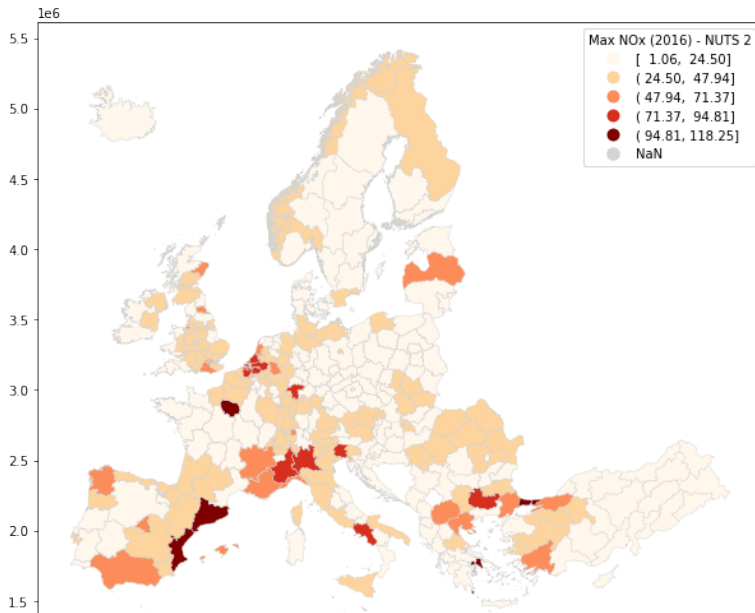


- Mund të vizatojmë vlerat e grumbulluara me një choropleth:

```
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

- 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, njësitet 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

Section 3

Praktikë



- Do ngarkojmë dataset që përmban raster me të dhëna globale të reshjeve nga viti 1950 deri në vitin 2017 në milimetra nga url https://raw.githubusercontent.com/endri81/instatgis/master/data/global_rasters/
- Duke përdorur cikle for, merrni raster-in e reshjeve çdo 5 vjet nga viti 1980 deri në vitin 2015.
- Për çdo raster, gjeneroni një grafik duke përdorur funksionin **plot_raster** (funksioni në 2 slidet e tjera) dhe printoni vlerat minimale, mesatare dhe maksimale.



Funksioni plot_raster

```
# Duke qenë se funksionaliteti i vizualizimit të rasterio është mjaft i komplikuar,  
# ne krijojmë një funksion për të vizualizuar një raster më lehtë.  
# Vini re vlerat e paracaktuara (Blues, 10, 10).  
# Sa i përket kompleksitetit, kjo është një funksion realist i përdorur në shkencën e të dhënave,  
# me "hack"-e për t'i bërë gjërat të funksionojnë për shkak të kufizimeve të paketës.
```

```
def plot_raster(rast, val_matrix, plot_title, value_label, cmap='Blues', width=10, height=10, diverge_zero=False):  
    """Vizualizon një rasterio raster me cilësime të arsyeshme dhe me legjendë.  
    @ rast: skedari rasterio (përdoret për të lexuar koordinatat gjeografike)  
    @ val_matrix: vlerat e nxjerra (përdoret për të lexuar vlerat e rasterit)  
    @ plot_title: titulli i figurës së plotë  
    @ value_label: sasia që shfaqet  
    @ diverge_zero: e vërtetë nëse përdorni një cmap divergjent për të përqendruar hartën e ngjyrave në zero  
    """  
    fig, ax = plt.subplots(figsize=(10,10))  
    # image_hidden është një "hack" për të treguar legjendën  
    if diverge_zero:  
        image_hidden = ax.imshow(val_matrix, cmap=cmap, norm=TwoSlopeNorm(0))  
    else:  
        image_hidden = ax.imshow(val_matrix, cmap=cmap)  
  
    ax.clear()
```



Funksioni plot_raster (vazhdim)

```
# vizualizoni raster: rast.transform lejon sistemin të tregojë koordinatat gjeografike
if diverge_zero:
    rast_plot = rasterio.plot.show(val_matrix, cmap=cmap, ax=ax, transform=rast.transform, norm=TwoSlopeNor
else:
    rast_plot = rasterio.plot.show(val_matrix, cmap=cmap, ax=ax, transform=rast.transform)

