МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Практикум №4

з курсу «Аналіз даних в інформаційнних системах»

на тему: «Вивідна статистика»

Викладач:
Ліхоузова Т.А.

Виконав:
студент 2 курсу
групи ІП-15 ФІОТ
Мєшков Андрій
Ігорович

Київ-2023

Практикум №4

Вивідна статистика

Мета роботи:

- ознайомитись з методами визначення точкових оцінок параметрів розподілу; дослідити, що впливає на якість точкових оцінок;
- методикою визначення інтервальних оцінок параметрів розподілу; дослідити, що впливає на якість інтервальних оцінок; методами перевірки статистичних гіпотез про вигляд закону розподілу;
- дослідити, що впливає на ширину критичної області.

Завдання:

Скачати потрібні дані.

Основне завдання

Скачати дані файлу Data2.csv.

1. Подивитись, проаналізувати структуру
2. Вказати, чи є параметри, що розподілені за нормальним законом
3. Перевірити гіпотезу про рівність середнього і медіани для одного з параметрів
4. Вказати, в якому регіоні розподіл викидів CO2 найбільш близький до нормального
5. Побудувати кругову діаграму населення по регіонам

Додаткове завдання

Завдання 1

1. Завантажити карту України Ukraine.jpg
2. Розмістити бульбашки, що відповідають їх населенню, на довільних 5 містах (статистику взяти в інтернеті)
3. Знайти найбільшу відстань між містами в пікселях та кілометрах

Завдання 2

1. Завантажити файл з даними про конфлікти conflicts.csv
2. Побудувати просторовий розподіл конфліктів у Європі та Україні
3. Для Європи побудувати ізокліни для розподілу конфліктів
4. Для України:
    1. визначити, який регіон представлено на реальній карті;
    2. класифікувати точки по рокам;
    3. побудувати розподіл щільності.

Хід роботи:

Основне завдання:

Імпортуємо потрібні бібліотеки.

```
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as stats
```

Напишемо шлях до файлу.

```
data_path = 'Data2.csv'
```

Зчитаємо файл.

```
def read_dataset(path):
    df = pd.read_csv(path, sep=';', encoding='cp1252')
    return df
```

Виправимо дані, використовуючи функції з попередньої роботи.

```
def remove_typo(df):
    df.rename(columns={"Populatiion": "Population"}, inplace=True)
    return df
def clean_up(df):
    df['Area'] = df['Area'].str.replace(',', '.').astype(float)
    df["GDP per capita"] = df["GDP per capita"].str.replace(',', '.').astype(float)
    df["CO2 emission"] = df["CO2 emission"].str.replace(',', '.').astype(float)
    return df
def fix_negative(df):
    fix_gdp = df[df['GDP per capita'] < 0]
    area_gdp = df[df['Area'] < 0]
    fix_gdp['GDP per capita'] *= -1
    area_gdp['Area'] *= -1
    df[df['GDP per capita'] < 0] = fix_gdp
    df[df['Area'] < 0] = area_gdp
    return df
def fix_NaN(df):
    df = df.fillna(df.mean())
    return df
```

Проаналізуємо структуру.

