Wykład 5. Wizualizacja danych

Michał Sochański

Plan wykładu

1. Wstępne uwagi o wizualizacji danych.
2. Wstęp do pakietu Matplotlib.
3. Wizualizacje jednej zmiennej.
4. Wizualizacje dwóch (i więcej) zmiennych.
5. Uwagi o najczęstszych błędach popełnianych przy wizualizowaniu danych.

1. Wstępne uwagi o wizualizacji danych. Znaczenie wizualizacje danych

- Graficzna prezentacja danych jest bardzo ważnym elementem analizy danych – zarówno jako narzędzie eksploracji danych, jak również przy prezentacji wyników analiz.
- Wizualizacje danych są używane m.in. w nauce, biznesie czy ekonomii – wszędzie, gdzie zachodzi jakakolwiek potrzeba podsumowania i zrozumienia danych.
- Wizualizacja (w szczególności wykres) pozwala na zobaczenie tego, co trudno jest dostrzec w tabeli wypełnionej liczbami. Jest to szczególnie ważne, kiedy mamy do czynienia z ogromną ilością danych i pilną potrzebą ich analizy, zrozumienia i zastosowania (jak to często ma miejsce w dzisiejszych czasach).

Cele wizualizacji danych

Najogólniej, celem wizualizacji jest pogłębianie zrozumienia struktury zbioru danych. W szczególności są to:

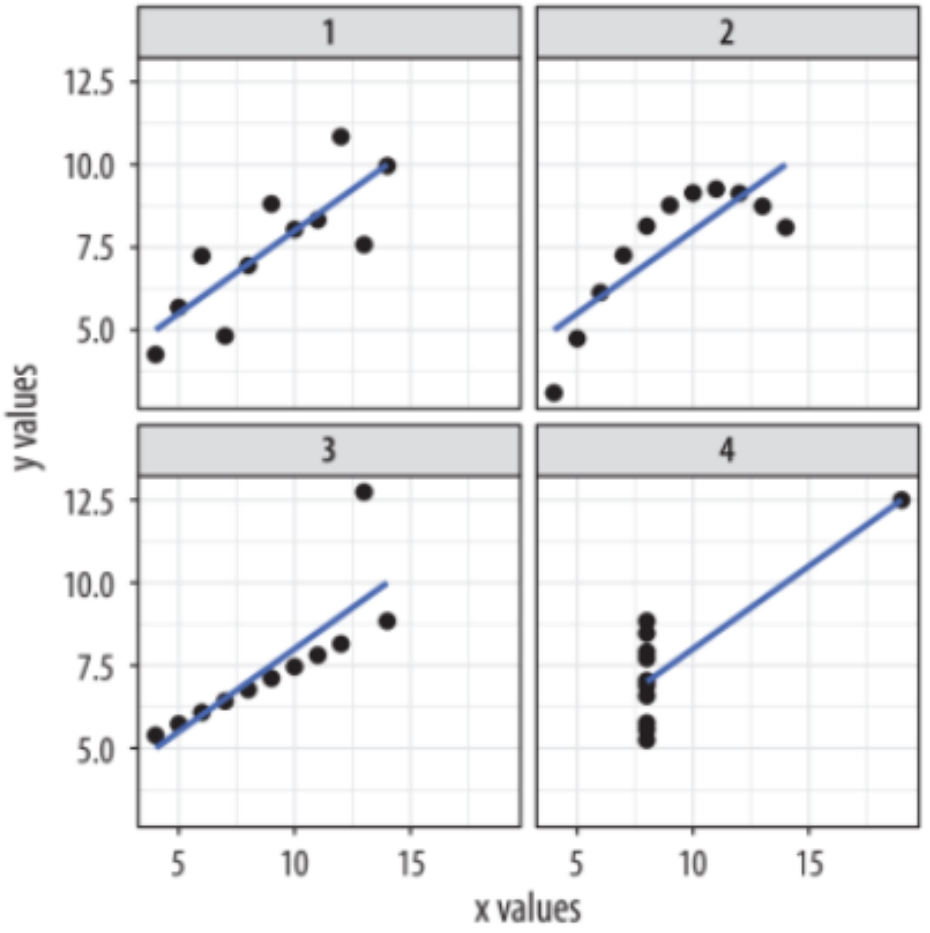
- Szersza (niż korzystająca tylko z podsumowań liczbowych) analiza rozkładów zmiennych.
- Badanie wzajemnych relacji pomiędzy zmiennymi.
- Identyfikacja ciekawych podzbiorów obserwacji.
- Odnajdowanie wartości odstających (*outliers*).

Można powiedzieć, że celem wizualizacji jest odnajdowanie zależności, które są „ukryte”, albo trudne do zauważenia, jeśli posługujemy się tylko danymi w postaci tabelarycznej i korzystamy tylko z analiz numerycznych.

