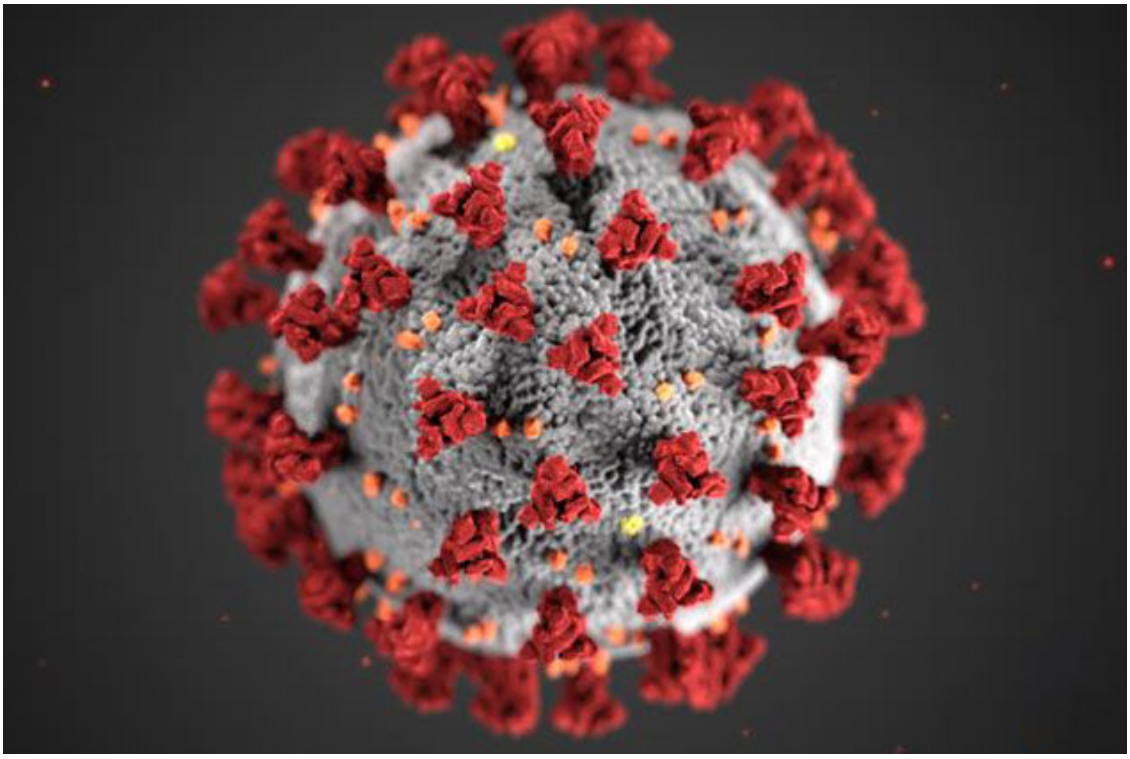
Analiza danych COVID-19

T. P, T. Ś

30.04.2020

[1]:
```
from IPython.display import Image
Image(filename='1.jpg', width=500)
```

[1]:



Koronawirus 2019-nCoV to wirus należący do rodziny koronawirusów. Koronawirusy występują u zwierząt i powodują u nich różne choroby (układu oddechowego, układu pokarmowego, wątroby, układu nerwowego), wiele zakażeń przebiega też bezobjawowo. Wirusy te często mutują i mają dużą zdolność do zakażania nowych gatunków. Wszystkie poznane dotąd koronawirusy powodujące zakażenia u ludzi są wirusami, które wywołują objawy ze strony układu oddechowego, bardzo rzadko ze strony innych układów i narządów. Możliwe, że oprócz zakażenia układu oddechowego u dzieci do dwunastego miesiąca życia mogą wywoływać biegunkę. Do 2019 roku poznano sześć wirusów powodujących zakażenia u ludzi. Cztery z nich (229E, OC43, NL63, HKU1) są przyczyną przeziębienia o łagodnym przebiegu. Dwa pozostałe (wirusy SARS i MERS) mogą prowadzić do zagrażającej życiu ostrej niewydolności oddechowej. Koronawirus

2019-nCoV jest wirusem odpowiedzialnym za obecną epidemię zakażeń układu oddechowego, która rozpoczęła się w Wuhan, w Chinach i tam po raz pierwszy został zidentyfikowany w grudniu 2019 roku. COVID-19 to choroba wywołana przez koronawirus z Wuhan (2019-CoV). Wcześniej choroba nie miała nazwy i posługiwano się określeniami typu „choroba wywołana przez koronawirus" lub „zachorowanie z powodu zakażenia koronawirusem z Wuhan". Nazwa COVID-19 została ogłoszona przez Światową Organizację Zdrowia (WHO) i obowiązuje oficjalnie. "CO" w nazwie oznacza koronę (ang. corona), "VI" – wirus (ang. virus), "D" - chorobę (ang. disease), a liczba 19 wskazuje rok pojawienia się wirusa – 2019 (Corona-Virus-Disease-2019). Różnica między koronawirusem a COVID-19 jest taka, że koronawirus to czynnik wywołujący chorobę, a COVID-19 to choroba, czyli zespół objawów spowodowanych przez ten czynnik.

Źródło: https://www.mp.pl

Analizowane dane dotyczące COVID-19 pochodzą ze strony Johns Hopkins University. Dane są aktualizowane, tak więc każdorazowe uruchomienie notatnika będzie przedstawiało analizę dla aktualnych danych. W opracowaniu wykorzystane zostały dane (dla danego kraju lub regionu) dotyczące liczby potwierdzonych przypadków (Liczba_potwierdzonych_przypadków) COVID-19 w tym liczby zgonów (Liczba_zgonów), wyzdrowień (Liczba_wyzdrowień) i wciąż chorych osób (Liczba_chorych). Analizie poddany został również współczynnik śmiertelności (Współczynnik_śmiertelności) określony jako procentowy stosunek liczby zgonów do liczby potwierdzonych przypadków zakażenia. W niniejszej analizie wyznaczone zostały podstawowe wielkości statystyczne badanych zmiennych, przedstawione zostały wizualizacje porównujące wielkość danej zmiennej dla różnego kraju lub regionu (na wykresach przedstawiono wartości zmiennych dla 15 przypadków w których zmienne te przyjmują największe wartości) oraz wykonana została klasteryzacja danych.

[2]:
```
# Wczytywanie potrzebnych modułów

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import KMeans
import scipy as sp
from statsmodels.distributions.empirical_distribution import ECDF
```

[3]:
```
# Wczytanie analizowanych danych (dane są aktualizowne więc każdorazowe␣
 ↪wczytanie da aktualne wyniki)

url = "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/web-data/data/
 ↪cases_country.csv"

dane = pd.read_csv(url)
```

[4]:
```
# Preparacja danych - usunięte zostały dane które w większości przypadków␣
 ↪zawierały braki np. People_Hospitalized lub takie, które nie wnosiły
# nic do analizy np. ISO3 (kodowe nazwy regionów)

dane = dane.
 ↪drop(["Lat","Long_","People_Hospitalized","People_Tested","UID","ISO3","Incident_Rate"],axis␣
 ↪= 1)
dane.columns =␣
 ↪['Region','Ostatnia_aktualizacja','Liczba_potwierdzonych_przypadków','Liczba_zgonów','Liczba_
dane.head(10)
```

[4]:

	Region	Ostatnia_aktualizacja	Liczba_potwierdzonych_przypadków	Liczba_zgonów	Liczba_wyzdrowień	Liczba_chorych	Współczynnik_śmiertelności
0	Australia	2020-04-29 22:32:31	6746	90	5688	968	1.334124
1	Austria	2020-04-29 22:32:31	15402	580	12779	2043	3.765745
2	Canada	2020-04-29 22:32:31	52831	3152	20253	29426	5.966194
3	China	2020-04-29 22:32:31	83940	4637	78455	848	5.524184
4	Denmark	2020-04-29 22:32:31	9206	443	6558	2205	4.812079
5	Finland	2020-04-29 22:32:31	4906	206	2800	1900	4.198940
6	France	2020-04-29 22:32:31	166541	24121	49117	93303	14.483521
7	Germany	2020-04-29 22:32:31	161197	6405	120400	34392	3.973399
8	Iceland	2020-04-29 22:32:31	1797	10	1656	131	0.556483
9	Ireland	2020-04-29 22:32:31	20253	1190	13386	5677	5.875673

[5]:
```
# Sprawdzanie ilości braków

print('Suma braków w danej kolumnie:\n')
dane.isnull().sum()
```

```
Suma braków w danej kolumnie:
```

[5]:
```
Region                                0
Ostatnia_aktualizacja                 0
Liczba_potwierdzonych_przypadków      0
Liczba_zgonów                         0
Liczba_wyzdrowień                     0
Liczba_chorych                        0
Współczynnik_śmiertelności            0
```

[6]:
```
# Podstawowe parametry statystyczne

round(dane.drop(['Region','Ostatnia_aktualizacja'],axis = 1).describe(),2)
```

[6]:

	Liczba_potwierdzonych_przypadków	Liczba_zgonów	Liczba_wyzdrowień	\
count	185.00	185.00	185.00	
mean	17227.19	1227.98	5251.20	
std	82816.36	5848.03	18940.75	
min	1.00	0.00	0.00	
25%	95.00	2.00	21.00	
50%	738.00	15.00	252.00	
75%	6200.00	157.00	1288.00	
max	1037526.00	60846.00	132929.00	

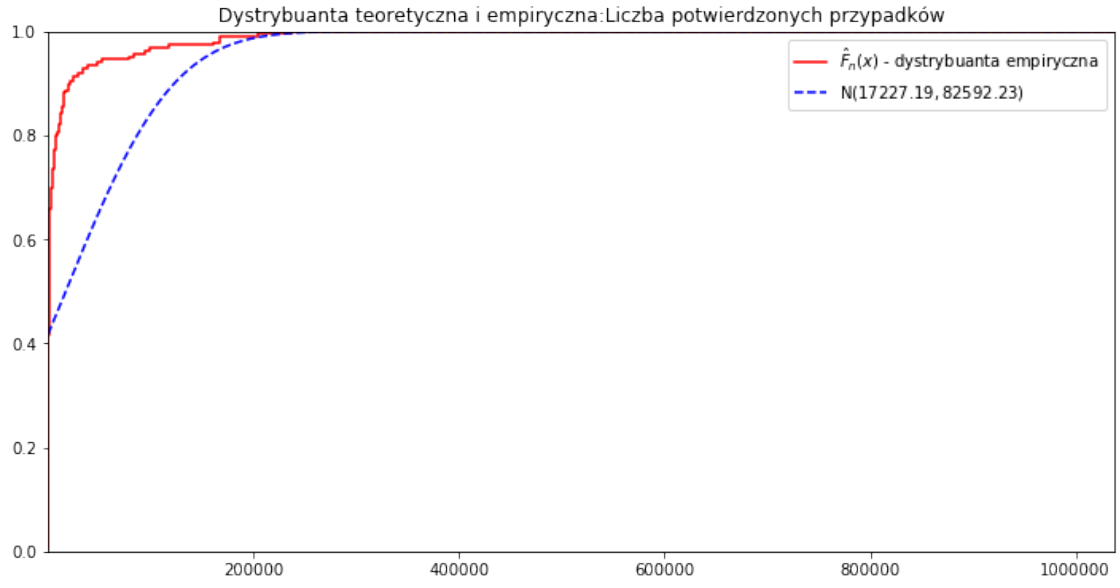
	Liczba_chorych	Współczynnik_śmiertelności
count	185.00	185.00
mean	10748.01	3.94
std	65109.57	4.17
min	0.00	0.00
25%	41.00	0.88
50%	453.00	2.87
75%	2550.00	5.52
max	856247.00	23.08

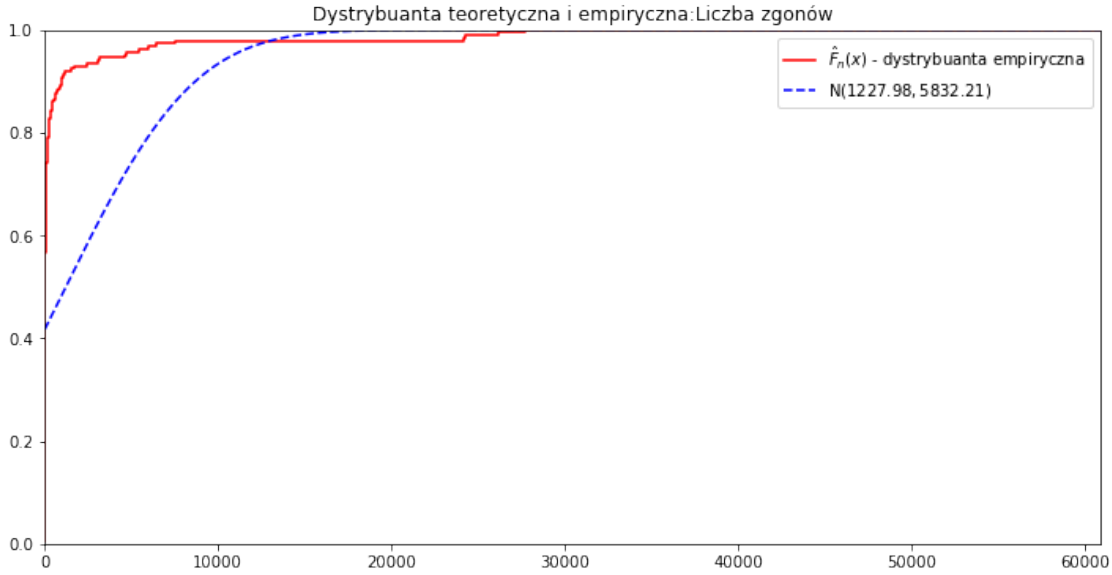
Zbiór danych który analizujemy zawiera 185 rekordów. Na podstawie otrzymanych wynikiów podstawowych parametrów statystycznych można stwierdzić duże zróżnicowanie badanych zmienych. Szczególnie należy zwrócić uwagę na odnotowane wartości maksymalne (dalej zostanie przedstawiona wizualizacja).