# vendosni titullin e grafikut
ax.set_title(plot_title, fontsize=14)

# shfaqni legjendën me etiketën
# "hack" për të rregulluar lartësinë
im_ratio = val_matrix.shape[0]/val_matrix.shape[1]
#plt.colorbar(im, fraction=0.046*im_ratio, pad=0.04)
cbar = fig.colorbar(image_hidden, ax=ax, fraction=0.046*im_ratio, pad=0.04)
cbar.ax.set_ylabel(value_label, rotation=270)
cbar.ax.get_yaxis().labelpad = 15
#ax.set_axis_off() # aktivizoni/çaktivizoni akset
plt.show()
```



```
import urllib.request

# shkarkoni skedarët. Asc qëndron për Ascii, një format i thjeshtë raster.
years = range(1980, 2016, 5)
base_url = 'https://raw.githubusercontent.com/endri81/instatgis/master/data/global_precipitation_1950_2017/raster'

for year in years:
    rast_url = base_url + 'precip_{}-total.asc'.format(year)
    local_file_name = 'data/global_precip_raster-{}.asc'.format(year)
    print(local_file_name)
    urllib.request.urlretrieve(rast_url, local_file_name)
    del rast_url, local_file_name
```



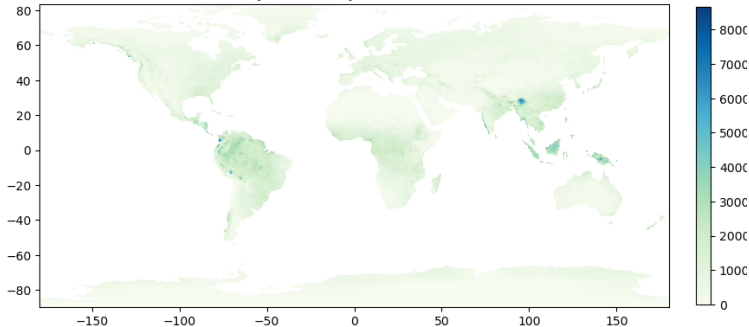
*# Për thjeshtësi, ne po i mbajmë të gjithë rastet në një sistem referimi gjeografik.
Në një studim shkencor, do të na duhej t'i projektonim ato për vizualizim.*

ndertojmë rasterat e reshjeve:

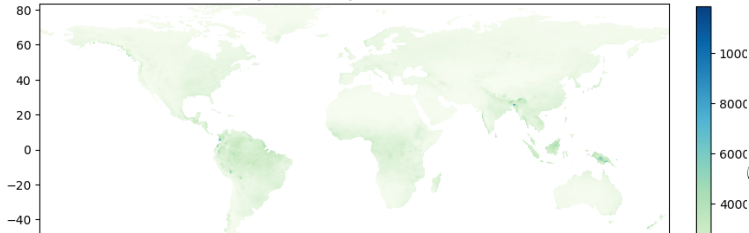
```
for year in years:
    # gjenerojme emër lokal
    local_file_name = 'data/global_precip_raster-{}.asc'.format(year)
    # open raster
    precip_rast = rasterio.open(local_file_name, mask=True)
    # plot
    plot_raster(precip_rast, precip_rast.read(1, masked=True),
                'Rreshjet totale vjetore (mm) '+str(year), 'mm',
                cmap='GnBu', width=14, height=14)
```



Rreshjet totale vjetore (mm) 1980



Rreshjet totale vjetore (mm) 1985



Endri Raço

- Duke përdorur veprimin e zbritjes nga algjebra e hartës, llogarisni dhe vizatoni diferencën e rasterit midis viteve 1980, 1990, 2000 dhe 2010 (3 gifte).
- Përdorni një cikël **for**.
- Ripërdorni funksionin **plot_raster**



