

SPRAWOZDANIE

Zajęcia: Eksploracja i wizualizacja danych

Prowadzący: prof. dr hab. inż. Vasyl Martsenyuk

Laboratorium nr 2 Data: 16.10.2021 Temat: Graficzna wizualizacja danych z użyciem bibliotek „matplotlib” i „seaborn” Wariant: 1	Piotr Rybka Informatyka II stopień, niestacjonarne, III semestr, gr. 1
--	---

Repozytorium z kodem programu:

https://github.com/prybka82-student/eksploracja_i_wizualizacja_danych/tree/zadanie02

Polecenie

Na podstawie danych ze zbioru [1] lub danych losowych, wygenerować wykresy analogiczne do przedstawionych na zajęciach.

Zadanie 1. Wygenerować wykres liniowy

Załadowanie bibliotek:

```
1 import matplotlib.pyplot as plt
2 %matplotlib inline
3
4 import numpy as np
5
6 import pandas as pd
7
8 import os
```

Załadowanie danych:

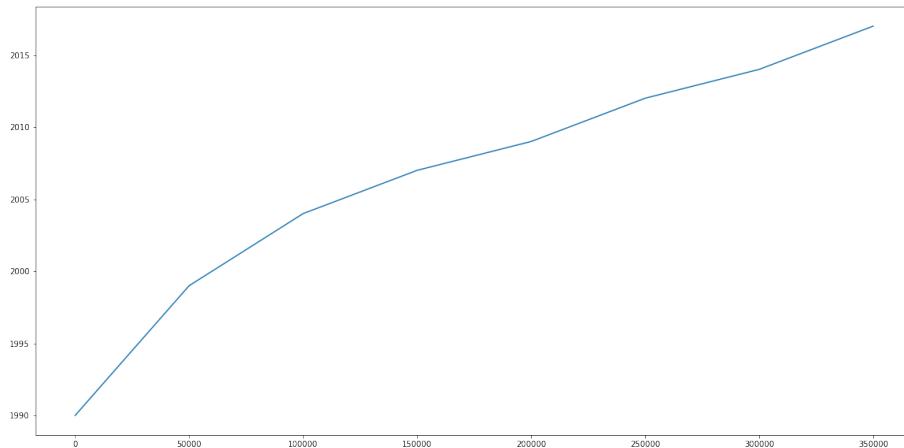
```
1 path = r"C:\Users\piotr\Downloads\EiWD_lab_zadanie02\
    IHME_DAH_DATABASE_1990_2020_Y2021M09D22.CSV"
2 folder = r"C:\Users\piotr\Documents\Semestr 3\EiWD_zadanie_02_wykresy"
3
4 data = pd.read_csv(path, low_memory=False)
```

Wygenerowanie wykresu dla co pięćdziesięciotysięcznej wartości:

```
1 years = data['year'][::50_000]
2
3 plt.rcParams["figure.figsize"] = (20,10)
4 plt.plot(years)
```

Uzyskany wykres – 1.

Rysunek 1: Wykres liniowy (dane rzeczywiste za [1])



Zadanie 2. Wygenerować kilka wykresów obok siebie

Utworz się siatki wykresów:

```
1 fig = plt.figure()
2 ax0 = fig.add_subplot(2,2,1)
3 ax1 = fig.add_subplot(2,2,2)
4 ax2 = fig.add_subplot(2,2,3)
```

Wyodrębnienie danych: co dziesięciotysięczna wartość z kolumny „elim_ch” i co tysięczna wartość z kolumny „prelim_est” z pominięciem pierwszych 350 tysięcy:

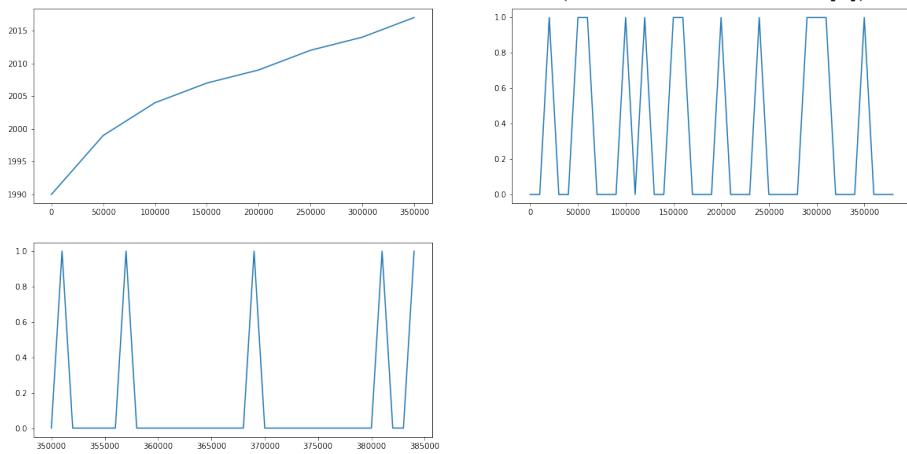
```
1 elim_ch = data['elim_ch'][::10_000]
2 prelim_est = data['prelim_est'][350_000::1_000]
```

Wygenerowanie wykresów:

```
1 ax0.plot(years)
2 ax1.plot(elim_ch)
3 ax2.plot(prelim_est)
```

Uzyskany wykres – 2.

Rysunek 2: Kilka wykresów obok siebie (dane rzeczywiste za [1])



Zadanie 3. Wygenerować wykres liniowy i zdefiniować jego formatowanie

Załadowanie biblioteki generującej losowe dane:

```
1 from numpy.random import randn
```

Wygenerowanie wykresu liniowego zawierającego 30 losowych wartości z przedziału od 0 do 1, przy czym każda kolejna wartość jest dodawana do poprzedniej. Formatowanie wykresu: zielone punkty połączone linią ciągłą.

```
1 plt.plot(randn(30).cumsum(), 'go-')
```

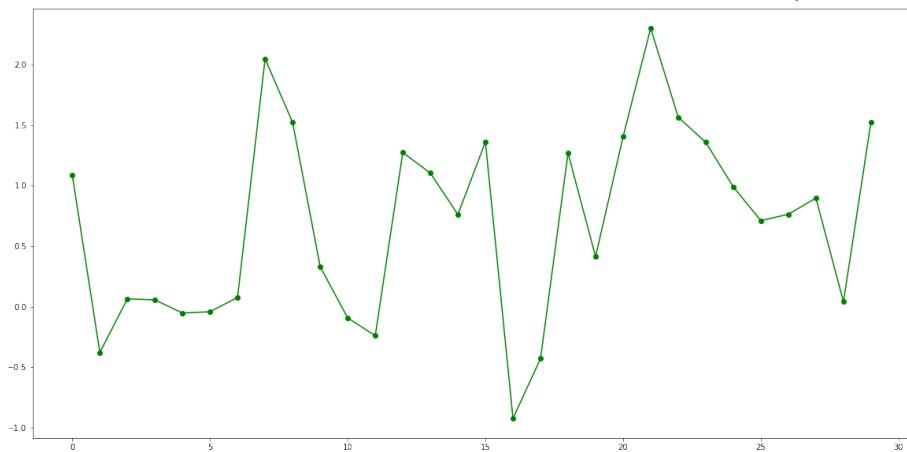
Uzyskany wykres – 3.