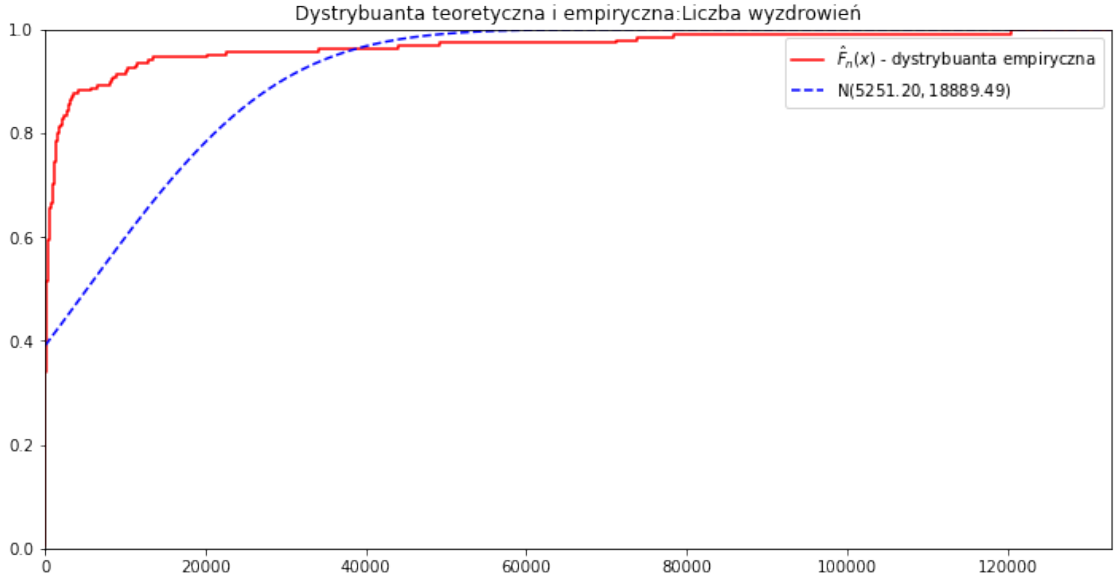
[7]:
```
class AnalizaStat:
    def __init__(self, k):
        self.k = k

        if self.k =='Liczba potwierdzonych przypadków':
             self.k1 = dane['Liczba_potwierdzonych_przypadków'].values
        elif self.k =='Liczba zgonów':
             self.k1 = dane['Liczba_zgonów'].values
        elif self.k =='Liczba wyzdrowień':
             self.k1 = dane['Liczba_wyzdrowień'].values
        elif self.k =='Liczba chorych':
             self.k1 = dane['Liczba_chorych'].values
        elif self.k =='Współczynnik śmiertelności':
             self.k1 = dane['Współczynnik_śmiertelności'].values
        else:
            print('Nie ma takiej kolumny!')

    def rozkład(self):
        alfa = 0.05
        m, s = sp.stats.norm.fit(self.k1)
        plt.figure(figsize=(12,6))
        plt.title('Dystrybuanta teoretyczna i empiryczna:'+ self.k)
        plt.axis([min(self.k1), max(self.k1), 0, 1])
        ecdf = ECDF(self.k1)
        plt.step(ecdf.x, ecdf.y,'r-', label = r'$\hat{F}_n(x)$ - dystrybuanta␣
↪empiryczna')
        t1 = np.linspace(min(self.k1), max(self.k1), 1000)
        t2 = sp.stats.norm.cdf(t1, loc = m, scale = s)
        plt.plot(t1, t2,'b--', label = r'$\mathrm{N}(%.2f, %.2f)$'% (m, s))
        plt.legend(loc ='best')
        test = sp.stats.shapiro(self.k1)
        print('Dla zmiennej '+self.k+' wartosc statystyki testowej: %s p-value:␣
↪%s'%(round(test[0],4),round(test[1],4)))
        if round(test[1],4) < alfa:
            print('Na podstanie przeprowadzonego testu odrzucamy hipotezę o␣
↪normalności rozkładu zmiennej: %s\n'%self.k)
        else:
            print('Nie ma podstaw do odrzucenia hipotezy o normalności rozkładu␣
↪zmiennej: %s\n'%self.k)

A1 = AnalizaStat('Liczba potwierdzonych przypadków')
A1.rozkład()
A2 = AnalizaStat('Liczba zgonów')
A2.rozkład()
A3 = AnalizaStat('Liczba wyzdrowień')
A3.rozkład()
A4 = AnalizaStat('Liczba chorych')
```

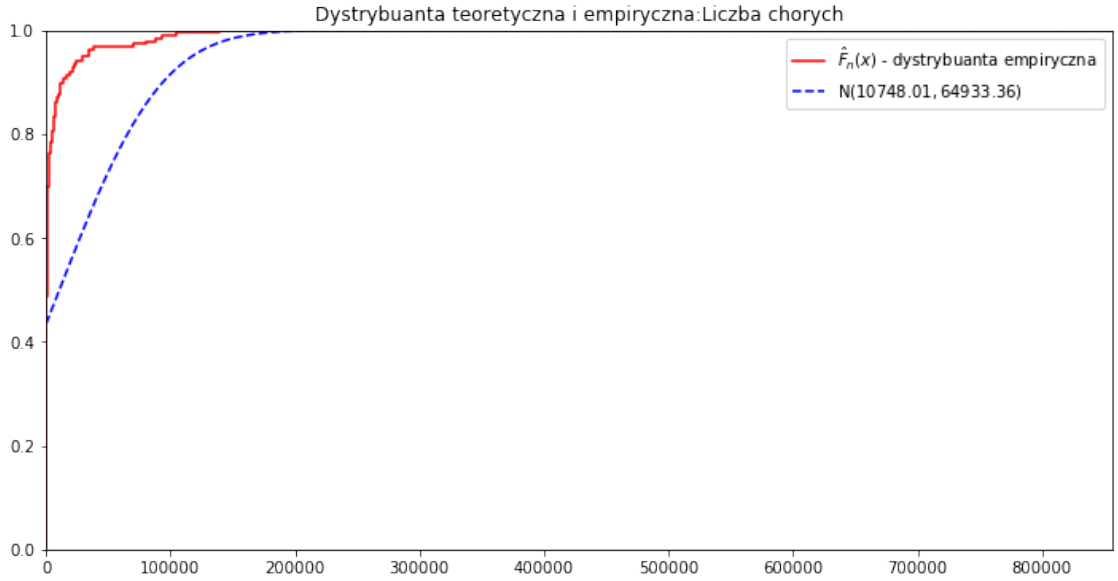
```
A4.rozkład()
A5 = AnalizaStat('Współczynnik śmiertelności')
A5.rozkład()
```

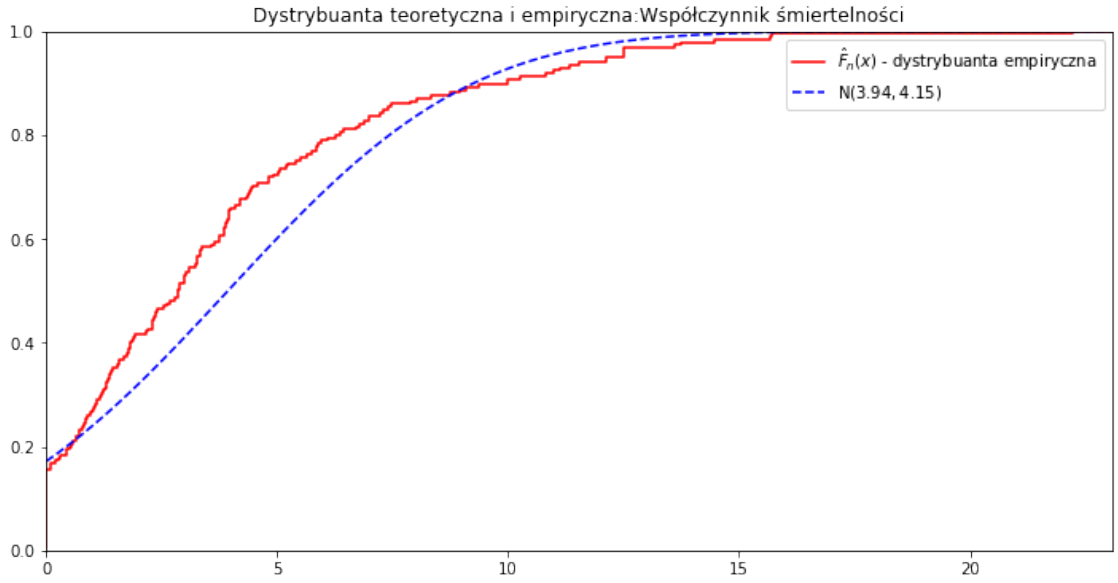
```
Dla zmiennej Liczba potwierdzonych przypadków wartosc statystyki testowej:
0.1876 p-value: 0.0
Na podstanie przeprowadzonego testu odrzucamy hipotezę o normalności rozkładu
zmiennej: Liczba potwierdzonych przypadków

Dla zmiennej Liczba zgonów wartosc statystyki testowej: 0.2106 p-value: 0.0
Na podstanie przeprowadzonego testu odrzucamy hipotezę o normalności rozkładu
zmiennej: Liczba zgonów

Dla zmiennej Liczba wyzdrowień wartosc statystyki testowej: 0.2942 p-value: 0.0
Na podstanie przeprowadzonego testu odrzucamy hipotezę o normalności rozkładu
zmiennej: Liczba wyzdrowień

Dla zmiennej Liczba chorych wartosc statystyki testowej: 0.1342 p-value: 0.0
Na podstanie przeprowadzonego testu odrzucamy hipotezę o normalności rozkładu
zmiennej: Liczba chorych

Dla zmiennej Współczynnik śmiertelności wartosc statystyki testowej: 0.8278
p-value: 0.0
Na podstanie przeprowadzonego testu odrzucamy hipotezę o normalności rozkładu
zmiennej: Współczynnik śmiertelności
```