```
years = [1980,1990,2000,2010]
# loop sipas viteve
for i in range(len(years)-1):
    year1 = years[i]
    year2 = years[i+1]
    print(year1,year2)
# fusnin kodin ketu
```



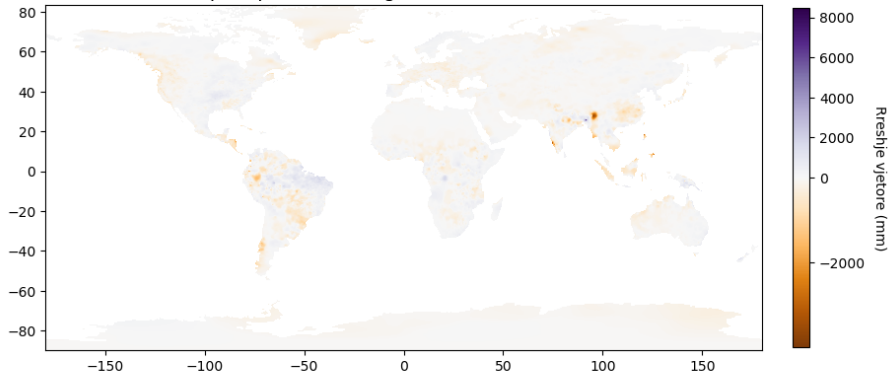

```
years = [1980,1990,2000,2010]
# loop sipas viteve
for i in range(len(years)-1):
    year1 = years[i]
    year2 = years[i+1]
    print("Krahaso:", year1, year2)
    # Ngarkojme dy rastera per krahasim
    rast1 = rasterio.open('data/global_precip_raster-{}.asc'.format(year1), mask=True)
    rast2 = rasterio.open('data/global_precip_raster-{}.asc'.format(year2), mask=True)
    vals1 = rast1.read(1, masked=True)
    vals2 = rast2.read(1, masked=True)
    # Llogarisim diferencen
    vals_diff = vals2-val1
    print("Statistikat e differences:", vals_diff.min(), round(vals_diff.mean(),2), vals_diff.max())
    # Vizatoni diferencen
    plot_raster(rast1, vals_diff, 'Total precipitation change between {} and {}'.format(year1,year2),
                'Rreshje vjetore (mm)', 'PuOr', width=10, height=10, diverge_zero=True)
```



Krahaso: 1980 1985

Statistikat e differences: -3996.5 -21.82 8455.1

Total precipitation change between 1980 and 1985



Krahaso: 1985 1990

Statistikat e differences: -2144.1 18.02 2649.0999

- Sa është sasia e reshjeve totale në çdo vend në vitin 2015?
- Duke përdorur kufijtë e botës, përdorni statistikat zonale për t'iu përgjigjur kësaj pyetjeje.
- Për çdo vend, llogarisni reshjet minimale, maksimale, mesatare, mediane dhe totale.
- Ruani rezultatet në një tabelë CSV me një rresht për çdo vend dhe një kolonë për çdo statistikë përshkruese.



- Vini re se të dhënat e reshjeve janë shumë dhe rezultatet mund të paraqesin gabime të mëdha për vendet e vogla.
- Përsëriteni analizën për vitin 1980: Duhet të bëni ndryshime minimale në kod.



```
year = 2015
output_file = 'tmp/precipitation_country_stats_' + str(year) + '.csv'
print(output_file)

# futni kodin tuaj këtu
```



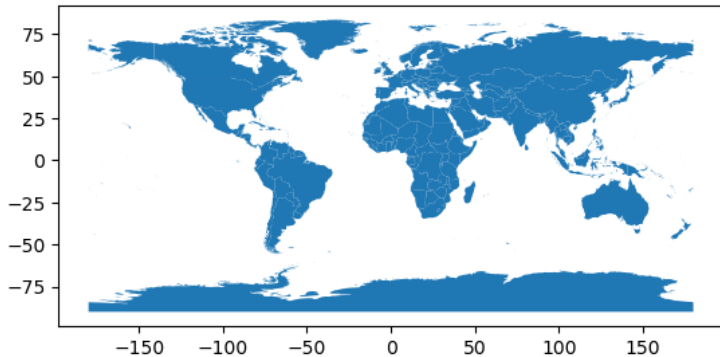
```
# shkarkojmë hartën e botës  
# Shkarkoni gjithashtu kufijtë e vendeve  
url_boundaries = 'https://github.com/endri81/instatgis/blob/master/data/gis4/natural_earth_world_boundaries_50m'  
file_name_boundaries = 'data/natural_earth_world_boundaries_50m_2018.geojson'  
urllib.request.urlretrieve(url_boundaries,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



```
# Ngarkojmë kufijtë
countries_df = gpd.read_file('data/natural_earth_world_boundaries_50m_2018.geojson')
# heqim vendet e panjohura
countries_df = countries_df[countries_df.iso_a3 != '-99']
print(len(countries_df))
countries_df.plot()
```