Przykład

Załączony przykład pokazuje, dlaczego często warto zwizualizować pełny rozkład danych, a nie polegać tylko na podsumowaniach i agregacjach.

Średnia wartości na obu osiach dla każdego z powyższych wykresów jest równa. Także korelacja pomiędzy parami zmiennych jest w każdym przypadku równa 0.81!



Wizualizacje jako narzędzie eksploracji danych bądź prezentacji wyników

- Można wyróżnić dwa główne konteksty, w którym używamy wizualizacji danych: eksploracja i raportowanie.
- Eksploracja to zapoznawanie się ze zbiorem danych na wczesnym etapie jako analizy. Wtedy najczęściej próbujemy różnych opcji, wykonując wiele wizualizacji „na brudno”.
- W przypadku raportowania (w postaci pracy zaliczeniowej, artykułu czy raportu dla szefa), wizualizacja służy do przedstawienia wniosków z naszych analiz w jak najbardziej komunikatywny sposób. Wtedy każda z nich powinna być dopracowana i odpowiednio dobrana – tak, aby jak najlepiej przedstawiała idee, które chcemy przekazać.

Wizualizacje – środki wyrazu

- Są różne wizualne środki, za pomocą których możemy reprezentować dane liczbowe. Wielkości te możemy reprezentować np. za pomocą:
  - Długości odpowiednich elementów (np. prostokątów, linii)
  - Pola figur
  - Wielkości kątów
- Dane liczbowe, jak również jakościowe, mogą być też reprezentowane za pomocą:
  - Kolorów
  - Natężenia
  - Kształtów
- Relacje pomiędzy różnymi zmiennymi, albo jej wartościami reprezentujemy dodatkowo przez:
  - Wzajemne położenie poszczególnych elementów
  - Odległość pomiędzy poszczególnymi elementami
- Wykres może też zawierać dodatkowe pola tekstowe lub znaki, jak np. strzałki, aby uwypuklić odpowiednie jego elementy.

Ogólne cechy dobrej wizualizacji

Ogólnie rzecz biorąc, wizualizacja danych powinna oferować wgląd w dane, tzn. wydobywać ciekawe i istotne informacje dotyczące różnych aspektów danych. Informacje te powinny dodatkowo być przekazane w jasny, łatwy do odczytania sposób.

W ogólności trudno jest podać jednoznaczne zasady postępowania, które zawsze doprowadzą nas do dobrej wizualizacji. Jakość wizualizacji zależy bowiem m.in. od tego, na ile dobrze zrozumieliśmy dane, bądź też jakie informacje udało nam się z tego zbioru „wydobyć” – zależy więc w dużym stopniu od kreatywności. Ponadto, wizualizacja powinna być dostosowana do odbiorcy (poszczególne wizualizacje będą zrozumiałe dla osób, które potrafią je zinterpretować).

Cechy „formalne” dobrej wizualizacji

Można jednak podać ogólne cechy, które dobra wizualizacja powinna zawsze posiadać:

1) Dopasowanie typu wizualizacji do charakteru zmiennych (np. histogram, jeśli chcemy pokazać rozkład zmiennej numerycznej)
2) Czytelność, tzn. możliwość odczytania odpowiednich własności wizualnych (wielkość, kolor) gołym okiem.
3) Unikanie nadmiaru wizualnych środków wyrazu.
4) Wizualizacje powinny oczywiście być też oparte na prawidłowych i dobrze dobranych danych.
5) Poprawne wykonanie wizualizacji, w szczególności: jeśli wizualizujemy dane na wykresie, powinien on mieć podpis (albo ew. tytuł) oraz odpowiednio podpisane osie.

Cechy dobrej wizualizacji – dodatkowe wskazówki

- Wizualizacja może mieć walory estetyczne, chociaż nie powinny one być głównym celem w jej tworzeniu. Czasem mogą przeszkadzać w przekazywaniu informacji.
- Najlepsze wizualizacje zawierają bardzo dużo istotnych danych na jednym obrazku, pokazują ukryte wcześniej dla nas związki i własności oraz pozwalają na ich dłuższe studiowanie, w trakcie których „stopniowo” odkrywamy zależności i wyrabiamy sobie pogląd na strukturę wizualizowanych danych.
- Wizualizacja (tak ja każda analiza czy prezentacja) powinna być wreszcie dostosowana do odbiorcy.
- Uwaga! Dobra wizualizacja danych wzmacnia siłę naszych wyników, źle dobrana wizualizacja potrafi jednak ukryć bądź zniekształcić informacje, które chcemy przekazać.