Inne formatowanie – wykres w kolorze brązowym, znaczniki w kształcie kółek o wielkości 22 pikseli, linia przerywana o szerokości 1 piksela:

```
1 plt.plot(years, color='brown', marker='o', markersize=22, linestyle='--', linewidth=1)
```

Uzyskany wykres – 4.

Rysunek 3: Wykres liniowy z dodanym formatowaniem własnym (dane losowe)



Zadanie 4. Wygenerować wykres liniowy z dodanym wykresem interpolacji liniowej i legendą

Wygenerowanie losowych danych (30 wartości, suma skumulowana):

```
1 data = np.random.randn(30).cumsum()
```

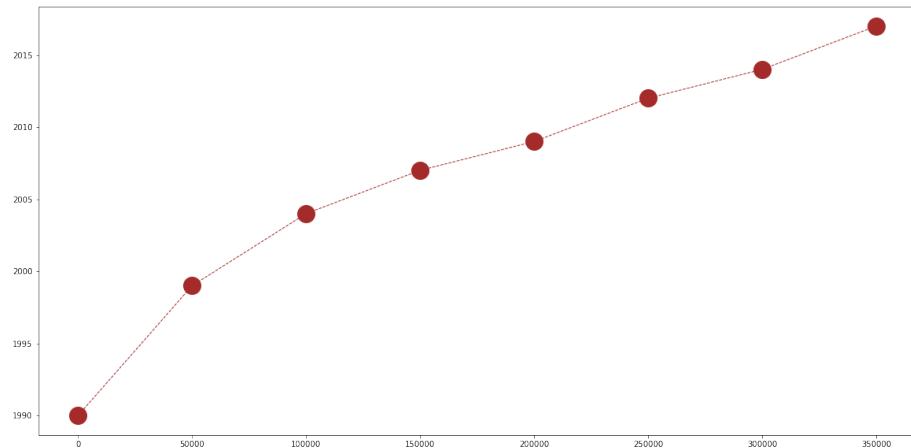
Wygenerowanie wykresów:

- dane właściwe – wykres czarny podpisany „Etykieta” wykonany linią przerywaną;
- wykres interpolacji liniowej – wykres czerwony linią ciągłą.

```
1 data = np.random.randn(30).cumsum()
2 plt.plot(data, 'k--', label='Etykieta')
3 plt.plot(data, 'r-', label="steps-post", drawstyle="steps-post")
4 plt.legend(loc='best')
```

Uzyskany wykres – 5.

Rysunek 4: Wykres liniowy z dodanym formatowaniem własnym (dane rzeczywiste za [1])



Zadanie 5. Wygenerować wykres liniowy i dodać do niego samodzielnie zdefiniowane etykiety osi

Wygenerowanie pustego wykresu (zmienna „ax”) i danych losowych (tysiąc wartości skumulowanych):

```
1 fig = plt.figure()
2 ax = fig.add_subplot(1,1,1)
3 ax.plot(np.random.randn(1000).cumsum())
```

Wyznaczenie miejsc na obu osiach, w których mają znaleźć się etykiety:

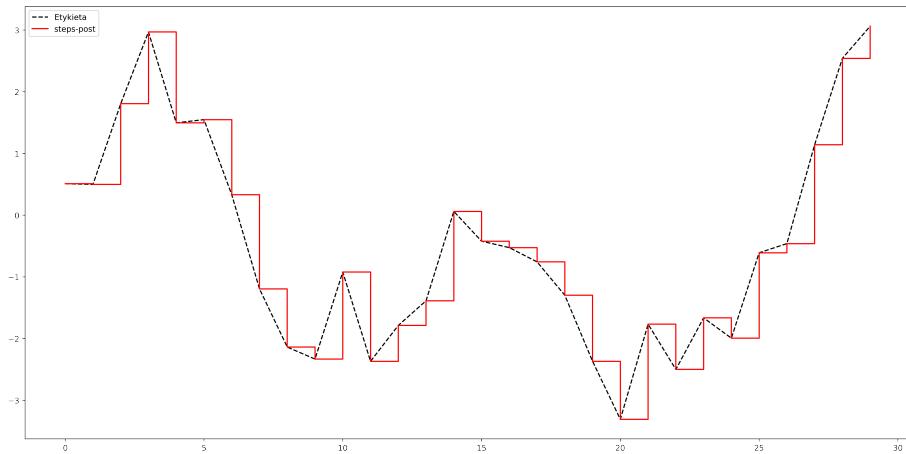
```
1 ax.set_xticks([0, 250, 500, 750, 1000])
2 ax.set_yticks([-50, -25, 0, 25, 50])
```

Dodanie etykiet wartości na osi x oraz etykiet obu osi oraz tytułu wykresu przy użyciu słownika (zmienna „props”):

```
1 ax.set_xticklabels(['one', 'two', 'three', 'four', 'five'], rotation=90,
                     fontsize='small')
2
3 props = {
4     'title': 'Tytul',
5     'xlabel': 'Kroki',
6     'ylabel': 'Wartosci'
7 }
8 ax.set(**props)
```

Uzyskany wykres – 6.

Rysunek 5: Wykres liniowy z dodanym wykresem interpolacji liniowej (dane losowe)



Zadanie 6. Wygenerować kilka wykresów liniowych w różnych kolorach i nałożyć je na siebie

Utworzenie wykresu zawierającego jeden podwykres:

```
1 fig = plt.figure();
2 ax = fig.add_subplot(1,1,1)
```

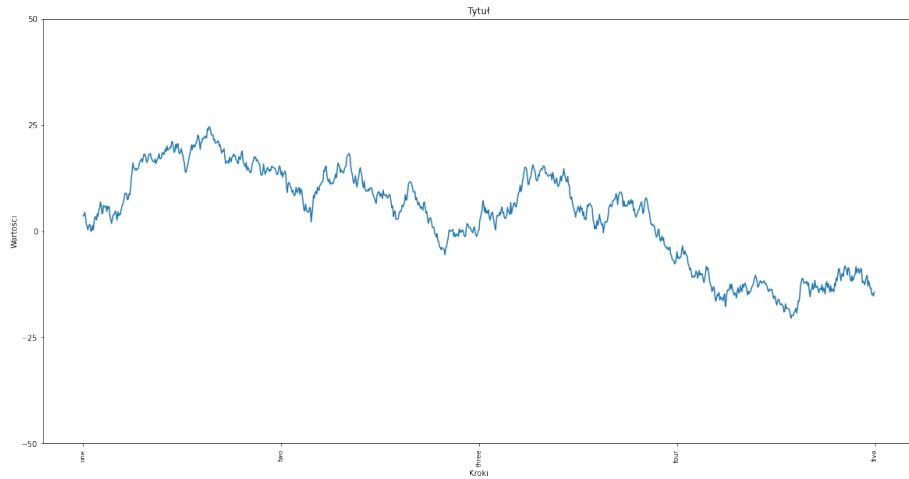
Naniesienie losowych wartości i dodanie legenty oraz następującego formatowania:

- wykres 1. – kolor czarny, linia ciągła, etykieta: „one”,
- wykres 2. – kolor czerwony, linia przerywana, etykieta: „two”,
- wykres 3. – kolor zielony, linia kropkowana, etykieta: „three”.

```
1 ax.plot(randn(1000).cumsum(), 'k', label='one')
2 ax.plot(randn(1000).cumsum(), 'r--', label='two')
3 ax.plot(randn(1000).cumsum(), 'g.', label='three')
4 ax.legend(loc='best')
```

Uzyskany wykres – 7.