Dystrybuanta teoretyczna i empiryczna:Liczba zgonów
$\hat{F}_n(x)$ - dystrybuanta empiryczna
N(1227.98, 5832.21)
1.0
0.8
0.6
0.4
0.2
0.0
0
10000
20000
30000
40000
50000
60000

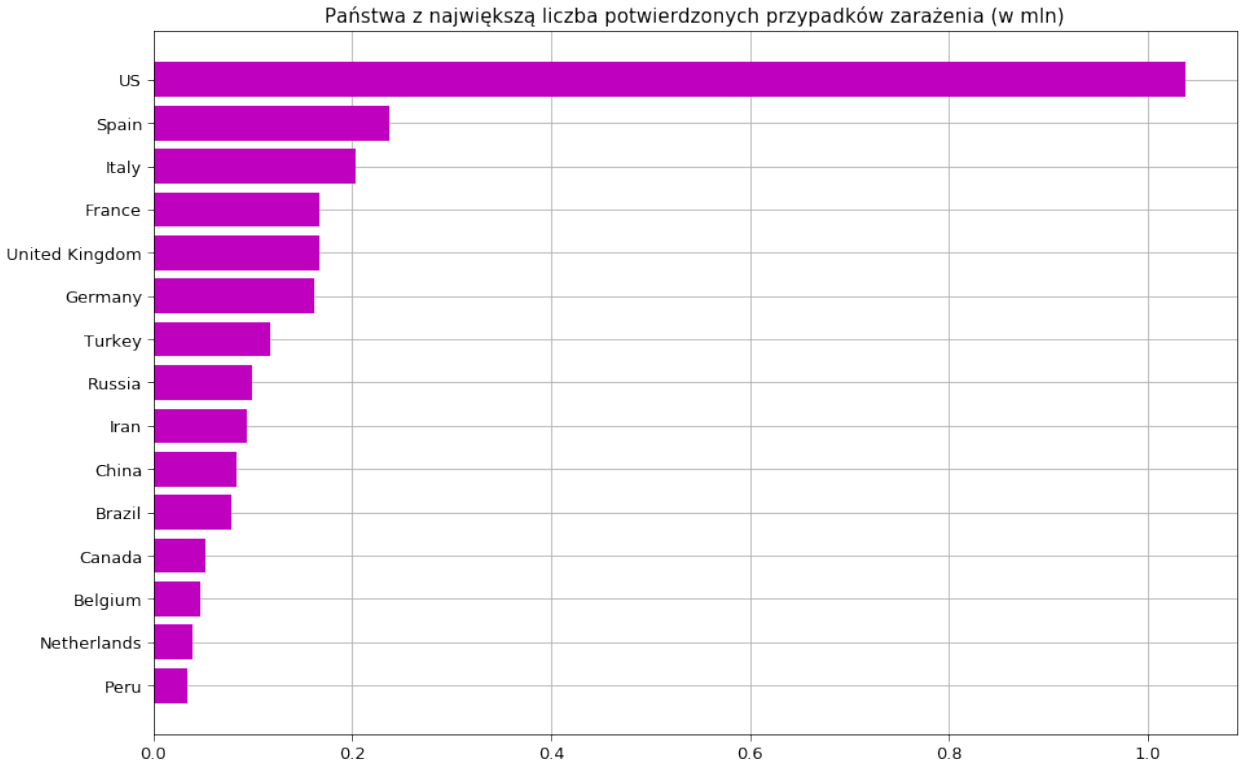

Dystrybuanta teoretyczna i empiryczna:Liczba wyzdrowień
$\hat{F}_n(x)$ - dystrybuanta empiryczna
N(5251.20, 18889.49)
1.0
0.8
0.6
0.4
0.2
0.0
0
20000
40000
60000
80000
100000
120000



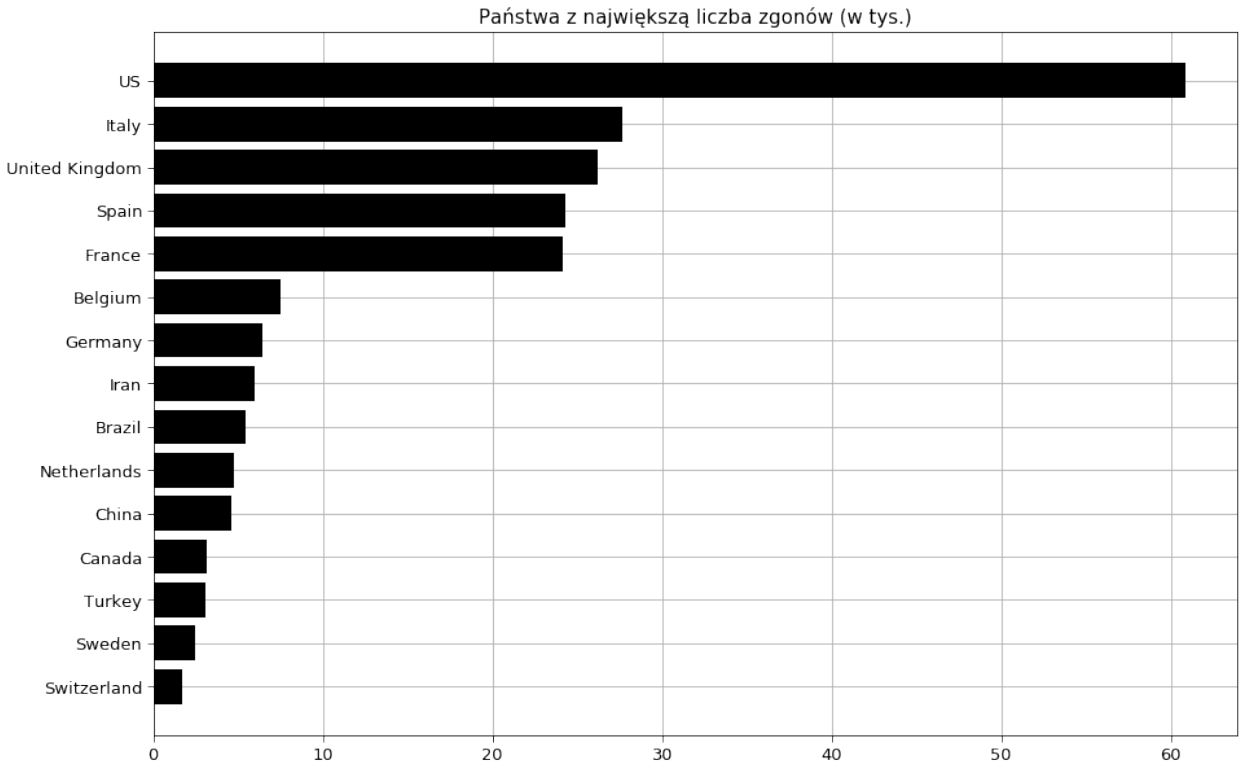


[8]:
```
# Wizualizacja danych

dane_kp = dane.copy().drop(['Ostatnia_aktualizacja'],axis =1)
dane_kp.index = dane_kp["Region"]
dane_kp = dane_kp.drop(['Region'],axis=1)
```