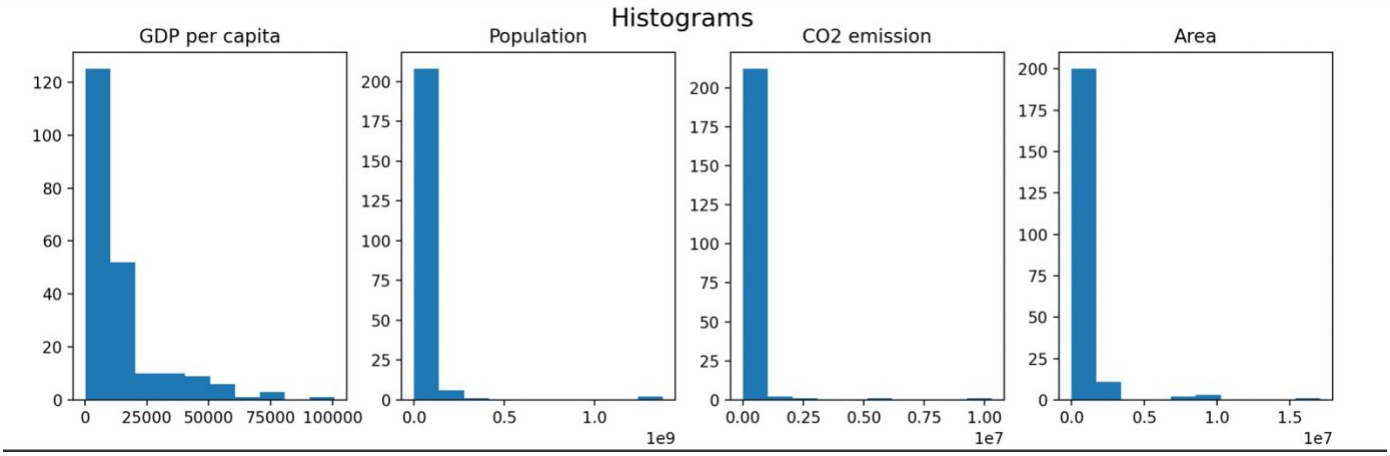
```
def print_exploring(df):
    print('Data frame info:')
    df.info()

    pd.set_option("display.max_columns", None)
    print('\nFirst 5 rows:')
    print(df.head())

    print('\nDescriptive statistics of the dataframe:')
    print(df.describe())
```

```
Data frame info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 217 entries, 0 to 216
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Country Name    217 non-null    object
 1   Region          217 non-null    object
 2   GDP per capita  217 non-null    float64
 3   Population      217 non-null    float64
 4   CO2 emission    217 non-null    float64
 5   Area            217 non-null    float64
dtypes: float64(4), object(2)
memory usage: 10.3+ KB

First 5 rows:
     Country Name                     Region  GDP per capita  Population  \
0     Afghanistan                 South Asia      561.778746  34656032.0
1         Albania      Europe & Central Asia     4124.982390   2876101.0
2         Algeria  Middle East & North Africa     3916.881571  40606052.0
3  American Samoa       East Asia & Pacific    11834.745230     55599.0
4         Andorra      Europe & Central Asia    36988.622030     77281.0

    CO2 emission       Area
0    9809.225000   652860.0
1    5716.853000    28750.0
2  145400.217000  2381740.0
3  165114.116337      200.0
4     462.042000      470.0

Descriptive statistics of the dataframe:
       GDP per capita    Population  CO2 emission          Area
count      217.000000  2.170000e+02  2.170000e+02  2.170000e+02
mean     13445.593416  3.432256e+07  1.651141e+05  6.188441e+05
std      16873.922101  1.344477e+08  8.100511e+05  1.827830e+06
min        285.727442  1.109700e+04  1.100100e+01  2.000000e+00
25%       2361.160205  7.956010e+05  1.954511e+03  1.088700e+04
50%       7179.340661  6.293253e+06  1.156205e+04  9.303000e+04
75%      14428.140260  2.369592e+07  8.256251e+04  4.474200e+05
max     100738.684200  1.378665e+09  1.029193e+07  1.709825e+07
```

Перевіремо, чи є параметри, що розподілені за нормальним законом.

```
def normally_visual_test(df):
    fig, axs = plt.subplots(1, 4, figsize=(16, 4))
    fig.suptitle('Histograms', fontsize=16)
    axs[0].set_title('GDP per capita')
    axs[0].hist(df['GDP per capita'])
    axs[1].set_title('Population')
    axs[1].hist(df['Population'])
    axs[2].set_title('CO2 emission')
    axs[2].hist(df['CO2 emission'])
    axs[3].set_title('Area')
    axs[3].hist(df['Area'])
```

```
    plt.show()
```



Проведемо тести за критеріями.

```
column = ['GDP per capita', 'Population', 'CO2 emission', 'Area']
def shapiro_test(df, columns=0, alpha=0.05):
    print("\nShapiro–Wilk test:")
    if columns==0:
        data = df
        columns = [1]
    for column in columns:
        if column != 1:
            data = df[column]
        stat, p = stats.shapiro(data)
        print('Statistics=%.3f, p=%.3f' % (stat, p))
        if p > alpha:
            print('The data correspond to a normal distribution')
        else:
            print('The data do not correspond to a normal distribution')
def ks_test(df, columns=0, alpha=0.05):
    print("\nKolmogorov–Smirnov test:")
    if columns==0:
        data = df
        columns = [1]
    for column in columns:
        if column != 1:
            data = df[column]
        stat, p = stats.kstest(data, 'norm')
        print('Statistics=%.3f, p=%.3f' % (stat, p))
        if p > alpha:
            print('The data correspond to a normal distribution')
        else:
            print('The data do not correspond to a normal distribution')

def dagostino_test(df, columns=0, alpha=0.05):
    print("\nD'Agostino's test:")
    if columns==0:
        data = df
        columns = [1]
    for column in columns:
        if column != 1:
```

```
    data = df[column]
  stat, p = stats.normaltest(data)
  print('Statistics=%.3f, p=%.3f' % (stat, p))
  if p > alpha:
    print('The data correspond to a normal distribution')
  else:
    print('The data do not correspond to a normal distribution')
```

```
Shapiro-Wilk test:
Statistics=0.731, p=0.000
The data do not correspond to a normal distribution
Statistics=0.217, p=0.000
The data do not correspond to a normal distribution
Statistics=0.174, p=0.000
The data do not correspond to a normal distribution
Statistics=0.338, p=0.000
The data do not correspond to a normal distribution

Kolmogorov-Smirnov test:
Statistics=1.000, p=0.000
The data do not correspond to a normal distribution
Statistics=1.000, p=0.000
The data do not correspond to a normal distribution
Statistics=1.000, p=0.000
The data do not correspond to a normal distribution
Statistics=0.995, p=0.000
The data do not correspond to a normal distribution

D'Agostino's test:
Statistics=110.278, p=0.000
The data do not correspond to a normal distribution
Statistics=370.214, p=0.000
The data do not correspond to a normal distribution
Statistics=406.218, p=0.000
The data do not correspond to a normal distribution
Statistics=284.697, p=0.000
The data do not correspond to a normal distribution
```