Wizualizacje a percepcja

- Wiedza o percepcji kształtów, kolorów i przestrzennych relacjach między obiektami wykresu może pomóc nam stworzyć efektywne wizualizacje, tzn. takie, z których można łatwo odczytać informacje, które chcemy przekazać. W szczególności może pomóc w doborze najlepszych środków do przekazania informacji, którą chcemy zakomunikować za pomocą wizualizacji.
- Ważne tutaj są dwa zagadnienia:

1) Które środki wyrazu (kształty, kolory, itd.) pozwalają oku łatwiej i sprawniej wychwycić poszczególne wielkości i relacje pomiędzy nimi?
2) Które przestrzenne własności wykresów skłaniają nas do wyciągania określonych wniosków (czasem nieświadomie) o reprezentowanych danych?

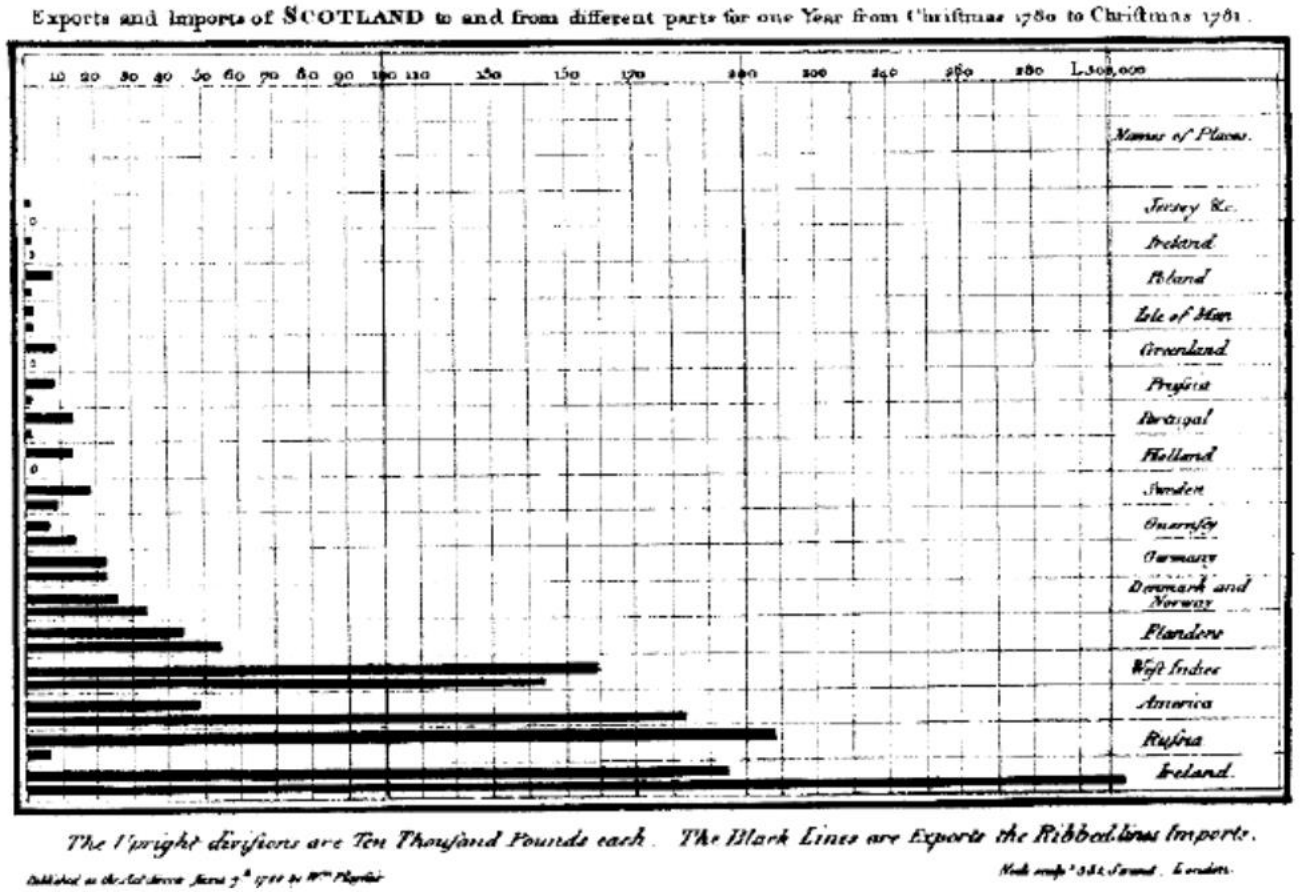
Wizualizacje a percepcja

- Oto wybrane zasady, którymi warto się kierować tworząc wizualizacje:
  - Długość (np. kolumny) lepiej reprezentuje wartości liczbowe niż pole.
  - Kolory są przydatne do reprezentowania zmiennych jakościowych, ale zazwyczaj nie nadają się do prezentowania wartości liczbowych.
  - Należy unikać wykresów 3-wymiarowych – są trudne w odczytaniu i zazwyczaj zaciemniają informacje, które chcemy przekazać.
  - Należy unikać „przeładowania” wykresu informacjami, tak aby nasz zmysł wzroku był w stanie odczytać i wyróżnić istotne fakty.
- Więcej np. na stronie http://www.biecek.pl/Eseje/indexHistoria.html