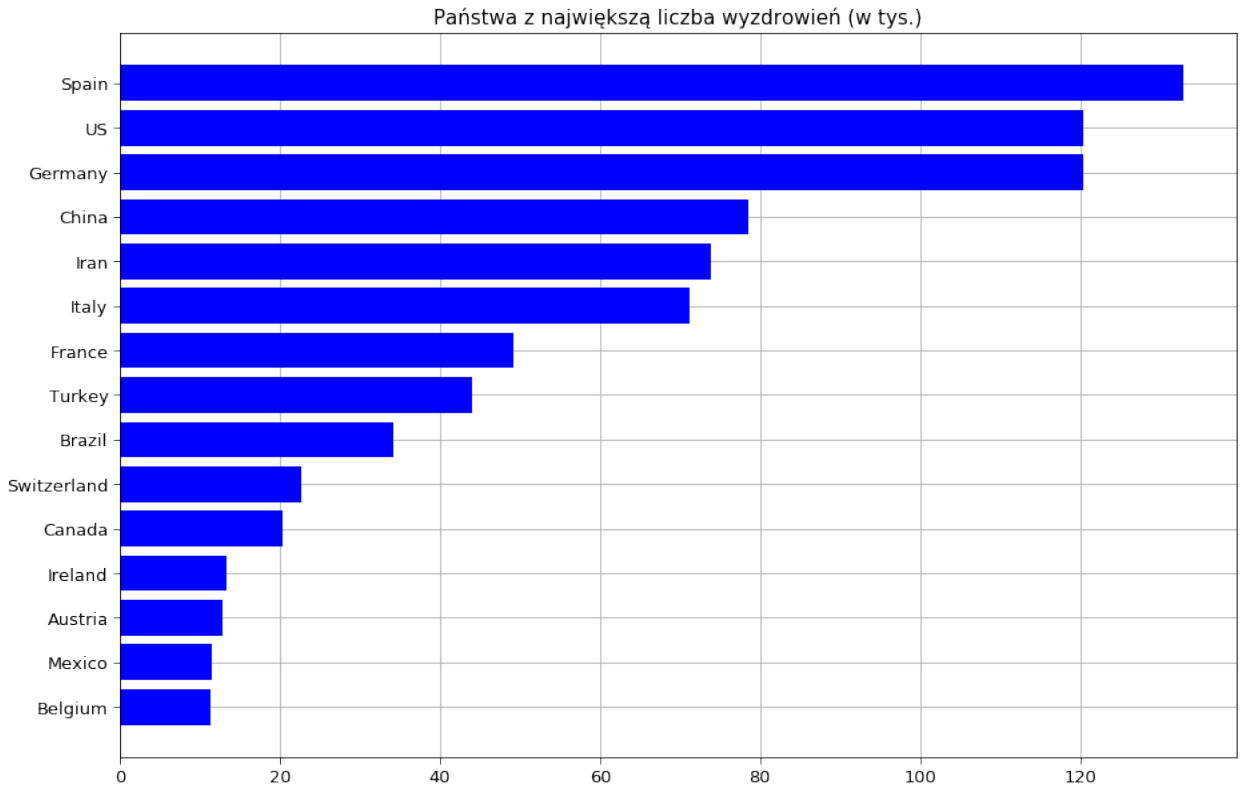
[9]:
```
plt.figure(figsize = (15,10))
plt.axes(axisbelow = True)
plt.barh(dane_kp.
 ↪sort_values('Liczba_potwierdzonych_przypadków')["Liczba_potwierdzonych_przypadków"].
 ↪index[-15:],dane_kp.
 ↪sort_values('Liczba_potwierdzonych_przypadków')["Liczba_potwierdzonych_przypadków"].
 ↪values[-15:]/1000000, color = 'm')
plt.tick_params(size = 5,labelsize = 13)
plt.title("Państwa z największą liczba potwierdzonych przypadków zarażenia (w␣
 ↪mln)",fontsize = 15)
plt.grid()
```



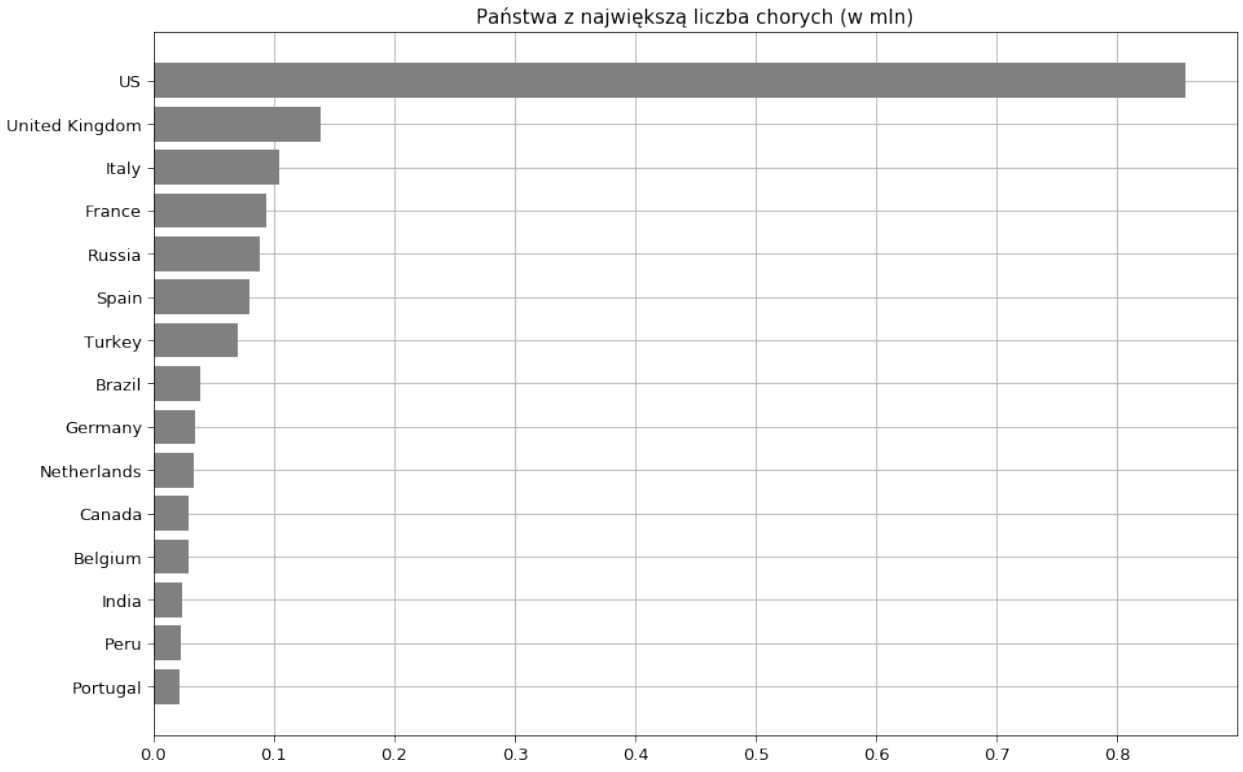
[10]:
```
plt.figure(figsize = (15,10))
plt.axes(axisbelow = True)
plt.barh(dane_kp.sort_values('Liczba_zgonów')["Liczba_zgonów"].index[-15:
 ↪],dane_kp.sort_values('Liczba_zgonów')["Liczba_zgonów"].values[-15:]/1000,␣
 ↪color = 'k')
plt.tick_params(size = 5,labelsize = 13)
plt.title("Państwa z największą liczba zgonów (w tys.)", fontsize = 15)
plt.grid()
```