Rysunek 6: Wykres liniowy z dodanymi własnymi etykietami (dane losowe)

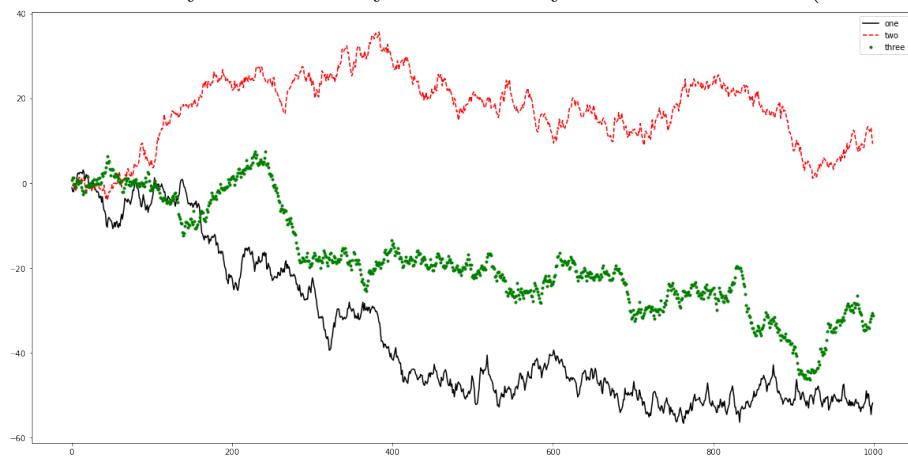


Zadanie 7. Zapisać wygenerowany wykres do pliku

Kod pozwalający na zapisanie wykresu ze wskazaniem rozdzielczości (400 dpi) i szerokości marginesów (wąskie – *tight*):

```
1 path = r"C:\Users\piotr\Downloads\wykres01.svg"
2
3 plt.savefig(path, dpi=400, bbox_inches='tight')
```

Rysunek 7: Kilka wykresów liniowych odróżnionych formatowaniem (dane losowe)



Zadanie 8. Wygenerować wykres liniowy z użyciem indeksu

Zimportowanie bibliotek i zdefiniowanie maksymalnego zakresu osi OX :

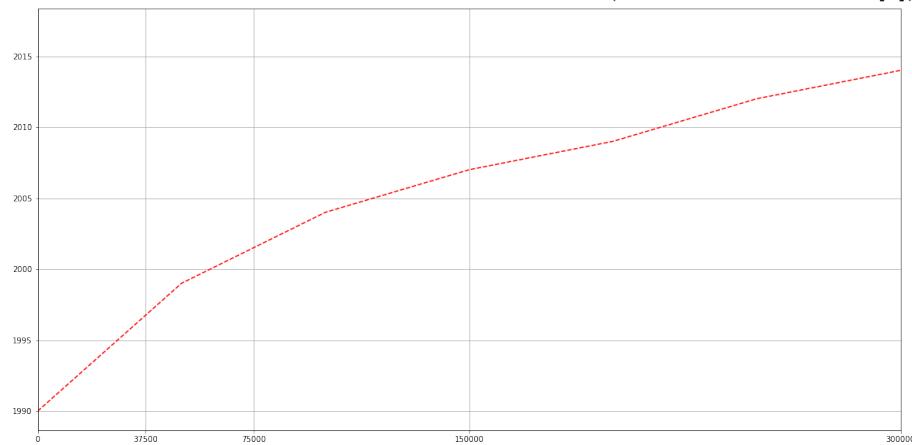
```
1 import numpy as np
2 import pandas as pd
3
4 xmax = 300_000
```

Wygenerowanie wykresu z indeksem zawierającym 50 tysięcy wartości w przedziale od 0 do 1 499 950 000:

```
1 s = pd.Series(years, index=np.arange(0, xmax*50_000, 50_000))
2 s.plot(kind='line', logy=False, xticks=[0, xmax/8, xmax/4, xmax/2, xmax],
         xlim=[0, xmax], grid=True, style='r--')
```

Uzyskany wykres – 8.

Rysunek 8: Wykres liniowy z użyciem indeksu (dane rzeczywiste za [1])



Zadanie 9. Wygenerować kilka wykresów liniowych ułożonych jeden nad drugim

Wygenerowanie losowych danych (6 serii po 10 losowych wartości każda, serie opatrzone etykietami „A”, „B”, „C”, „D”, „E”, „F” i 10 indeksami o wartościach 0, 10, 20, 30, 40, 50, 60, 70, 80, 90):

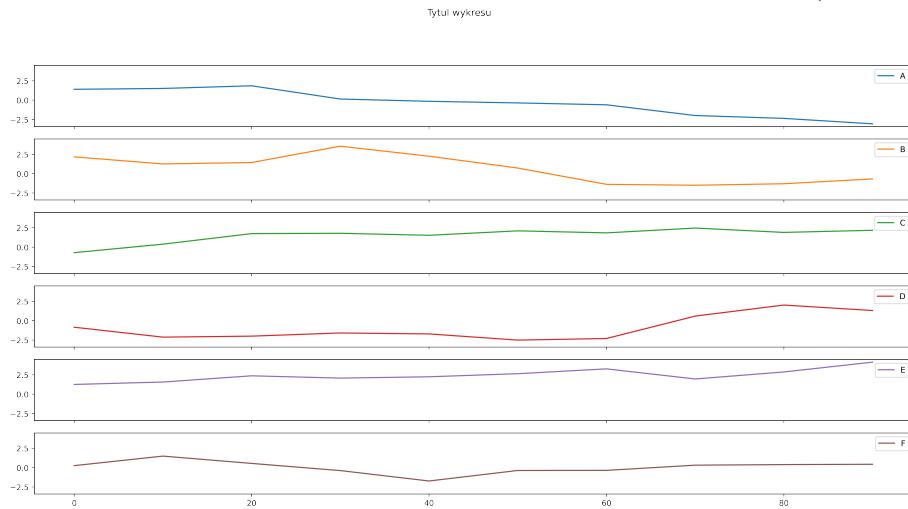
```
1 df = pd.DataFrame(np.random.randn(10, 6).cumsum(0), columns=['A', 'B', 'C',  
   'D', 'E', 'F'], index=np.arange(0, 100, 10))
```

Wygenerowanie wykresu

```
1 df.plot(subplots=True, sharex=True, sharey=True, title='Tytuł wykresu',  
   sort_columns=True)
```

Uzyskany wykres – 9.