AXES: >




```
from rasterstats import zonal_stats, gen_zonal_stats

for year in [1980, 2015]:
    print("\nDuke llogaritur statistikat zonale midis kufijve të vendeve të botës dhe reshjeve në {}...".format(year))
    # ngarkoni rasterin e reshjeve
    rast_file = 'data/global_precip_raster-{}.asc'.format(year)
    # gjeneroni rrugën e skedarit të daljes
    output_file = 'tmp/precipitation_country_stats-{}.csv'.format(year)
    # llogarit statistikat zonale
    # zgjidhni vetëm kolonat përkatëse nga countries_df
    zon_stats = zonal_stats(countries_df[['iso_a3', 'name', 'geometry']], rast_file,
                           stats="count min median mean max sum", geojson_out=True)
    # gjeneroni një kornizë të dhënash nga rezultatet e zonal_stats (listë fjalorësh)
    stats_gdf = gpd.GeoDataFrame.from_features(zon_stats)
    # konverto nga geodataframe në dataframe, pasi nuk kemi nevojë për gjeometrinë
    stats_df = pd.DataFrame(stats_gdf.drop(columns='geometry'))
    # hiqni vendet për të cilat nuk kemi vëzhgime
    stats_df = stats_df[stats_df['count'] > 0]
    stats_df = stats_df.sort_values('sum')
    # printoni statistikat
    print(stats_df.describe())
    # ruani rezultatet në skedar
    stats_df.to_csv(output_file, index=False)
    print("Rezultatet janë të", output_file)
```



Duke llogaritur statistikat zonale midis kufijve të vendeve të botës dhe reshjeve në 1980...

	min	max	mean	count	sum \
count	184.000000	184.000000	184.000000	184.000000	1.840000e+02
mean	584.955435	2141.630439	1098.585055	462.461957	2.443332e+05
std	609.162886	1570.111066	793.979461	2072.098838	6.774382e+05
min	0.000000	65.300003	18.021053	1.000000	2.209000e+02
25%	92.725000	967.824982	520.113208	13.000000	1.358843e+04
50%	476.000000	1799.950012	911.571913	62.000000	4.707215e+04
75%	811.799988	2919.250000	1579.894516	229.750000	1.584133e+05
max	3355.699951	8657.000000	3709.935547	23984.000000	4.941679e+06

	median
count	184.000000
mean	1052.217392
std	782.798438
min	4.800000
25%	467.162506
50%	841.800018
75%	1472.725006
max	3429.199951

Rezultatet janë të tmp/precipitation_country_stats_1980.csv



- Duke përdorur **.rank()**, gjeneroni renditje për vendet për sa i përket reshjeve të tyre (renditja 1 korrespondon me vendin më të lagësht).
- Tregoni 10 vendet më të thata dhe më të lagështa në botë në vitin 1980 dhe 2015.
- A mund të vini re shumë ndryshime?



```
years = [1980, 2015]
for year in years:
    input_stats_file = 'tmp/precipitation_country_stats_' + str(year) + '.csv'
    print(input_stats_file)
    # ngarkoni skedarin dhe gjeneroni renditje
```



```
# përcaktojmë "wet rank" bazuar në precipitimin mesatar.
years = [1980,2015]

# bashkojmë statistikat në vite
wet_rank_df = countries_df[['iso_a3','name']]

for year in years:
    input_stats_file = 'tmp/precipitation_country_stats_'+str(year)+'.csv'
    print("ranking",input_stats_file)
    # ngarkojmë skedarin
    stats_df = pd.read_csv(input_stats_file)
    # gjenerojmë renditje
    rank_field = 'wet_rank_'+str(year)
    stats_df[rank_field] = stats_df['mean'].rank(ascending=False)
    # bashkojmë rankimin me rezultatet finale
    wet_rank_df = wet_rank_df.merge(stats_df[['iso_a3',rank_field]],
                                    on='iso_a3')

# sort dhe ruaj rezultate
wet_rank_df['wet_rank_change'] = wet_rank_df['wet_rank_1980']-wet_rank_df['wet_rank_2015']
wet_rank_df = wet_rank_df.sort_values('wet_rank_2015')
wet_rank_df.to_csv('tmp/precipitation_country_stats.csv', index=False)
# vendet më të lagështa 2015
wet_rank_df.head(10)
```