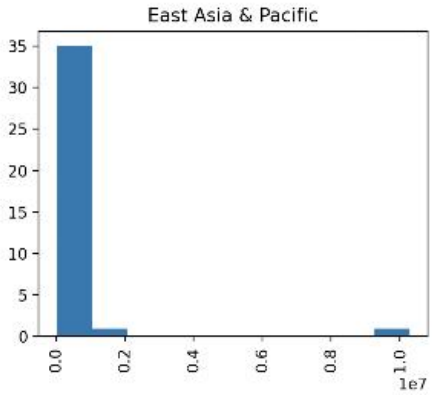
Перевіримо гіпотезу про рівність середнього і медіани для одного з параметрів

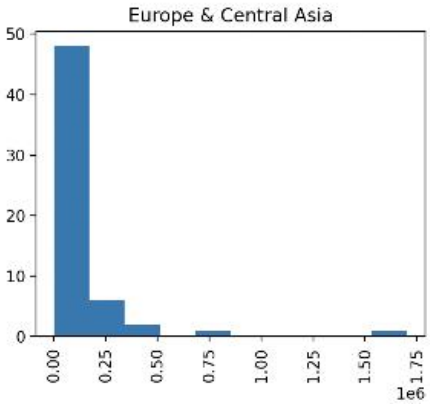
```
def mean_median(df, columns):
  for column in columns:
    print(f'\n{column}:')
    mean_gdp = df[column].mean()
    median_gdp = df[column].median()
    print(f'Mean {column}', ' - ' ,mean_gdp)
    print(f'Median {column} ',  ' - ' , median_gdp, '\n')
    if mean_gdp == median_gdp:
      print(f"The mean and median {column} are the same: {mean_gdp}")
```

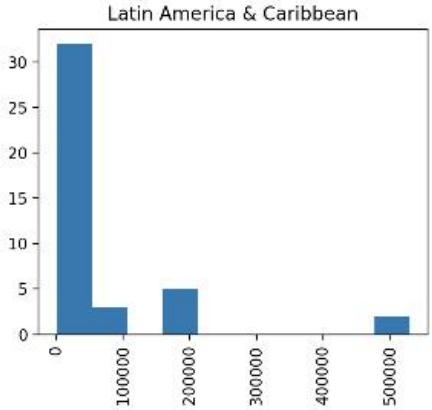
```
GDP per capita:
Mean GDP per capita  -  13445.593416057369
Median GDP per capita   -  7179.340661


Population:
Mean Population  -  34322559.875
Median Population   -  6293253.0


CO2 emission:
Mean CO2 emission  -  165114.1163365854
Median CO2 emission   -  11562.051


Area:
Mean Area  -  618844.1023041474
Median Area   -  93030.0
```

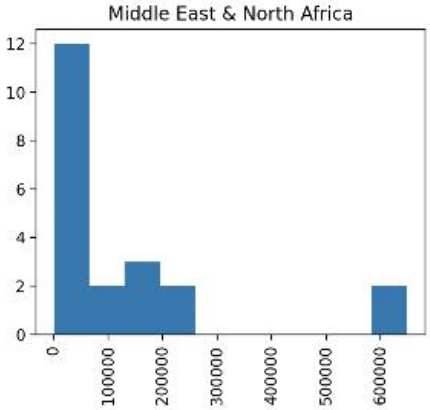
Перевіримо, в якому регіоні розподіл викидів CO2 найбільш близький до нормального

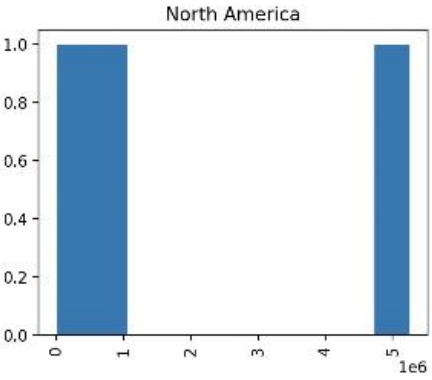
```
def closest_co2(df):
    df['CO2 emission'].hist(by=df['Region'], layout=(4, 2), figsize=(10, 20))
    plt.show()
    for region in df['Region'].unique():
        region_emissions = df[df['Region'] == region]['CO2 emission']
        print(f'\nCheck for the region {region}:')
        try:
            shapiro_test(region_emissions)
        except ValueError as e:
            print(str(e))
        try:
            ks_test(region_emissions)
        except ValueError as e:
            print(str(e))
        try:
            dagostino_test(region_emissions)
        except ValueError as e:
            print(str(e))
```