Rysunek 9: Kilka wykresów liniowych o wspólnych osiach OX i OY (dane losowe)



Zadanie 10. Wygenerować wykresy słupkowe ułożone poziomo i pionowo

Wyodrębnienie danych (10 pierwszych danych kolumn „wb_location_id” i „gbd_region_id”):

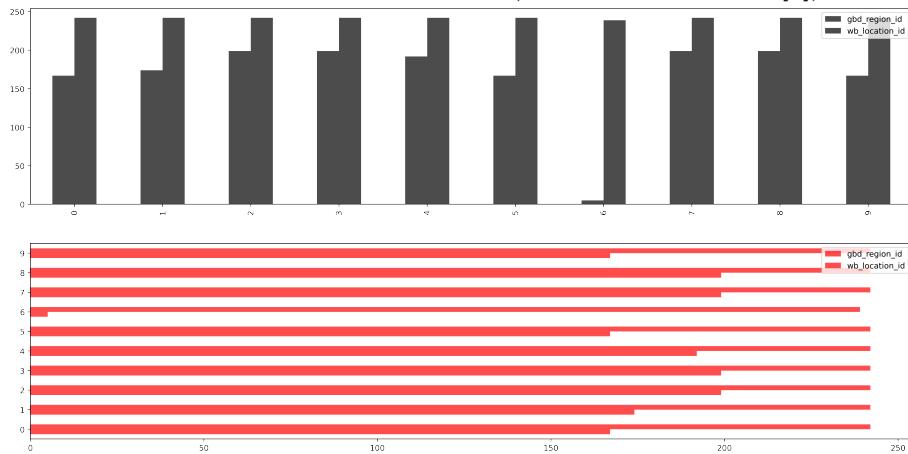
```
1 data = pd.DataFrame(num_data[['wb_location_id', 'gbd_region_id']][:10],  
                      columns=['gbd_region_id', 'wb_location_id'])
```

Wygenerowanie dwóch wykresów (jednego nad drugim – pierwszy, pionowy w kolorze czarnym, drugi, poziomy – czerwonym):

```
1 fig, axes = plt.subplots(2, 1)  
2  
3 data.plot.bar(ax = axes[0], color='k', alpha=0.7)  
4 data.plot.barh(ax = axes[1], color='r', alpha=0.7)
```

Uzyskany wykres – 10.

Rysunek 10: Wykresy słupkowe (dane rzeczywiste za [1])



Zadanie 11. Wygenerować wykres słupkowy z pogrupowanymi danymi

Wygenerowanie losowych danych (5 serii po 3 losowe wartości):

```
1 data = np.random.rand(3, 5)
```

Wygenerowanie wykresu słupkowego o nazwie „Nazwa”. Etykiety serii – „a”, „b”, „c”; etykiety wartości w każdej serii: „1”, „2”, „3”, „4”, „5”:

```
1 df = pd.DataFrame(data, index=['a', 'b', 'c'], columns=pd.Index(['1', '2',  
    '3', '4', '5'], name='Nazwa'))  
2 df.plot.bar()
```

Uzyskany wykres – 11.

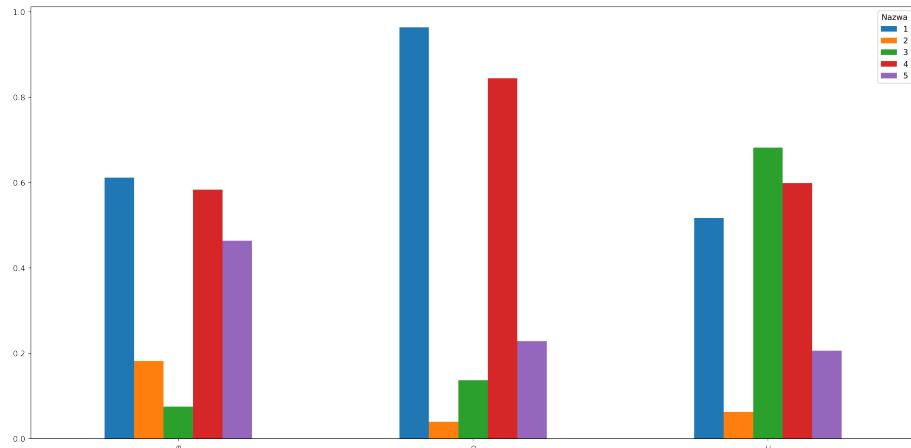
Zadanie 12. Wygenerować wykres słupkowy skumulowany

Wygenerowanie wykresu (dane losowe jak poprzednio):

```
1 df.plot.bar(stacked=True, alpha=0.8)
```

Uzyskany wykres – 12.

Rysunek 11: Wykresy słupkowe (dane losowe)



Zadanie 13. Wygenerować wykres słupkowy przy użyciu biblioteki *seaborn*

Zimportowanie biblioteki i wygenerowanie wykresu dla kolumn „elim_ch” (osi pionowa) i „year” (osi pozioma):

```
1 import seaborn as sns
2
3 max = 100_000
4 per = 1_000
5 sns.barplot(y=num_data.elim_ch[:max:per], x = num_data.year[:max:per])
```

Uzyskany wykres – 13.

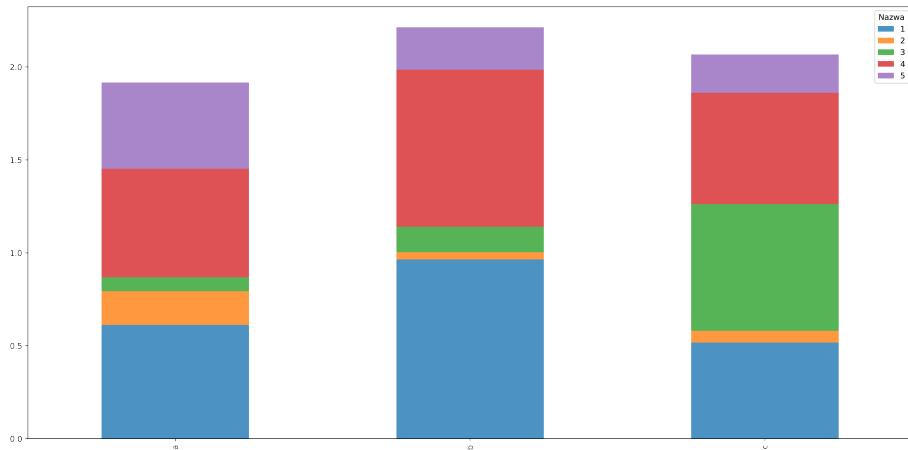
Zadanie 14. Wygenerować histogram

Wygenerowanie histogramu w kolorze zielonym na podstawie danych z kolumny „year” (wartości podzielone na 10 grup):

```
1 num_data['year'].plot.hist(bins=10, color='green')
```

Uzyskany wykres – 14.