Ranking temp/precipitation_country_stats_2019.csv

	iso_a3	name	wet_rank_1980	wet_rank_2015	wet_rank_change
148	COM	Comoros	4.0	1.0	3.0
40	SLB	Solomon Is.	3.0	2.0	1.0
145	CRI	Costa Rica	1.0	3.0	-2.0
162	BRN	Brunei	2.0	4.0	-2.0
167	BTN	Bhutan	12.0	5.0	7.0
81	MYS	Malaysia	6.0	6.0	0.0
58	PNG	Papua New Guinea	8.0	7.0	1.0
5	VUT	Vanuatu	74.0	8.0	66.0
47	WSM	Samoa	5.0	9.0	-4.0
27	TWN	Taiwan	36.0	10.0	26.0

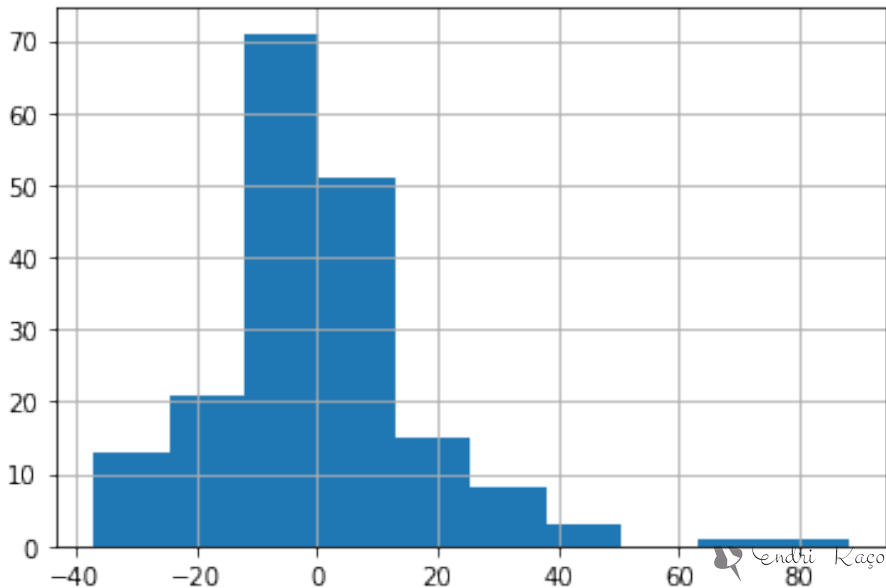
```
# vendet më të thata 2015  
wet_rank_df.tail(10)
```



	iso_a3	name	wet_rank_1980	wet_rank_2015	wet_rank_change
94	KWT	Kuwait	171.0	175.0	-4.0
2	YEM	Yemen	175.0	176.0	-1.0
137	DJI	Djibouti	179.0	177.0	2.0
46	SAU	Saudi Arabia	176.0	178.0	-2.0
180	DZA	Algeria	177.0	179.0	-2.0
73	ESH	W. Sahara	183.0	180.0	3.0
15	ARE	United Arab Emirates	174.0	181.0	-7.0
61	OMN	Oman	181.0	182.0	-1.0
87	LBY	Libya	182.0	183.0	-1.0
134	EGY	Egypt	184.0	184.0	0.0