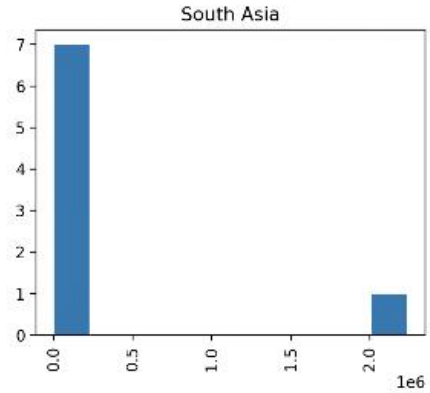

East Asia & Pacific
35
30
25
20
15
10
5
0
0.0
0.2
0.4
0.6
0.8
1.0
1e7

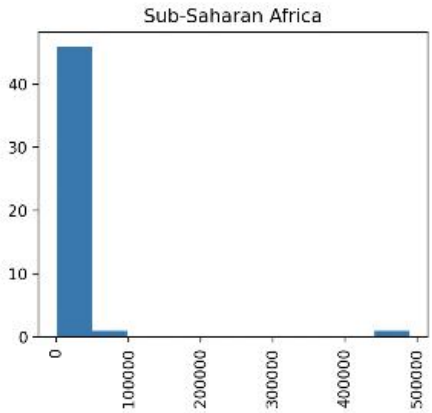

Europe & Central Asia
50
40
30
20
10
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
1e6


Latin America & Caribbean
30
25
20
15
10
5
0
0
100000
200000
300000
400000
500000


Middle East & North Africa
12
10
8
6
4
2
0
0
100000
200000
300000
400000
500000
600000