Rysunek 12: Wykres słupkowy skumulowany (dane losowe)



Zadanie 15. Wygenerować wykres gęstości prawdopodobieństwa i nałożyć go na histogram

Wykresy wygenerowanie przy użyciu biblioteki „pandas” (wykres gęstości w kolorze czerwonym podpisany etykietą „Density”):

```
1 # wygenerowanie histogramu
2 num_data['year'].plot.hist(bins=10, color='green', alpha=0.2)
3
4 # wygenerowanie wykresu gestosci
5 ax = num_data['year'].plot.kde(color='red', linewidth=2.0, secondary_y=True
   )
6
7 # dodanie etykiety do dodatkowego wykresu
8 ax.set_ylabel("Frequency", fontsize=10)
```

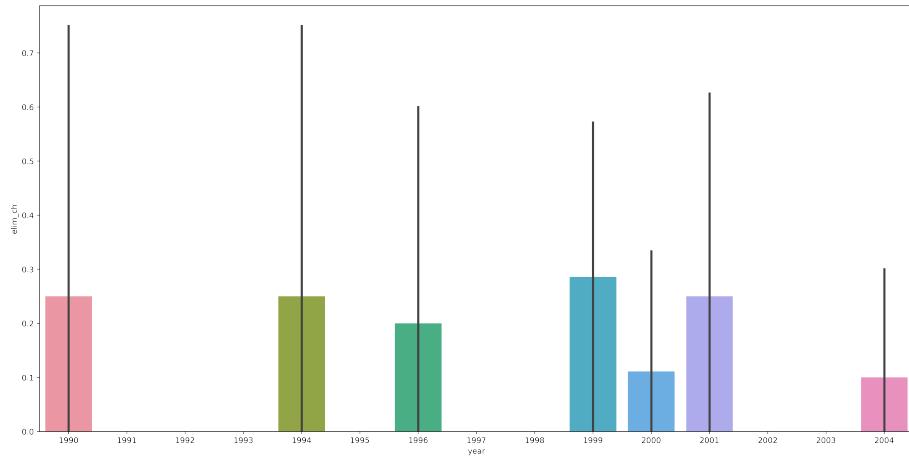
Uzyskany wykres – 15.

Taki sam wykres uzyskany przy użyciu biblioteki „seaborn”:

```
1 sns.distplot(num_data['year'], bins=10, color='r')
```

Uzyskany wykres – 16.

Rysunek 13: Wykres słupkowy wygenerowany przy użyciu biblioteki `seaborn` z zaznaczeniem 95-proc. przedziału ufności (dane rzeczywiste za [1])



Zadanie 16. Wygenerować wykres regresji liniowej

Wygenerowanie losowych danych:

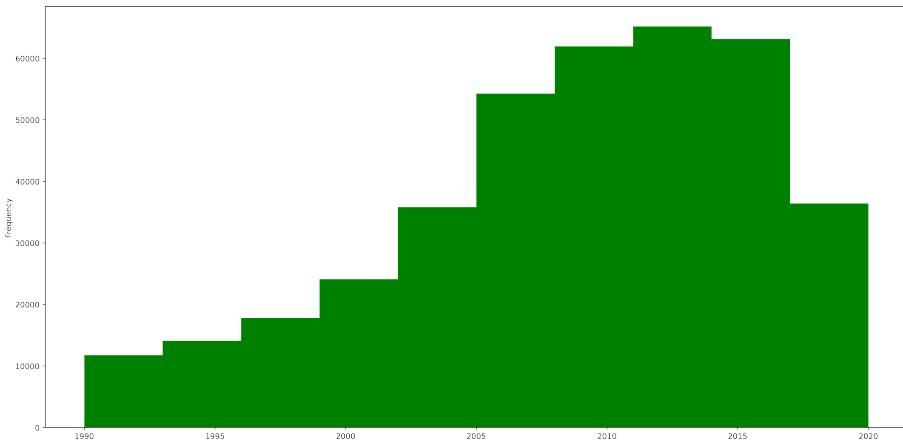
```
1 n = 1_000
2 data = pd.DataFrame({
3     'x': np.random.rand(n)*5-2,
4     'y': np.random.rand(n)*2+5
5 })
6 # roznica logarytmow
7 data = np.log(data).diff()
```

Wygenerowanie wykresu i dodanie tytułu:

```
1 sns.regplot(x='x', y='y', data=data)
2 plt.title('Zaleznosc $\log{x}$ od $\log{y}$')
```

Uzyskany wykres – 17.

Rysunek 14: Histogram (dane rzeczywiste za [1])



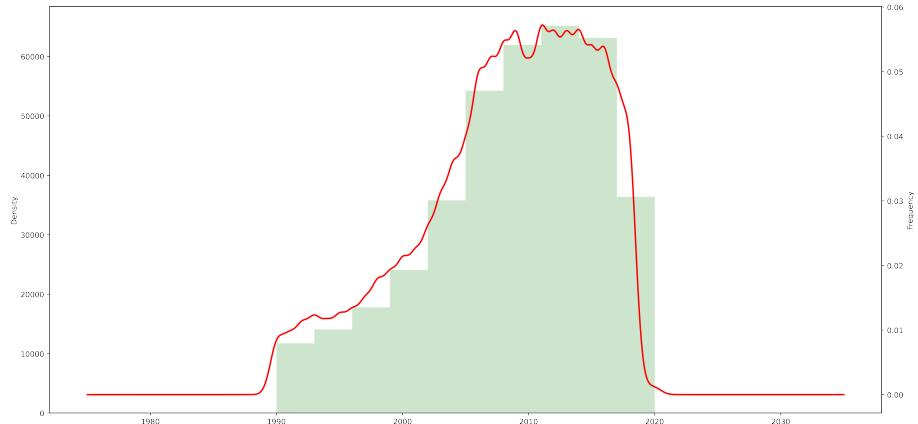
Zadanie 17. Wygenerować wszystkie możliwe wykresy rozrzutu oraz wykresy gęstości

Wygenerowanie wykresu (użyto tych samych danych co w poprzednim zadaniu – zbiory „x”, „y” i „z”):

```
1 sns.pairplot(data, diag_kind='kde', plot_kws={'alpha': 0.2})
```

Uzyskany wykres – 18.

Rysunek 15: Wykres gęstości prawdopodobieństwa nałożony na histogram (dane rzeczywiste za [1])



Zadanie 18. Wygenerować wykres słupkowy z uwzględnieniem danych kategorycznych

Pobranie danych i wstępna eksploracja polegająca na ustaleniu, jakie wartości kolumn „year” i „source” pojawiają się dla „prelim_est = 0” oraz „prelim_est = 1”:

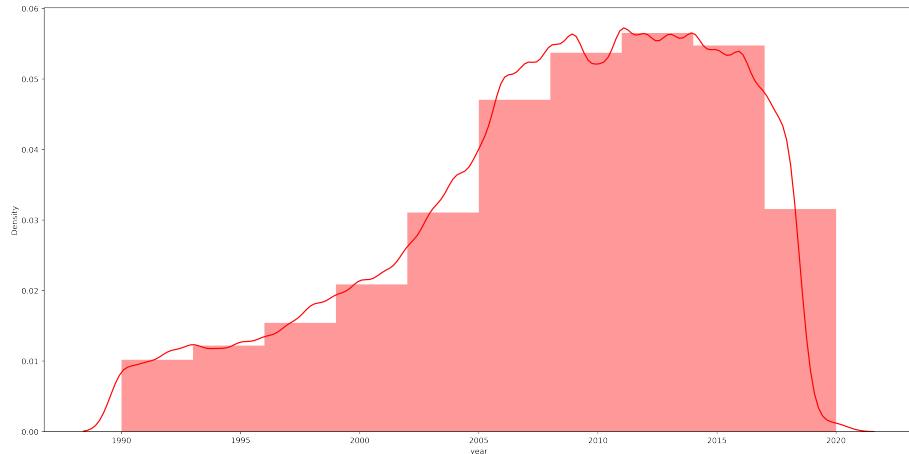
```

1 path = r"C:\Users\piotr\Downloads\EiWD_lab_zadanie02\
      IHME_DAH_DATABASE_1990_2020_Y2021M09D22.CSV"
2
3 data = pd.read_csv(path, low_memory=False)
4
5 print(data.source.drop_duplicates())
6 print(data[data['prelim_est'] == 1].source.drop_duplicates()) # Australia,
      Austra, Belgium, Canada, Denmark, Finland, Germany, Greece...
7 print(data[data['prelim_est'] == 1].year.drop_duplicates()) # 1990, 1991,
      2017, 2018, 2019, 2020
8
9 print(data.year.drop_duplicates())
10 print(data[data['prelim_est'] == 0].source.drop_duplicates()) # Australia,
      Austria, Belgium, Canada, China, Denmark, Finland, France, Germany,
      Greece...
11 print(data[data['prelim_est'] == 0].year.drop_duplicates()) # 1990-2020

```

Wygenerowanie wykresów słupkowych dla każdej wartości w kolumnie „prelim_est” oraz wybranych państw (kolumna „source” równa „Australia”, „Belgium”, „Denmark”, „Finland”, „Germany”) i lat (kolumna „year” równa „2017”, „2018”, „2019” i „2020”):

Rysunek 16: Wykres gęstości prawdopodobieństwa i histogram uzyskany przy użyciu biblioteki „seaborn” (dane rzeczywiste za [1])



```
1 data = data[(data.year.isin([2017, 2018, 2019, 2020]) & data.source.isin(['Australia', 'Belgium', 'Denmark', 'Finland', 'Germany']))]  
2  
3 sns.catplot(x='source', y='elim_ch', hue='year', col='prelim_est', kind='bar', data=data)
```

Uzyskany wykres – 19.

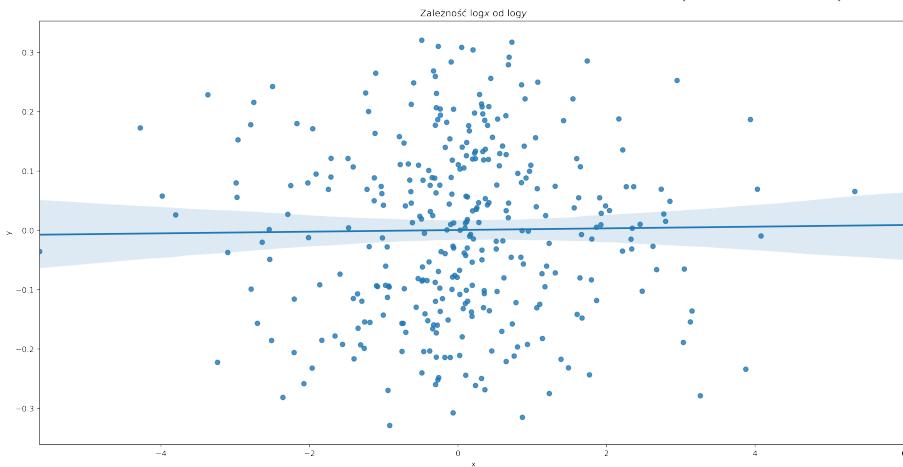
Zadanie 19. Wygenerować wykresy słupkowy uwzględniające dwie kolumny danych kategorycznych

Dane jak w poprzednim zadaniu. W kodzie generującym wykres znajduje się dodatkowy parametr „row” wskazujący na drugą kolumnę kategoryczną „gbd_region”, z której wybrano 3 wartości: „Asia, South”, „Asia, East” oraz „Asia, Central”:

```
1 sns.catplot(x='source', y='elim_ch', hue='year', row='gbd_region', col='prelim_est', kind='bar', data=data[data.gbd_region.isin(['Asia, South', 'Asia, East', 'Asia, Central'])])
```

Uzyskany wykres – 20.

Rysunek 17: Wykres rozrzutu i regresji liniowej (dane losowe)



Zadanie 20. Wygenerować wykres pudełkowy

Dodanie kolumny „swap_hss_total_dah_20_num” zawierającej zero lub wartość logarytmu przy podstawie 10 z wartości w kolumnie „swap_hss_total_dah_20” przekonwertowanej do liczby całkowitej i powiększonej o 1:

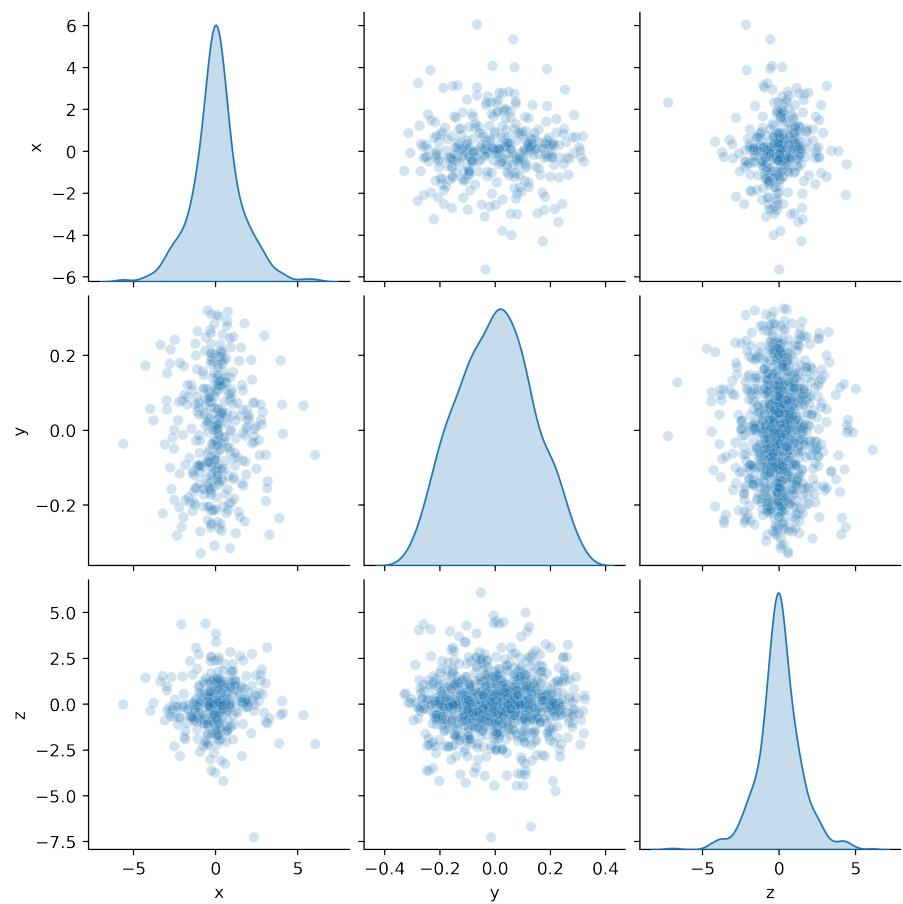
```
1 import math
2
3 data['swap_hss_total_dah_20_num'] = data.swap_hss_total_dah_20.apply(lambda
    x: 0 if x=='-' else math.log(abs(int(x)+1),10))
```