 Endri Raco


```
# vendet me ndryshime ekstreme në klasifikim  
wet_rank_df['wet_rank_change'].hist()
```



```
# vendet me ndryshime ekstreme  
wet_rank_df = wet_rank_df.sort_values('wet_rank_change', ascending=False)  
# Kontrolllojmë diapazonin  
extreme_change_df = wet_rank_df[-wet_rank_df.wet_rank_change.between(-20, 20)]  
extreme_change_df
```



	iso_a3	name	wet_rank_1980	wet_rank_2015	wet_rank_change
60	PAK	Pakistan	161.0	73.0	88.0
5	VUT	Vanuatu	74.0	8.0	66.0
182	AFG	Afghanistan	156.0	109.0	47.0
120	GMB	Gambia	129.0	85.0	44.0
177	ARM	Armenia	149.0	111.0	38.0
57	PRY	Paraguay	81.0	48.0	33.0
35	LKA	Sri Lanka	47.0	14.0	33.0
62	NOR	Norway	116.0	83.0	33.0
45	SEN	Senegal	135.0	103.0	32.0
95	KEN	Kenya	140.0	112.0	28.0
124	NCL	New Caledonia	75.0	49.0	26.0
27	TWN	Taiwan	36.0	10.0	26.0
11	SGS	S. Geo. and the Is.	54.0	28.0	26.0
172	BGD	Bangladesh	33.0	11.0	22.0
160	BFA	Burkina Faso	110.0	89.0	21.0
91	LVA	Latvia	108.0	129.0	-21.0
21	TTO	Trinidad and Tobago	20.0	42.0	-22.0
142	CUB	Cuba	43.0	67.0	-24.0

Endri Raco

- Duke përdorur **urllib.request.urlretrieve**, shkarkoni këtë dataset që përmban aeroportet globale.
- Ngarkojeni në një dataframe gjeo me geopandas dhe printoni sa rreshta përmban.
- Zgjidhni disa attribute prej saj, duke përfshirë emrin e aeroportit, kodin IATA të aeroportit, kodin e vendit, lartësinë dhe tipin.



```
import urllib.request

# URL e azhurnuar për skedarin e aeroportit
airports_url = 'https://raw.githubusercontent.com/endri81/instatgis/master/data/gis4/airports_2020.geojson'

# Skedari lokal ku do të ruhet të dhënat
local_file_name = 'data/airports_2020.geojson'
print(local_file_name)

# Shkarkoni skedarin nga URL-ja
urllib.request.urlretrieve(airports_url, local_file_name)

# futni kodin tuaj këtu
```



```
airports_all_df = gpd.read_file('data/airports_2020.geojson')  
print('n airports =', len(airports_all_df))  
print(airports_all_df.columns)  
airports_all_df.sample(5)
```



FID	id	ident	type	name	latitude_d	longitude_d	elevation_	continent	iso_countr	...	he_latitud	he_longitu	he_elevati	he_heading	he_displac	ObjectID	type_1	descriptio	frie
2199	2200	5946.0	SBMY	medium_airport	Manicoré Airport	-5.811380	-61.278301	174.0	SA	BR	...	NaN	NaN	None	None	None	31257.0	RDO	RDO
846	847	3248.0	HSSM	medium_airport	Malakal Airport	9.558970	31.652201	1291.0	AF	SS	...	9.56544	31.6586	1291	224	None	17242.0	INFO	INFO
2940	2941	26991.0	YGEL	medium_airport	Geraldton Airport	-28.796101	114.707001	121.0	OC	AU	...	-28.78750	114.7100	120	198	None	40006.0	CTAF	CTAF
...	Larnaca


```
airports_df = airports_all_df[['id', 'iata_code', 'name', 'type', 'iso_country', 'elevation_', 'geometry']]  
# rename elevation to show it is in feet  
airports_df = airports_df.rename(columns={"elevation_": "elevation_ft"})  
# show sample  
airports_df.sample(5)
```



	id	iata_code	name	type	iso_countr	elevation_ft	geometry
1355	4219.0	LAI	Lannion-Côte de Granit Airport	medium_airport	FR	290.0	POINT Z (-3.47166 48.75440 0.00000)
876	3376.0	ALW	Walla Walla Regional Airport	medium_airport	US	1194.0	POINT Z (-118.28800 46.09490 0.00000)
2248	6048.0	ZAL	Pichoy Airport	medium_airport	CL	59.0	POINT Z (-73.08610 -39.65000 0.00000)
544	2668.0	KLR	Kalmar Airport	medium_airport	SE	17.0	POINT Z (16.28760 56.68550 0.00000)
902	3415.0	BKW	Raleigh County Memorial Airport	medium_airport	US	2504.0	POINT Z (-81.12420 37.78730 0.00000)