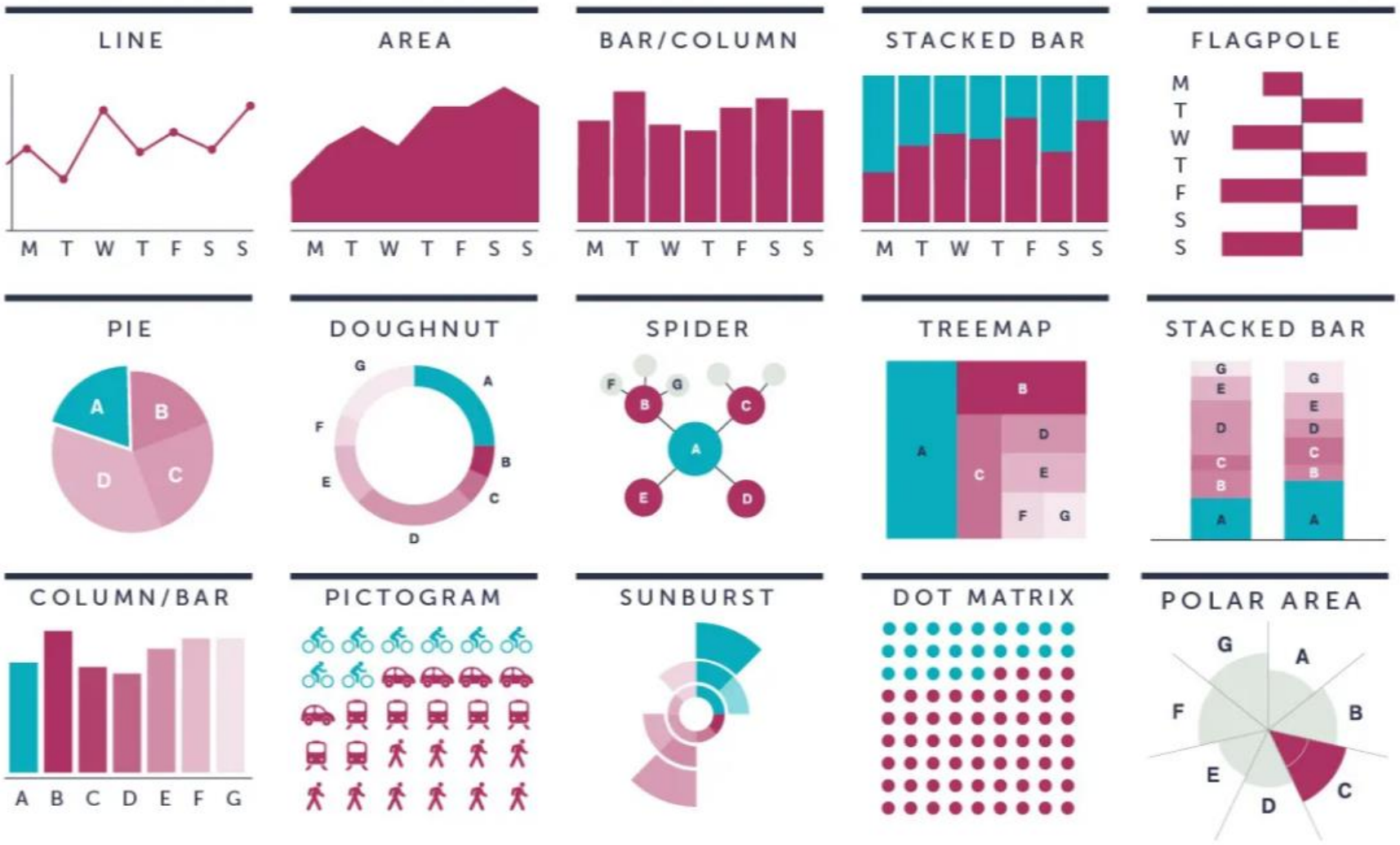
Typy wizualizacji danych

- Istnieje bardzo wiele typów wizualizacji danych. My poznamy najważniejsze, standardowe, typy wizualizacji, od których należy zacząć praktykę wizualizowania danych. Takie standardowe wykresy można też względnie łatwo stworzyć za pomocą dostępnych narzędzi (oczywiście pod warunkiem posiadania odpowiednio przygotowanych danych).
- Jednak istotą wizualizacji danych jest swoboda i kreatywność w doborze środków wyrazu do przekazania odpowiedniej informacji. Aby stworzyć bardziej złożoną wizualizację trzeba jednak często użyć bardziej skomplikowanych narzędzi.