North America
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
1e6


South Asia
7
6
5
4
3
2
1
0
0.0
0.5
1.0
1.5
2.0
1e6

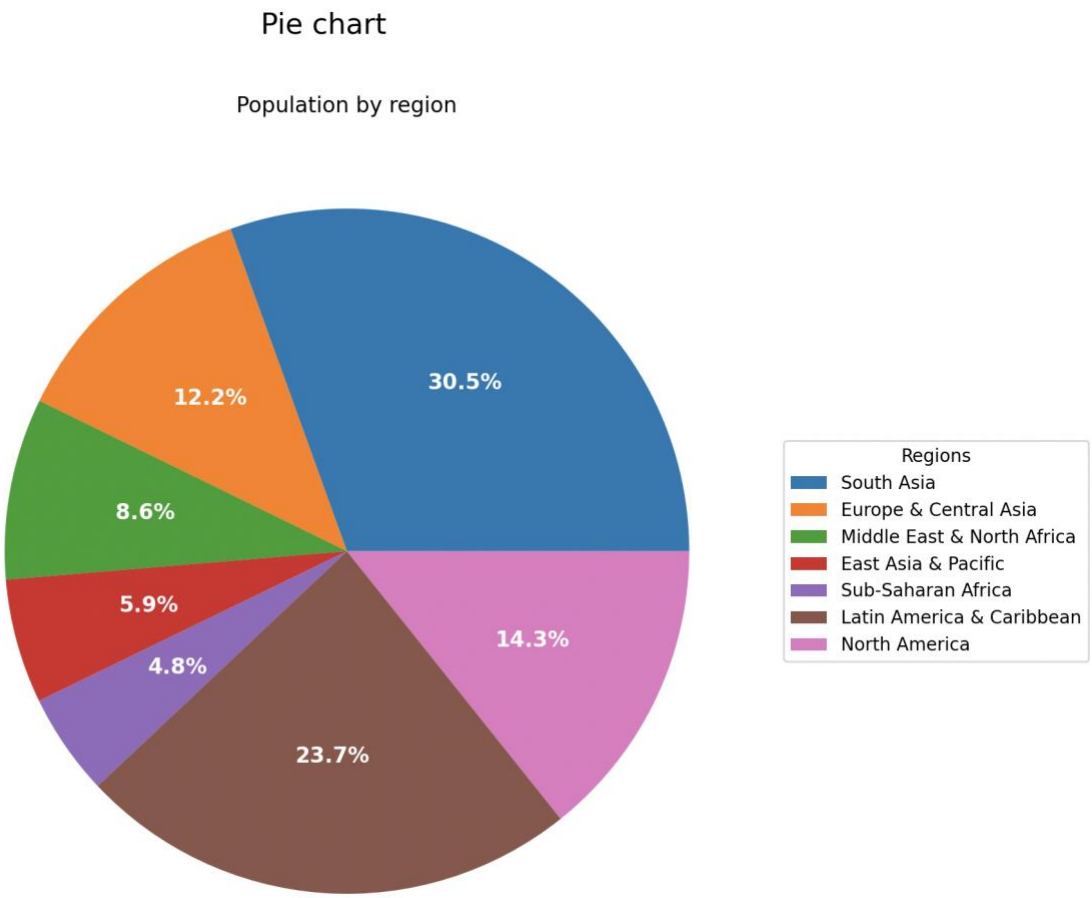

Sub-Saharan Africa
40
30
20
10
0
0
100000
200000
300000
400000
500000

```
Check for the region South Asia:

Shapiro-Wilk test:
Statistics=0.473, p=0.000
The data do not correspond to a normal distribution

Kolmogorov-Smirnov test:
Statistics=1.000, p=0.000
The data do not correspond to a normal distribution

D'Agostino's test:
/Users/andrey/Library/Python/3.9/lib/python/site-packa
  warnings.warn("kurtosistest only valid for n>=20 ...
Statistics=22.551, p=0.000
The data do not correspond to a normal distribution

Check for the region Europe & Central Asia:

Shapiro-Wilk test:
Statistics=0.470, p=0.000
The data do not correspond to a normal distribution

Kolmogorov-Smirnov test:
Statistics=1.000, p=0.000
The data do not correspond to a normal distribution

D'Agostino's test:
Statistics=95.175, p=0.000
The data do not correspond to a normal distribution

Check for the region Middle East & North Africa:

Shapiro-Wilk test:
Statistics=0.664, p=0.000
The data do not correspond to a normal distribution

Kolmogorov-Smirnov test:
Statistics=1.000, p=0.000
The data do not correspond to a normal distribution

D'Agostino's test:
Statistics=22.295, p=0.000
The data do not correspond to a normal distribution
```

```
Check for the region East Asia & Pacific:

Shapiro-Wilk test:
Statistics=0.229, p=0.000
The data do not correspond to a normal distribution

Kolmogorov-Smirnov test:
Statistics=1.000, p=0.000
The data do not correspond to a normal distribution

D'Agostino's test:
Statistics=84.483, p=0.000
The data do not correspond to a normal distribution

Check for the region Sub-Saharan Africa:

Shapiro-Wilk test:
Statistics=0.215, p=0.000
The data do not correspond to a normal distribution

Kolmogorov-Smirnov test:
Statistics=1.000, p=0.000
The data do not correspond to a normal distribution

D'Agostino's test:
Statistics=104.031, p=0.000
The data do not correspond to a normal distribution

Check for the region Latin America & Caribbean:

Shapiro-Wilk test:
Statistics=0.540, p=0.000
The data do not correspond to a normal distribution

Kolmogorov-Smirnov test:
Statistics=1.000, p=0.000
The data do not correspond to a normal distribution

D'Agostino's test:
Statistics=47.312, p=0.000
The data do not correspond to a normal distribution

Check for the region North America:

Shapiro-Wilk test:
Statistics=0.826, p=0.178
The data correspond to a normal distribution

Kolmogorov-Smirnov test:
Statistics=1.000, p=0.000
The data do not correspond to a normal distribution

D'Agostino's test:
skewtest is not valid with less than 8 samples; 3 samples were given.
```

Побудуємо кругову діаграму населення по регіонам

```
def pie_chart(df):
    fig, ax = plt.subplots(figsize=(8, 8))
    fig.suptitle('Pie chart', fontsize=16)
    labels = pd.unique(df['Region'])
    wedges, texts, autotexts = ax.pie(df.groupby('Region').sum()['Population'], labels=labels,
                                      autopct='%1.1f%%', textprops=dict(color='w'))
    ax.set_title('Population by region')
    ax.legend(wedges, labels,
          title='Regions',
          loc='center left',
          bbox_to_anchor=(1, 0, 0, 1))
    plt.setp(autotexts, size=12, weight='bold')
    plt.show()
```



Додаткове завдання:

Завдання 1

Імпортуємо потрібні бібліотеки.

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from scipy.spatial import distance
```

Завантажемо карту.