- Analizoni densitetin e aeroporteve në botë.
- Së pari, duke përdorur një histogram 2D me numër të ndryshëm të koshave (nga 20 në 100).
- Bëni analizën për të gjitha aeroportet dhe vetëm për aeroportet e mëdha (`type==‘large_airport’`).

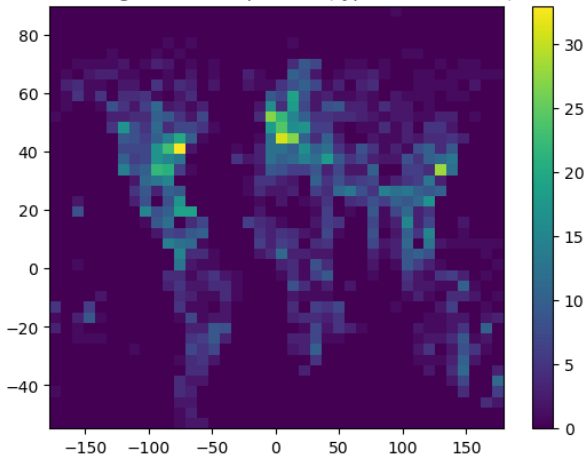


```
# ndryshojmë numrin e bin:
for bin_n in range(40,201,40):
    print("bin_n",bin_n)

# ndryshojmë tipin
for atype in ['all', 'large_airport']:
    df = airports_df
    if atype != 'all':
        df = airports_df[airports_df['type'] == atype]
    assert len(df) > 0
    # aeroportet e kërkuara janë në df
    h = plt.hist2d(df.geometry.x, df.geometry.y, bins=bin_n, density=False)
    plt.colorbar(h[3])
    plt.title("2D histograma e aeroportëve (type={}, bins={})".format(atype, bin_n))
    plt.show()
```



2D histograma e aeroporteve (type=all, bins=40)



2D histograma e aeroporteve (type=large_airport, bins=40)



- Gjeneroni KDE për aeroportet për Britaninë dhe SHBA-në duke ndryshuar gjerësinë e brezit në tre vlera të ndryshme që kapin shpërndarjen e aeroporteve në një mënyrë të përshtatshme.
- Ku janë zonat më të dendura në botë? Ndani analizën midis të gjitha aeroporteve dhe vetëm aeroporteve të mëdha, duke minimizuar përsëritjen e kodit.



```
import geoplot
import matplotlib.pyplot as plt

# Për secilin vend, 'US' dhe 'GB'
for country in ['US', 'GB']:
    # zgjidh aeroportet në adf
    adf = airports_df[airports_df['iso_countr'] == country]
    cdf = countries_df[countries_df['iso_a2'] == country]

    # nëse vendi është 'GB', largoni një aeroport RAF në Qipro
    if country == 'GB':
        adf = adf[adf['iata_code'] != 'AKT']

    # sigurohuni që të dhënat e vendit dhe aeroportit të mos jenë bosh
    assert len(cdf) > 0
    assert len(adf) > 0

    # ndryshoni bandwidth
    for bandwidth in [.01, .05, .1]:
        title = 'KDE për aeroportet (vendi={}, bandwidth={})'.format(country, bandwidth)
        ax = geoplot.kdeplot(adf, shade=False, bw=bandwidth, figsize=(12, 12), alpha=.5)

        # shtoni vijën bregdetare
        cdf.plot(ax=ax, color='lightgray', edgecolor="none", linewidth=.5)

        # shtoni titullin
        plt.title(title, fontsize=18)

    # shfaqni figurën
    plt.show()
```

KDE për aeroportet (vendi=US, bandwidth=0.01)