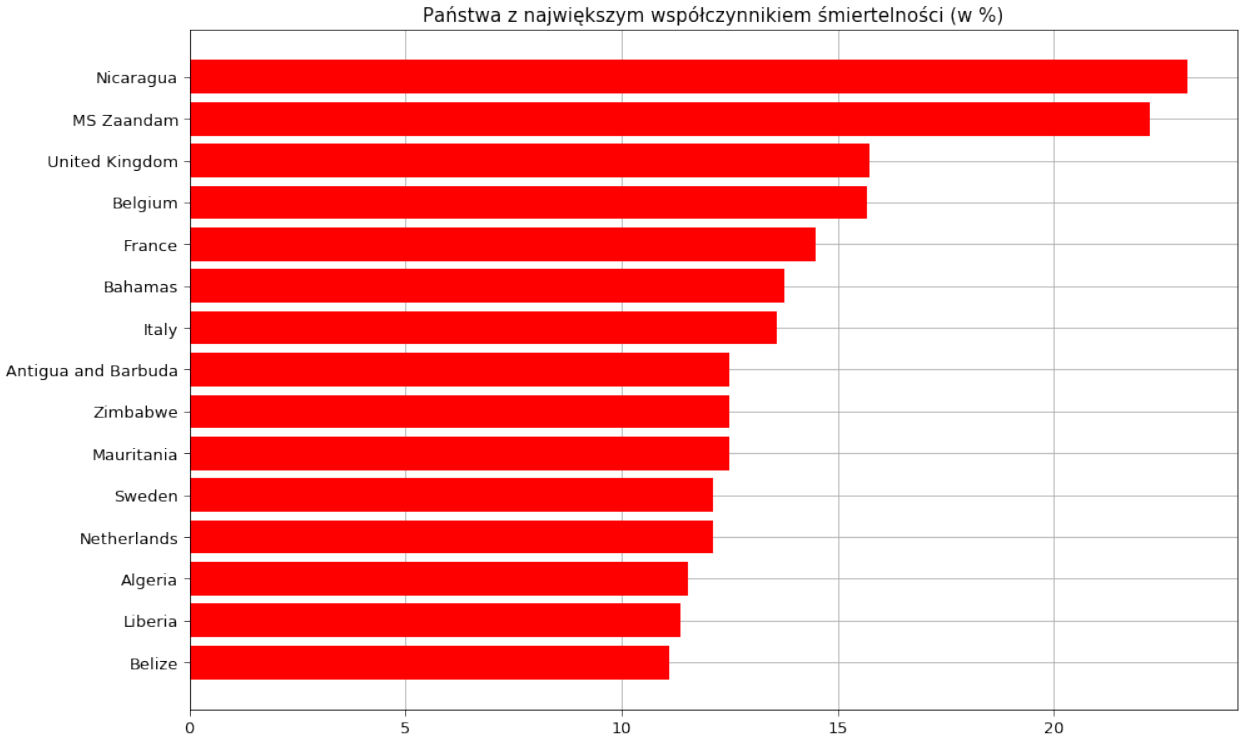
[11]:
```
plt.figure(figsize = (15,10))
plt.axes(axisbelow = True)
plt.barh(dane_kp.sort_values('Liczba_wyzdrowień')["Liczba_wyzdrowień"].index[-15:
 ↪],dane_kp.sort_values('Liczba_wyzdrowień')["Liczba_wyzdrowień"].values[-15:]/
 ↪1000,color = 'b')
plt.tick_params(size=5,labelsize = 13)
plt.title("Państwa z największą liczba wyzdrowień (w tys.)",fontsize = 15)
plt.grid()
```



[12]:
```
plt.figure(figsize = (15,10))
plt.axes(axisbelow = True)
plt.barh(dane_kp.sort_values('Liczba_chorych')["Liczba_chorych"].index[-15:
 ↪],dane_kp.sort_values('Liczba_chorych')["Liczba_chorych"].values[-15:]/
 ↪1000000, color = 'grey')
plt.tick_params(size = 5,labelsize = 13)
plt.title("Państwa z największą liczba chorych (w mln)",fontsize = 15)
plt.grid()
```



[13]:
```
plt.figure(figsize = (15,10))
plt.axes(axisbelow = True)
plt.barh(dane_kp.
 ↪sort_values('Współczynnik_śmiertelności')["Współczynnik_śmiertelności"].
 ↪index[-15:],dane_kp.
 ↪sort_values('Współczynnik_śmiertelności')["Współczynnik_śmiertelności"].
 ↪values[-15:], color = 'r')
plt.tick_params(size = 5, labelsize = 13)
plt.title("Państwa z największym współczynnikiem śmiertelności (w %)", fontsize␣
 ↪= 15)
plt.grid()
```



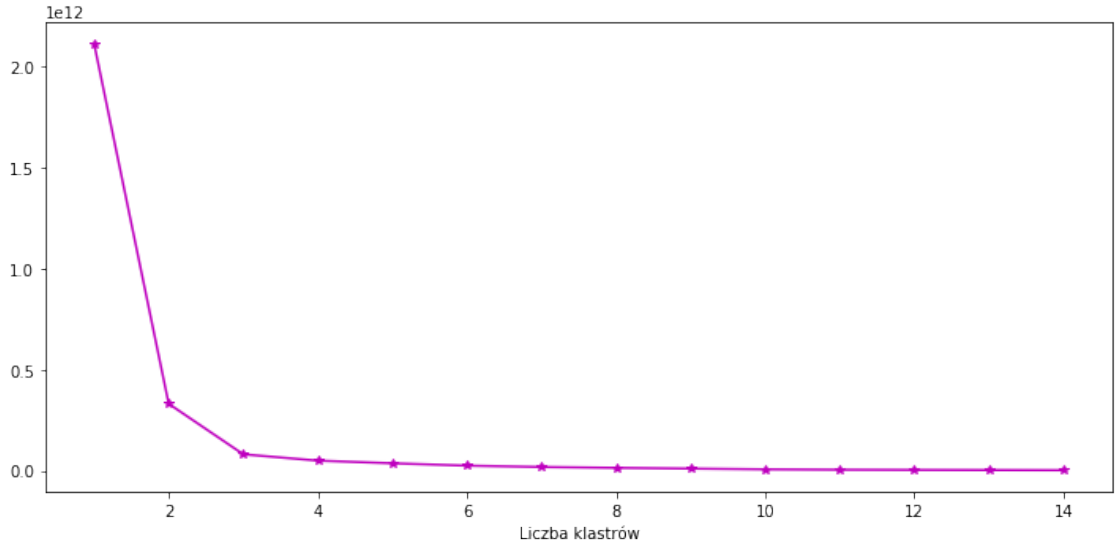
[14]:
```
# Macierz korelacji

dane_kp.iloc[:,:].corr().style.background_gradient(cmap = 'copper', low = 0.5,␣
 ↪high = 0.5)
```

[14]:
```
<pandas.io.formats.style.Styler at 0x7f7d500988d0>
```

Wysokie wartości korelacji między zmiennymi: Liczba potwierdzonych przypadków, Liczba zgonów, Liczba wyzdrowień, Liczba chorych są wyniki zależności liniowej jaką te zmienne są ze sobą powiązane.