```
img_path = 'Ukraine.jpg'
def map_downloading(img_path):
    map_img = mpimg.imread(img_path)
    return map_img
```

Розмістити бульбашки, що відповідають їх населенню, на довільних 5 містах

```
cities = ['Київ', 'Краматорськ', 'Харків', 'Львів', 'Миколаїв']
cities_coords = np.array([(387, 146), (715, 256), (647, 176), (91, 188), (452, 381)])
cities_population = np.array([2884, 185, 1419, 721, 486])
def bubles(map_img, coords, population):
    fig, ax = plt.subplots(figsize=(15, 15))
    fig.suptitle('Ukraine', fontsize=16)
    ax.imshow(map_img)
    ax.scatter(
        coords[:, 0],
        coords[:, 1],
        s=population * 2,
        c='green',
        alpha=0.5,
        linewidth=2
    )
    ax.axis('off')
    plt.show()
```

Знайдемо найбільшу відстань між містами в пікселях та кілометрах

```
def greatest_distance(map_img, cities, coords):
    distances = distance.cdist(coords, coords, 'euclidean')
    city_A, city_B = np.unravel_index(distances.argmax(), distances.shape)
    pixel_distance = distances[city_A, city_B]
    ukraine_width_km = 1316
    km_per_pixel = ukraine_width_km / map_img.shape[1]
    km_distance = distances[city_A, city_B] * km_per_pixel
    print(f'Найбільша відстань - між містами {cities[city_A]} та {cities[city_B]}.')
    print(f'Відстань у пікселях: {pixel_distance:.2f} пікселів')
    print(f'Відстань у кілометрах: {km_distance:.2f} км')
```

```
Найбільша відстань — між містами Краматорськ та Львів.
Відстань у пікселях: 627.69 пікселів
Відстань у кілометрах: 994.04 км
```

Завдання 2

Імпортуємо потрібні бібліотеки.

```
import geoviews as gv
from geoviews import dim
import pandas as pd
from geoviews.tile_sources import CartoDark
from geoviews.tile_sources import StamenTerrain
gv.extension('bokeh', 'matplotlib')
```

Зчитаємо файл.

```
data_path = 'conflicts.csv'
def read_dataset(path):
    df = pd.read_csv(path, sep=';', encoding='cp1252', decimal=',')
    print("Data:")
    print(df)
    return df
```

```
Data:
          id  year  type_of_violence                  conflict_name  \
0      64991  2011                 1                     Spain:Basque
1      66561  2011                 1  United Kingdom:Northern Ireland
2     156154  2011                 1      Azerbaijan:Nagorno-Karabakh
3     156157  2011                 1      Azerbaijan:Nagorno-Karabakh
4     156162  2011                 1      Azerbaijan:Nagorno-Karabakh
...      ...   ...               ...                              ...
1588  236744  2016                 1              Ukraine:Novorossiya
1589  236745  2016                 1              Ukraine:Novorossiya
1590  236746  2016                 1              Ukraine:Novorossiya
1591  236747  2016                 1              Ukraine:Novorossiya
1592  236748  2016                 1              Ukraine:Novorossiya

       latitude  longitude        country  region  date_start    date_end
0     43.260919  -2.938764          Spain  Europe  2011-11-07  2011-11-07
1     54.600000  -7.300000  United Kingdom  Europe  2011-04-02  2011-04-02
2     40.000000  47.000000     Azerbaijan  Europe  2011-01-17  2011-01-17
3     40.000000  47.000000     Azerbaijan  Europe  2011-01-20  2011-01-20
4     40.000000  47.000000     Azerbaijan  Europe  2011-01-25  2011-01-25
...         ...        ...            ...     ...         ...         ...
1588  48.920000  39.020000        Ukraine  Europe  2016-10-27  2016-10-28
1589  48.920000  39.020000        Ukraine  Europe  2016-08-27  2016-08-28
1590  48.140000  37.740000        Ukraine  Europe  2016-05-23  2016-05-24
1591  48.920000  39.020000        Ukraine  Europe  2016-04-29  2016-04-30
...