Wygenerowanie wykresu pudełkowego dla wartości w kolumnie „swap_hss_total_dah_20_num” pogrupowanych względem wartości w kolumnie „source”:

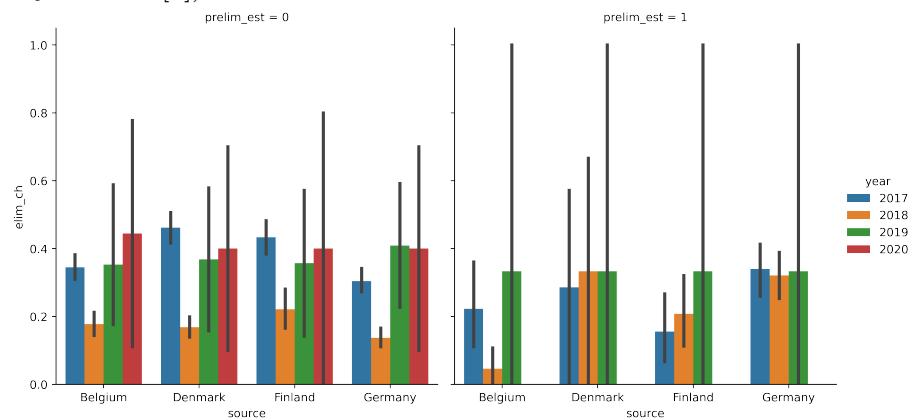
```
1 sns.catplot(x='source', y='swap_hss_total_dah_20_num', kind='box', data=
    data)
```

Uzyskany wykres – 21.

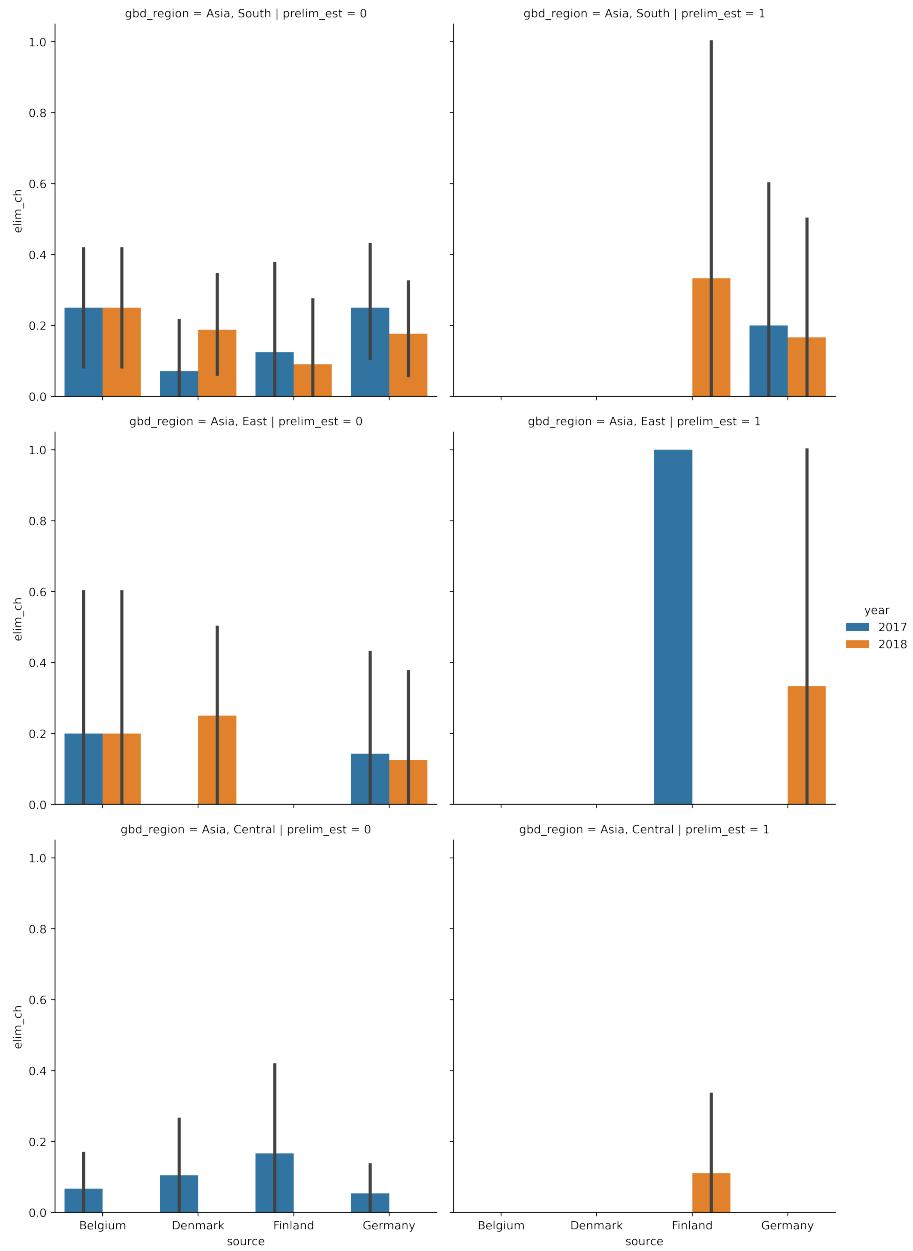
Rysunek 18: Wykresy rozrzutu i gęstości prawdopodobieństwa dla różnych zestawień danych (dane losowe)