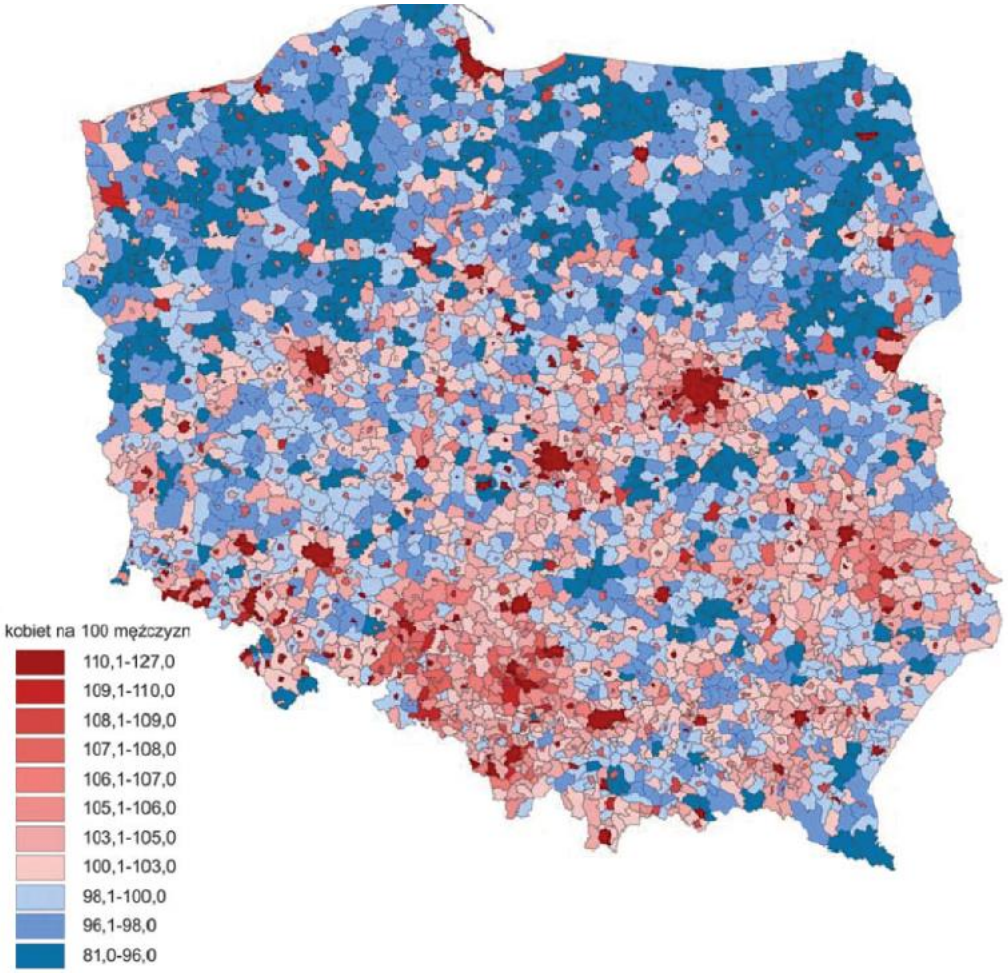
Odrobina historii – pierwszy wykres słupkowy