Poniżej za pomocą algorytmu k-średnich została wykonana klasteryzacja analizowanych danych czyli podział danych wejściowych na z góry ustaloną liczbę klas. Klasteryzacja zalicza się do nienadzorowanego uczenia maszynowego. Metod tego typu używa się do wykrycia zależności pomiędzy danymi bez korzystania z zewnętrznego ich opisu. Metoda dokonuje podziału danych na pewną liczbę rozłącznych zbiorów w taki sposób, aby każdy z otrzymanych zbiorów zawierał obiekty możliwie do siebie podobne (wedle ustalonego kryterium podobieństwa), przy jednoczesnym zachowaniu możliwie dużego niepodobieństwa wobec obiektów z pozostałych grup.

[15]:
```
# Używając metody łokcia znaleziona została optymalna liczba klastrów

X = dane.drop(['Region','Ostatnia_aktualizacja'], axis = 1)
d = []
for i in range(1, 15):
    km = KMeans(n_clusters = i, init = 'k-means++', n_init = 10, max_iter = 300,␣
 ↪random_state = 0)
    km.fit(X)
    d.append(km.inertia_)
plt.figure(figsize = (10,5))
plt.plot(range(1, 15), d, marker = '*', color = 'm')
plt.xlabel('Liczba klastrów')
plt.tight_layout()
plt.show()
```



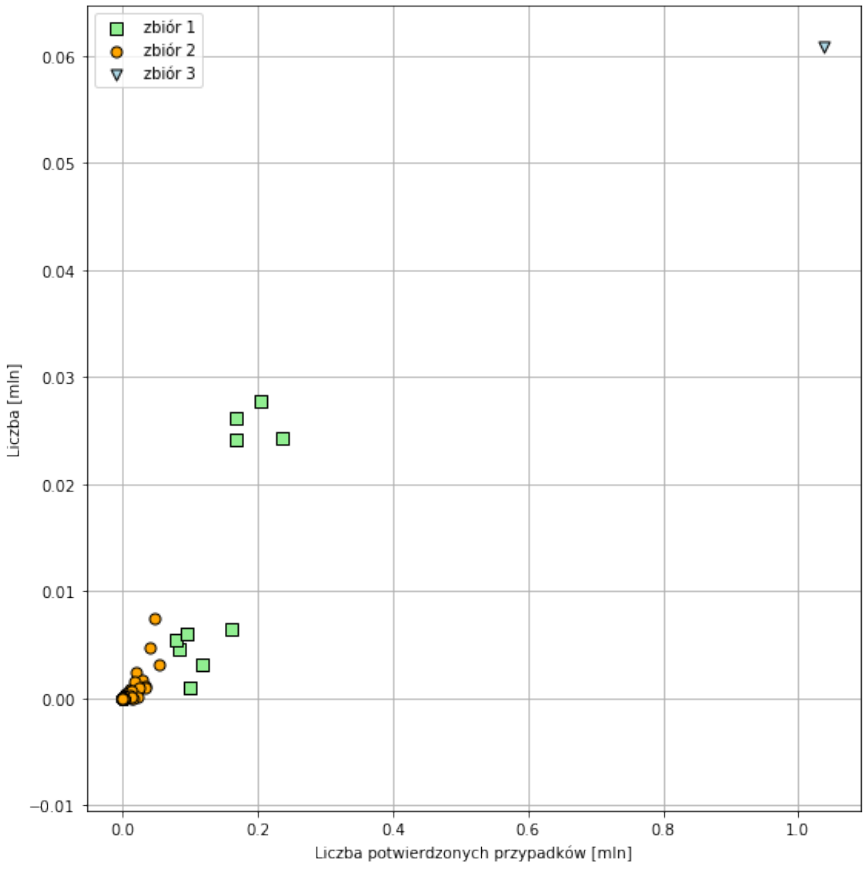
[16]:
```
# Klasteryzacja

km = KMeans(n_clusters = 3, init = 'random', n_init = 10, max_iter = 300, tol =␣
 ↪1e-04, random_state = 0)
X = dane.drop(['Region','Ostatnia_aktualizacja'], axis = 1)
y_km = km.fit_predict(X)
```

[17]:
```
# Wypisanie nazw krajów i regionów znajdujących się w każdym z klastrów

print('\nRegiony zostały podzielone na trzy grupy. Elementy grup zostały␣
 ↪wypisane poniżej:\n')
g0 = dane['Region'][y_km == 0]
print(g0.values,'\n')
g1 = dane['Region'][y_km == 1]
print(g1.values,'\n')
g3 = dane['Region'][y_km == 2]
print(g3.values)
```