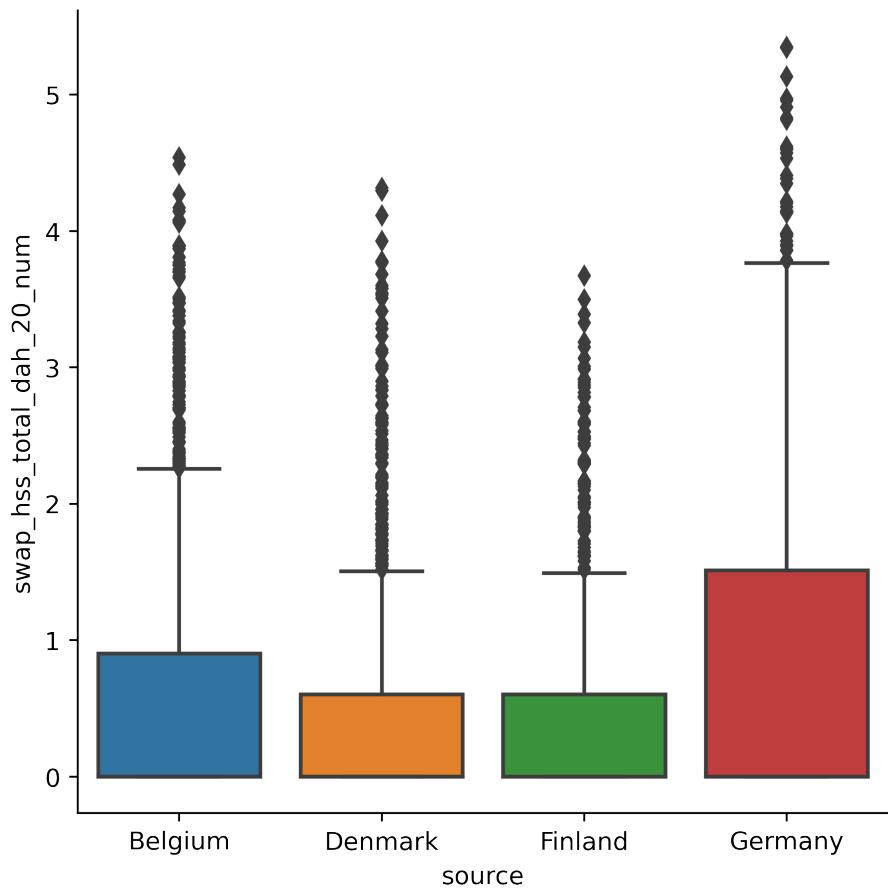
Rysunek 19: Wykres słupkowy uwzględniający dane kategoryczne z kolumny „prelim_est” (dane rzeczywiste za [1])



Rysunek 20: Wykres słupkowy uwzględniający dane kategoryczne z kolumn „gbd_region” i „prelim_est” (dane rzeczywiste za [1])



Rysunek 21: Wykres pudelkowy wartości w kolumnie „swap_hss_total_dah_20_num” pogrupowanych względem wartości w kolumnie „source” (dane rzeczywiste za [1])



Wnioski

Biblioteki „matplotlib” i „seaborn” pozwalają generować następujące rodzaje wykresów:

- wykresy liniowe:

```
1 import matplotlib.pyplot as plt  
2  
3 plt.plot(ramka_danych)
```

- wykresy liniowe: `ramka_danych.plot(kind='line', 'k--')`,
- wykresy słupkowe pionowe: `ramka_danych.plot.bar(color = 'k')`,
- wykresy słupkowe poziome: `ramka_danych.plot.bah(color = 'r')`,
- wykresy słupkowe skumulowane: `ramka_danych.plot.bar(stacked=True)`,
- wykresy słupkowe z przedziałami ufności (biblioteka „seaborn”):
`seaborn.barplot(y=ramka.daneA, x=ramka.daneB),`
- wykresy słupkowe z danymi kategorycznymi:

```
1 # jedna kategoria -- kolumnaD  
2 seaborn.catplot(x='kolumnaA', y='kolumnaB', kind='bar',  
3 hue='kolumnaC_podkategorie_odrozniane_kolorem',  
4 col='kolumnaD_podkategorie_na_podwykresach',  
5 data=ramka_danych)  
6  
7 # dwie kategorie -- kolumnaD i kolumnaE  
8 seaborn.catplot(x='kolumnaA', y='kolumnaB', kind='bar',  
9 hue='kolumnaC',  
10 col='kolumnaD',  
11 row='kolumnaE',  
12 data=ramka_danych)
```

- histogramy: `ramka_danych.plot.hist(bins=10)`,
- wykresy gęstości prawdopodobieństwa: `ramka_danych.plot.kde(color = 'g')`,

- histogram z nałożonym wykresem gęstości prawdopodobieństwa:

```

1 # biblioteka pandas
2 ramka_danych.plot.hist(bins=10)
3 podwykres = ramka_danych.plot.kde(secondary_y=True)
4 podwykres = set_ylabel("Etykieta wykresu gestosci", fontsize=10)
5
6 # biblioteka seaborn
7 seaborn.distplot(ramka_danych, bins=10)

```

- wykresy regresji liniowej: `seaborn.regplot(x='kolumna_a_ramki', y='kolumna_b_ramki', data=ramka_danych)`,
- wykresy rozrzutu: `seaborn.pairplot(ramka_danych, diag_kind='kde')`,
- wykresy pudełkowe: `seaborn.catplot(kind='box', x='kolumnaA', y='kolumnaB', data=ramka_danych)`

Parametry pozwalające formatować wykres:

- biblioteka `matplotlib.pyplot` funkcja `plot()`:
 - kolor wykresu: `plt.plot(color='brown')`, `plt.plot(color='k')`,
 - przeźroczystość wykresu: `plt.plot(alpha=0.1)`
 - kształt punktorów: `plt.plot(marker='o')`, `plt.plot(marker='.'`),
 - wielkość punktorów: `plt.plot(markersize=22)`,
 - rodzaj linii: `plt.plot(linestyle='--')`, `plt.plot(linestyle='.'`),
`plt.plot(linestyle='---')`,
 - szerokość linii: `plt.plot(linewidth=1)`,
 - kolor i rodzaj linii jednocześnie: `plt.plot(dane, 'k--')`,
 - etykieta wykresu: `plt.plot(label='Etykieta')`,
 - położenie legendy (etykiet): `plt.legend(loc='best')`, `plt.legend(loc='upper left')`, `plt.legend(loc='center right')`,
 - położenie etykiet na osiach: `plt.set_xticks([0, 100, 200])`, `plt.set_yticks([0, 100, 200])`,
 - etykiety osi: `plt.set_xticklabels(['a', 'b', 'c'])`, `rotation=90`, `font-size='small'`),
 - tytuł wykresu: `plt.plot(title='Tytul')`,
 - dodanie etykiet przy użyciu słownika:

```

1         etykiety = {
2             'title': 'Tytul',
3             'xlabel': 'Etykiety osi OX',
4             'ylabel': 'Etykiety osi OY'
5         }
6
7         plt.set(**etykiety)
8

```

- wartości w skali logarytmicznej: `ramka_danych.plot(logy=True)`,
- siatka: `ramka_danych.plot(grid=True)`,
- zakres osi: `ramka_danych.plot(xlim=[0, 100])`;

Generowanie kilku wykresów na jednym rysunku:

```

1 import matplotlib.pyplot as plt
2
3 # funkcja figure i kilka ramek danych:
4 ilustracja = plt.figure()
5 wykres1 = ilustracja.add_subplot(2,2,1)
6 wykres2 = ilustracja.add_subplot(2,2,2)
7 wykres3 = ilustracja.add_subplot(2,2,3)
8
9 wykres1.plot(ramka_danych1)
10 wykres2.plot(ramka_danych2)
11 wykres3.plot(ramka_danych3)
12
13 # funkcja plot z parametrem subplots i jedna ramka danych, wspolna os OX:
14 ramka_danych.plot(subplots=True, sharex=True, sharey=False, title='Tytul',
                      sort_columns=True)

```

Zapis do pliku:

```

1 plt.savefig(sciezka_pliku, dpi=300, bbox_inches='tight')

```

Strony internetowe

- [1] *Development Assistance for Health Database 1990-2020.* URL: <http://ghdx.healthdata.org/record/ihme-data/development-assistance-health-database-1990-2020> (term. wiz. 21.10.2021).