[1593 rows x 10 columns]
```

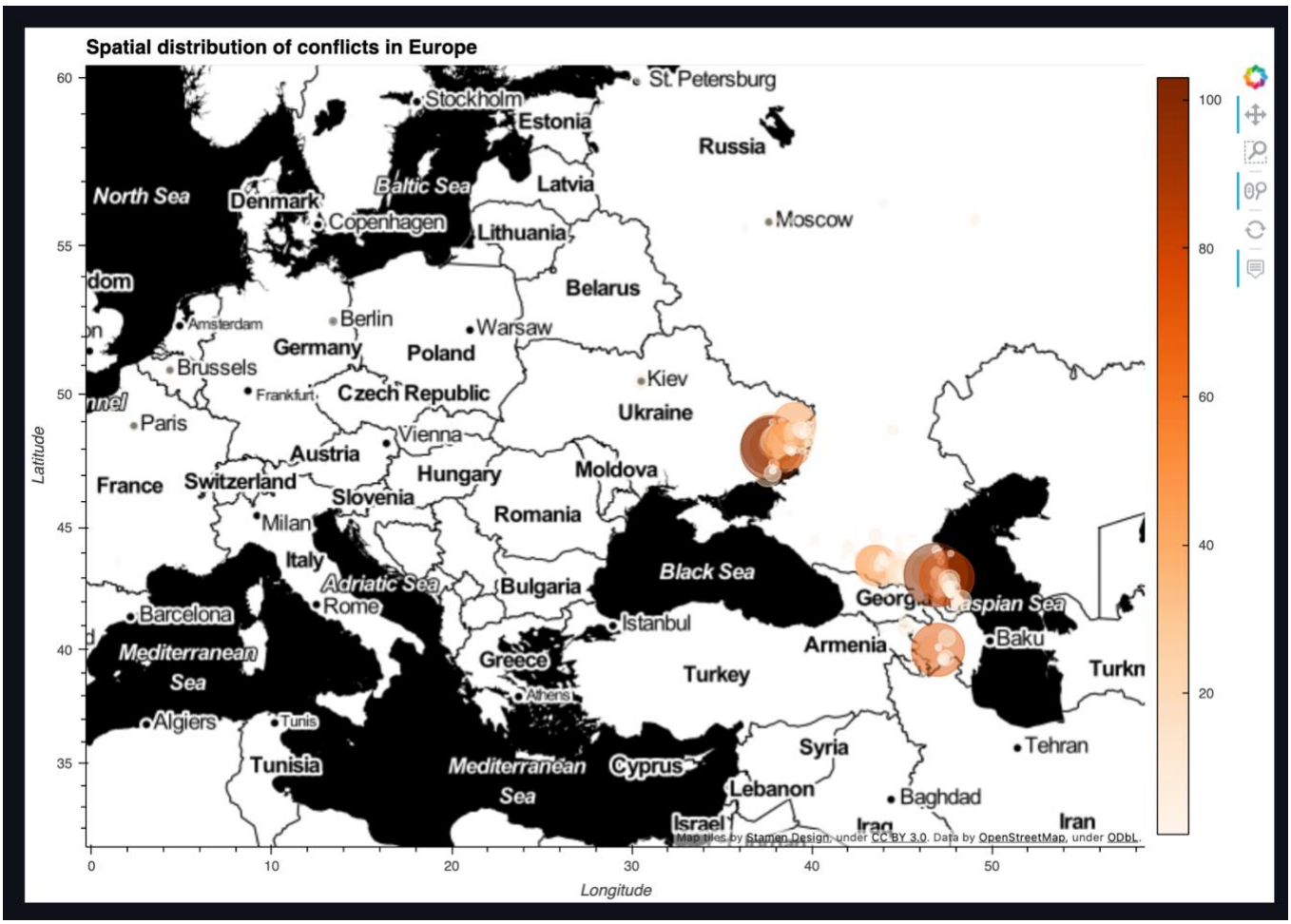
Згрупуємо дані

```
def grouping(conflicts):
    grouped_conflicts = conflicts.groupby(['country', 'longitude',
'latitude']).size().to_frame('quantity').reset_index()
    print("\nGrouped data:")
    grouped_conflicts
    print(grouped_conflicts)
    return grouped_conflicts
```

```
Grouped data:
            country  longitude   latitude  quantity
0           Armenia  43.847500  40.789444         1
1           Armenia  44.999670  41.205030         1
2           Armenia  45.056580  41.109250         1
3           Armenia  45.063810  41.120810         1
4           Armenia  45.166667  40.916667         6
..              ...        ...        ...       ...
367         Ukraine  39.751691  47.848613         1
368         Ukraine  39.775500  48.215000         1
369         Ukraine  39.822660  48.503600         1
370         Ukraine  39.845000  48.850000         2
371  United Kingdom  -7.300000  54.600000         1