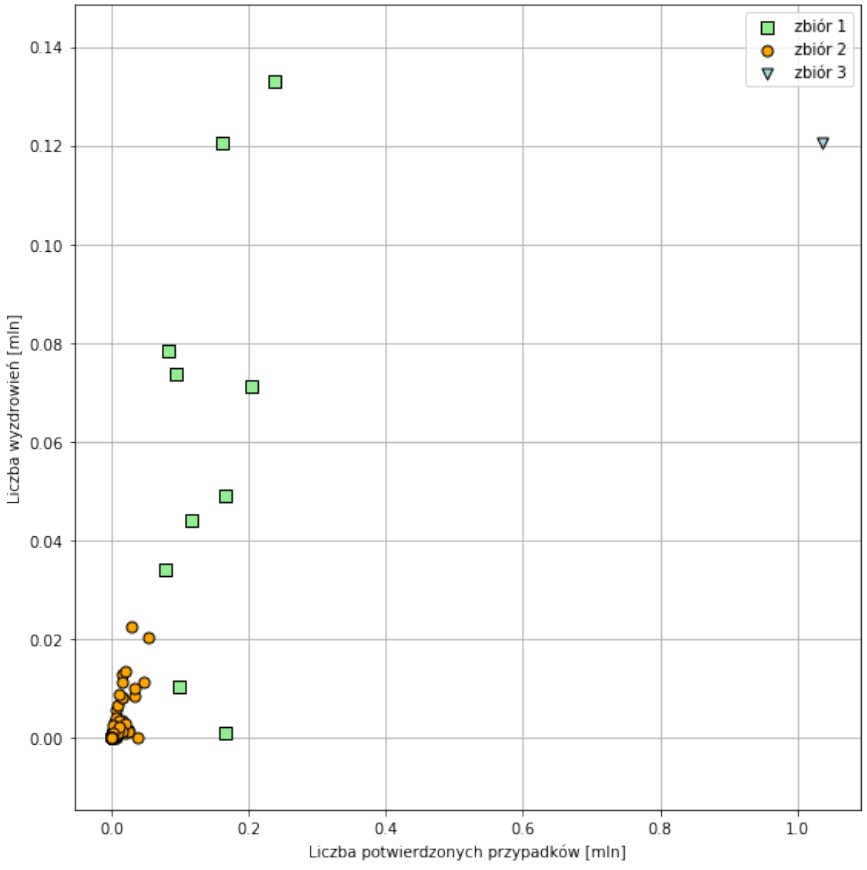
Regiony zostały podzielone na trzy grupy. Elementy grup zostały wypisane poniżej:

```
['China' 'France' 'Germany' 'Italy' 'Russia' 'United Kingdom' 'Brazil'
 'Iran' 'Spain' 'Turkey']

['Australia' 'Austria' 'Canada' 'Denmark' 'Finland' 'Iceland' 'Ireland'
 'Netherlands' 'Norway' 'Sweden' 'Switzerland' 'Afghanistan' 'Albania'
 'Algeria' 'Andorra' 'Angola' 'Antigua and Barbuda' 'Argentina' 'Armenia'
 'Azerbaijan' 'Bahamas' 'Bahrain' 'Bangladesh' 'Barbados' 'Belarus'
 'Belgium' 'Belize' 'Benin' 'Bhutan' 'Bolivia' 'Bosnia and Herzegovina'
 'Botswana' 'Brunei' 'Bulgaria' 'Burkina Faso' 'Burma' 'Burundi'
 'Cabo Verde' 'Cambodia' 'Cameroon' 'Central African Republic' 'Chad'
 'Chile' 'Colombia' 'Congo (Brazzaville)' 'Congo (Kinshasa)' 'Costa Rica'
 "Cote d'Ivoire" 'Croatia' 'Cuba' 'Cyprus' 'Czechia' 'Diamond Princess'
 'Djibouti' 'Dominica' 'Dominican Republic' 'Ecuador' 'Egypt'
 'El Salvador' 'Equatorial Guinea' 'Eritrea' 'Estonia' 'Eswatini'
 'Ethiopia' 'Fiji' 'Gabon' 'Gambia' 'Georgia' 'Ghana' 'Greece' 'Grenada'
 'Guatemala' 'Guinea' 'Guinea-Bissau' 'Guyana' 'Haiti' 'Holy See'
 'Honduras' 'Hungary' 'India' 'Indonesia' 'Iraq' 'Israel' 'Jamaica'
 'Japan' 'Jordan' 'Kazakhstan' 'Kenya' 'Korea, South' 'Kosovo' 'Kuwait'
 'Kyrgyzstan' 'Laos' 'Latvia' 'Lebanon' 'Liberia' 'Libya' 'Liechtenstein'
 'Lithuania' 'Luxembourg' 'MS Zaandam' 'Madagascar' 'Malawi' 'Malaysia'
 'Maldives' 'Mali' 'Malta' 'Mauritania' 'Mauritius' 'Mexico' 'Moldova'
 'Monaco' 'Mongolia' 'Montenegro' 'Morocco' 'Mozambique' 'Namibia' 'Nepal'
 'New Zealand' 'Nicaragua' 'Niger' 'Nigeria' 'North Macedonia' 'Oman'
 'Pakistan' 'Panama' 'Papua New Guinea' 'Paraguay' 'Peru' 'Philippines'
 'Poland' 'Portugal' 'Qatar' 'Romania' 'Rwanda' 'Saint Kitts and Nevis'
 'Saint Lucia' 'Saint Vincent and the Grenadines' 'San Marino'
 'Sao Tome and Principe' 'Saudi Arabia' 'Senegal' 'Serbia' 'Seychelles'
 'Sierra Leone' 'Singapore' 'Slovakia' 'Slovenia' 'Somalia' 'South Africa'
 'South Sudan' 'Sri Lanka' 'Sudan' 'Suriname' 'Syria' 'Taiwan*' 'Tanzania'
 'Thailand' 'Timor-Leste' 'Togo' 'Trinidad and Tobago' 'Tunisia' 'Uganda'
 'Ukraine' 'United Arab Emirates' 'Uruguay' 'Uzbekistan' 'Venezuela'
 'Vietnam' 'West Bank and Gaza' 'Western Sahara' 'Yemen' 'Zambia'
 'Zimbabwe']

['US']
```

[18]:
```
# Wizualizacja otrzymanych klastrów ze względu na różne kombinacje zmiennych

plt.figure(figsize = (8,8))
plt.axes(axisbelow=True)
plt.scatter(X.Liczba_potwierdzonych_przypadków[y_km == 0]/1000000, X.
 ↪Liczba_zgonów[y_km == 0]/1000000, s = 50, c = 'lightgreen', marker = 's',␣
 ↪edgecolor = 'black', label = 'zbiór 1')
plt.scatter(X.Liczba_potwierdzonych_przypadków[y_km == 1]/1000000, X.
 ↪Liczba_zgonów[y_km == 1]/1000000, s = 50, c = 'orange', marker = 'o',␣
 ↪edgecolor = 'black', label = 'zbiór 2')
plt.scatter(X.Liczba_potwierdzonych_przypadków[y_km == 2]/1000000, X.
 ↪Liczba_zgonów[y_km == 2]/1000000, s = 50, c = 'lightblue', marker = 'v',␣
 ↪edgecolor = 'black', label = 'zbiór 3')
plt.legend()
plt.xlabel('Liczba potwierdzonych przypadków [mln]')
plt.ylabel('Liczba [mln]')
plt.grid()
plt.tight_layout()
plt.show()
```



[19]:
```
plt.figure(figsize = (8,8))
plt.axes(axisbelow=True)
plt.scatter(X.Liczba_potwierdzonych_przypadków[y_km == 0]/1000000, X.
 ↪Liczba_wyzdrowień[y_km == 0]/1000000, s = 50, c = 'lightgreen', marker = 's',␣
 ↪edgecolor = 'black', label = 'zbiór 1')
plt.scatter(X.Liczba_potwierdzonych_przypadków[y_km == 1]/1000000, X.
 ↪Liczba_wyzdrowień[y_km == 1]/1000000, s = 50, c = 'orange', marker = 'o',␣
 ↪edgecolor = 'black', label = 'zbiór 2')
plt.scatter(X.Liczba_potwierdzonych_przypadków[y_km == 2]/1000000, X.
 ↪Liczba_wyzdrowień[y_km == 2]/1000000, s = 50, c = 'lightblue', marker = 'v',␣
 ↪edgecolor = 'black', label = 'zbiór 3')
plt.legend()
plt.xlabel('Liczba potwierdzonych przypadków [mln]')
plt.ylabel('Liczba wyzdrowień [mln]')
plt.grid()
plt.tight_layout()
plt.show()
```



[20]:
```
plt.figure(figsize = (8,8))
plt.axes(axisbelow=True)
plt.scatter(X.Liczba_potwierdzonych_przypadków[y_km == 0]/1000000, X.
 ↪Liczba_chorych[y_km == 0]/1000000, s = 50, c = 'lightgreen', marker = 's',␣
 ↪edgecolor = 'black', label = 'zbiór 1')
plt.scatter(X.Liczba_potwierdzonych_przypadków[y_km == 1]/1000000, X.
 ↪Liczba_chorych[y_km == 1]/1000000, s = 50, c = 'orange', marker = 'o',␣
 ↪edgecolor = 'black', label = 'zbiór 2')
plt.scatter(X.Liczba_potwierdzonych_przypadków[y_km == 2]/1000000, X.
 ↪Liczba_chorych[y_km == 2]/1000000, s = 50, c = 'lightblue', marker = 'v',␣
 ↪edgecolor = 'black', label = 'zbiór 3')
plt.legend()
plt.xlabel('Liczba potwierdzonych przypadków [mln]')
plt.ylabel('Liczba chorych [mln]')
plt.grid()
plt.tight_layout()
plt.show()
```