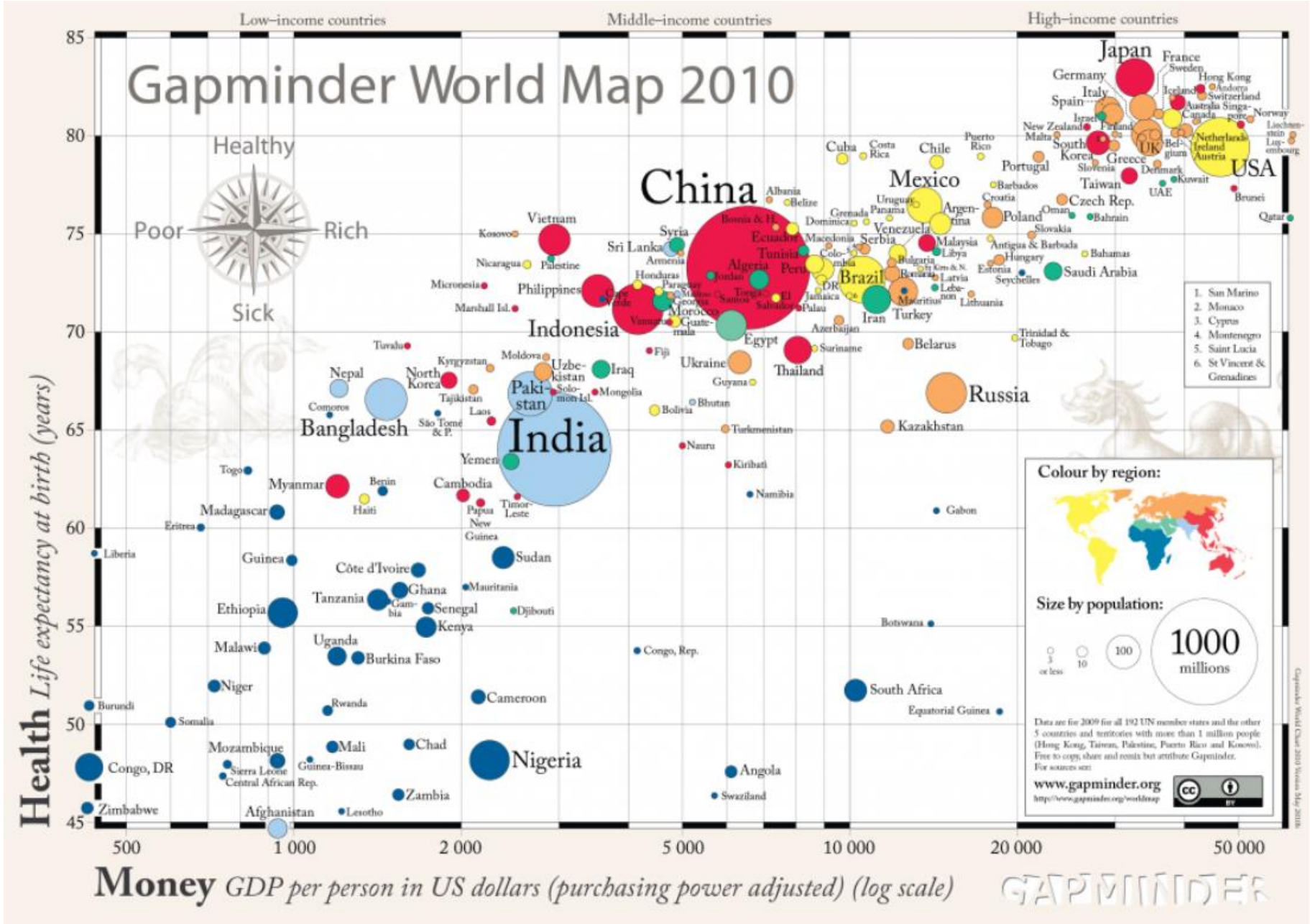
Wykres z *The Commercial and Political Atlas* Williama Playfaira (1786).



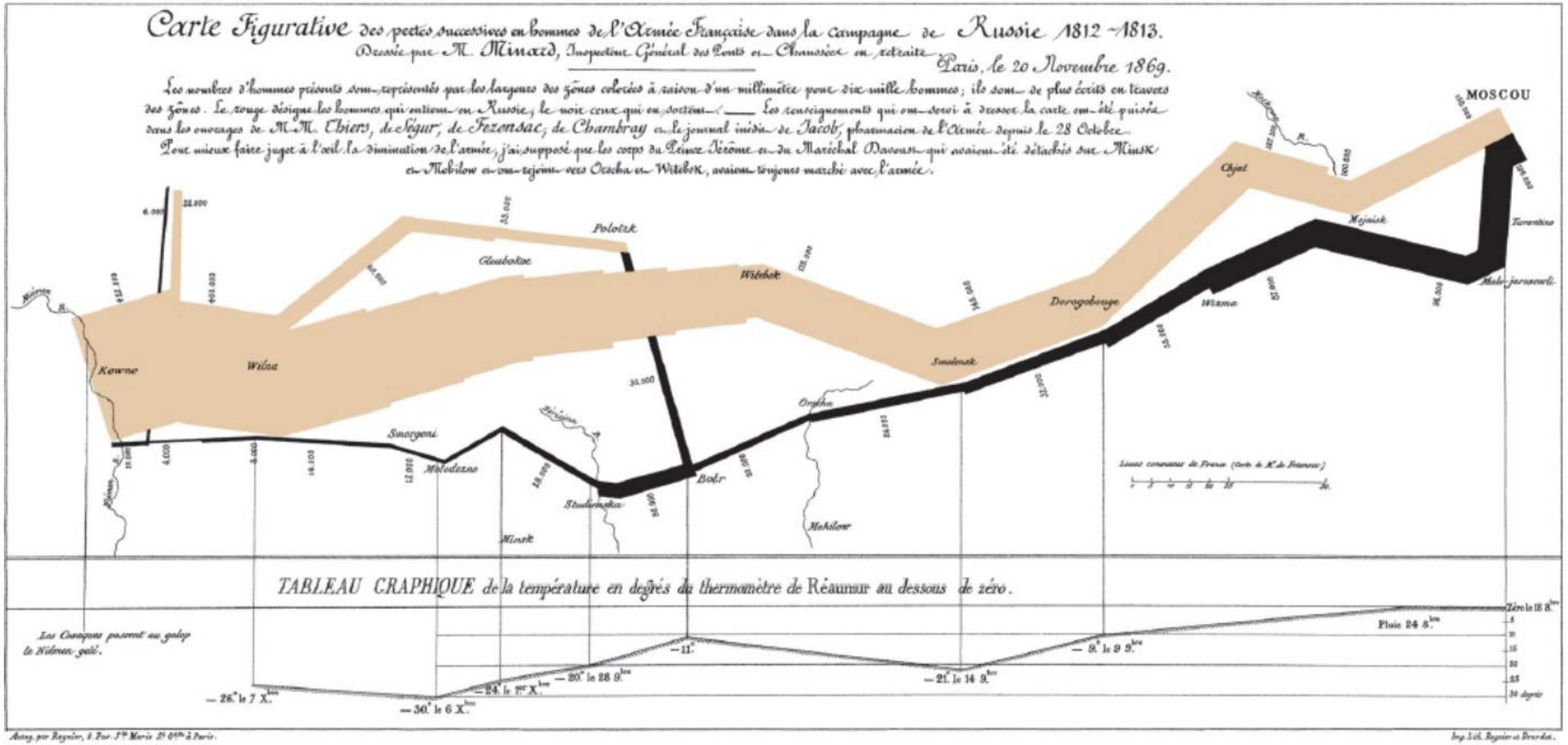
Krótki przegląd wybranych typów wizualizacji (źródło – www.theguardian.com)