[372 rows x 4 columns]
```

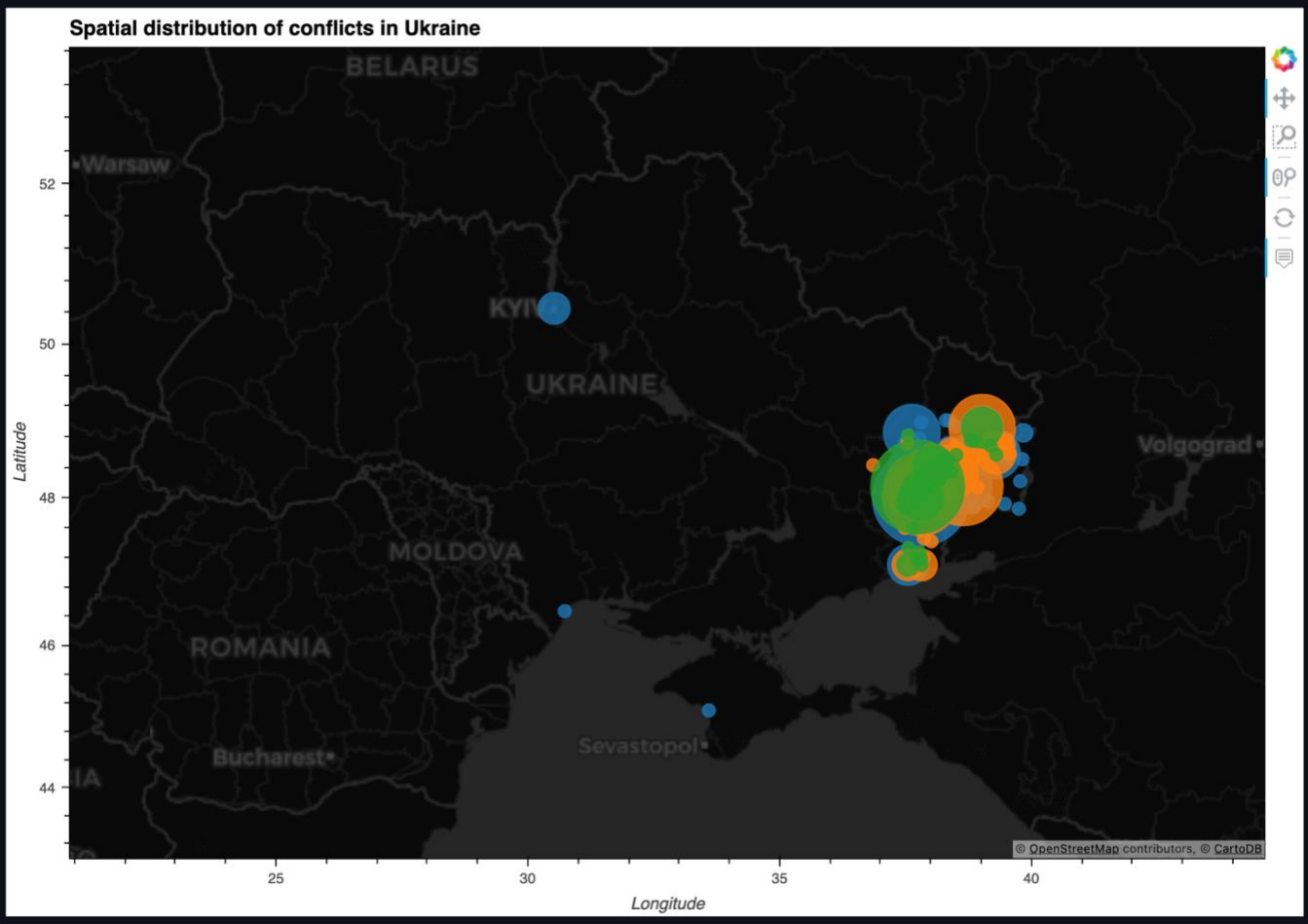
Побудуємо просторовий розподіл Європи

```
def visualization_europe( g_conflicts):
    # Europe
    points = gv.Points(g_conflicts, ['longitude', 'latitude'])
    tiles = gv.tile_sources. StamenToner
    gv.output(tiles * points.opts(
        title='Spatial distribution of conflicts in Europe',
        color='quantity', size=dim('quantity') ** (1/2) * 5,
        cmap='Oranges', tools=['hover'], width=1000, height=700,
        show_legend=False, alpha=0.5, colorbar=True
    ))
```




```
def visualization_ukraine(conflicts):
    # Ukraine
    ukr_conflicts = conflicts[conflicts['country'] == 'Ukraine']
    ukr_grouped_conflicts = ukr_conflicts.groupby(['year', 'longitude',
'latitude']).size().to_frame('quantity').reset_index()
    print(ukr_grouped_conflicts)

    ukr_grouped_conflicts['year'] = ukr_grouped_conflicts['year'].apply(str)
    points = gv.Points(ukr_grouped_conflicts, ['longitude', 'latitude'])
    tiles = gv.tile_sources.CartoDark

    gv.output(tiles * points.opts(
    title='Spatial distribution of conflicts in Ukraine',
    color='year', size=dim('quantity') ** (1/2) * 10,
    cmap='Category10', tools=['hover'], width=1000, height=700,
    show_legend=False, alpha=0.8
    ))
    # Number of conflicts in Ukraine by year
    print('\nNumber of conflicts in Ukraine by year:')
    print(ukr_conflicts.groupby(['year']).size())
```

```
     year  longitude   latitude  quantity
0    2014  30.523330  50.450000         6
1    2014  30.733333  46.466667         1
2    2014  33.595000  45.093056         1
3    2014  37.349593  48.462882         2
4    2014  37.482500  47.595000         1
..    ...        ...        ...       ...
136  2016  38.800000  48.750000         1
137  2016  38.870126  48.746225         1
138  2016  39.020000  48.920000        10
139  2016  39.190347  48.681487         1
140  2016  39.300000  48.566667         1

[141 rows x 4 columns]
```




```
Number of conflicts in Ukraine by year:
year
2014    269
2015    252
2016    123
dtype: int64
```

Висновок

За отриманими даними можна зробити висновок, що

- розподіл викидів CO2 найбільш близький до нормального в Північній Америці
- більшість конфліктів в Європі в проміжку 2011-2016 були в Україні.