kobiet na 100 mężczyzn
110,1-127,0
109,1-110,0
108,1-109,0
107,1-108,0
106,1-107,0
105,1-106,0
103,1-105,0
100,1-103,0
98,1-100,0
96,1-98,0
81,0-96,0

Źródło – GUS



Źródło – www.gapminder.org



Infografika przedstawia losy armii napoleońskiej na różnych etapach inwazji na Rosję. Źródło: Charles Minard *Figurative Map of the successive losses in men of the French Army in the Russian campaign 1812–1813*, rok 1869.

2. Wstęp do pakietu Matplotlib

- Matplotlib to pakiet Pythona, który jest najczęściej używany do wizualizacji danych. Zaimportujemy go używając następującego polecenia.

```
import matplotlib.pyplot as plt
```

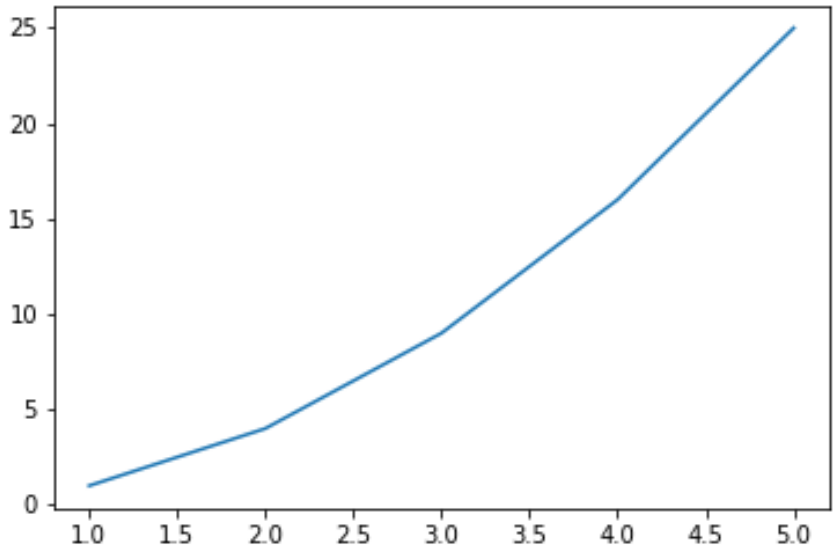
- Ogólne zasady składni są następujące:

1. W pierwszej linijce używamy polecenia tworzącego odpowiedni wykres, podając dane, na podstawie których będzie on tworzony, oraz wybrane przez nas parametry.
2. W kolejnych linijkach możemy dodać nowe elementy do naszego rysunku (np. etykiety, kolory, kolejne wykresy).
3. W ostatniej linijce albo używamy polecenia „plt.show()”, albo zapisujemy wykres do pliku za pomocą polecenia „plt.savefig('wykres.png')” (uwaga – możemy również zapisywać wykresy w innych formatach).

- Uwaga – przed sporządzeniem wykresu należy przygotować odpowiednio dane!

Przykład na wprowadzenie

```
x = [1,2,3,4,5]
y = [1,4,9,16,25]
plt.plot(x,y)
plt.show()
```

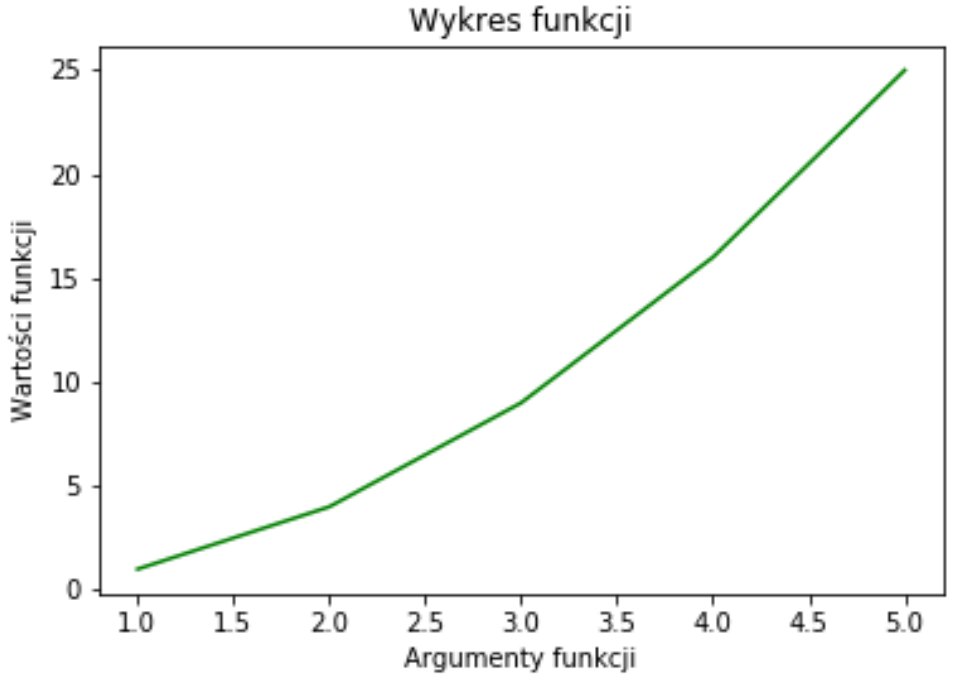


Generowany jest tutaj wykres liniowy – w tym przypadku matplotlib automatycznie łączy za pomocą odcinków punkty (1,1), (2,4), (3,9) itd.

Uwaga! Wprowadzane dane mogą być „zwykłą” listą, ale także w szczególności obiektem typu „pd.Series” a więc kolumną DataFrame’u.

Podpisujemy osie, dodajemy tytuł wykresu i kolor linii

```
plt.plot(x, y,  color='g')
plt.xlabel('Argumenty funkcji')
plt.ylabel('Wartości funkcji')
plt.title('Wykres funkcji')
plt.show()
```



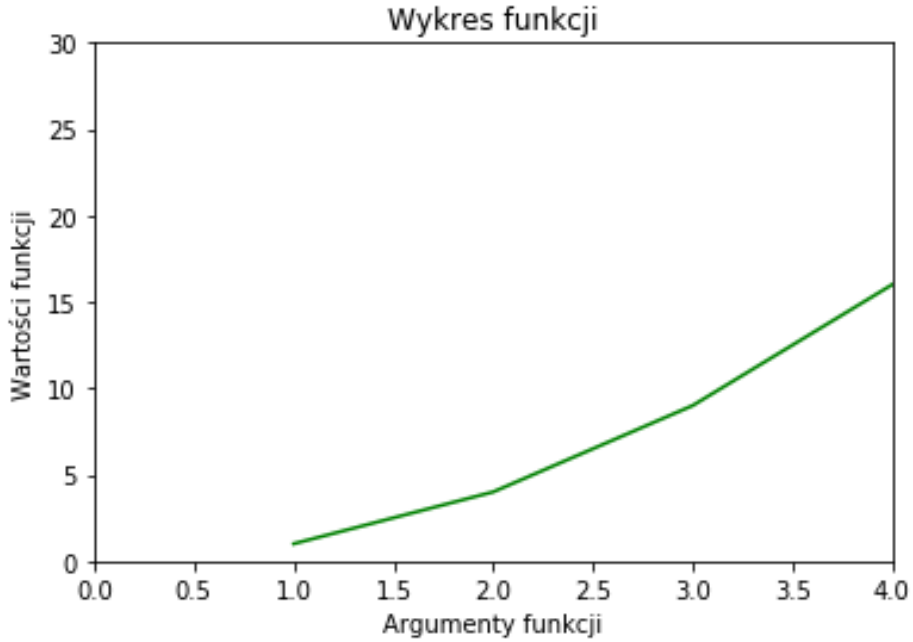
Dygresja - kolory

- b: blue
- g: green
- r: red
- c: cyan
- m: magenta
- y: yellow
- k: black
- w: white

Kolor można ustawić za pomocą znaczników znanych z HTMLa: color = ’#eeefff’, lub skorzystać z palety RGB: color=[1, 0.9, 0.01], gdzie kolory: czerwony, zielony i niebieski ustawiamy w zakresie od 0 do 1.

Zmiana zakresu liczb wyświetlanych na osiach

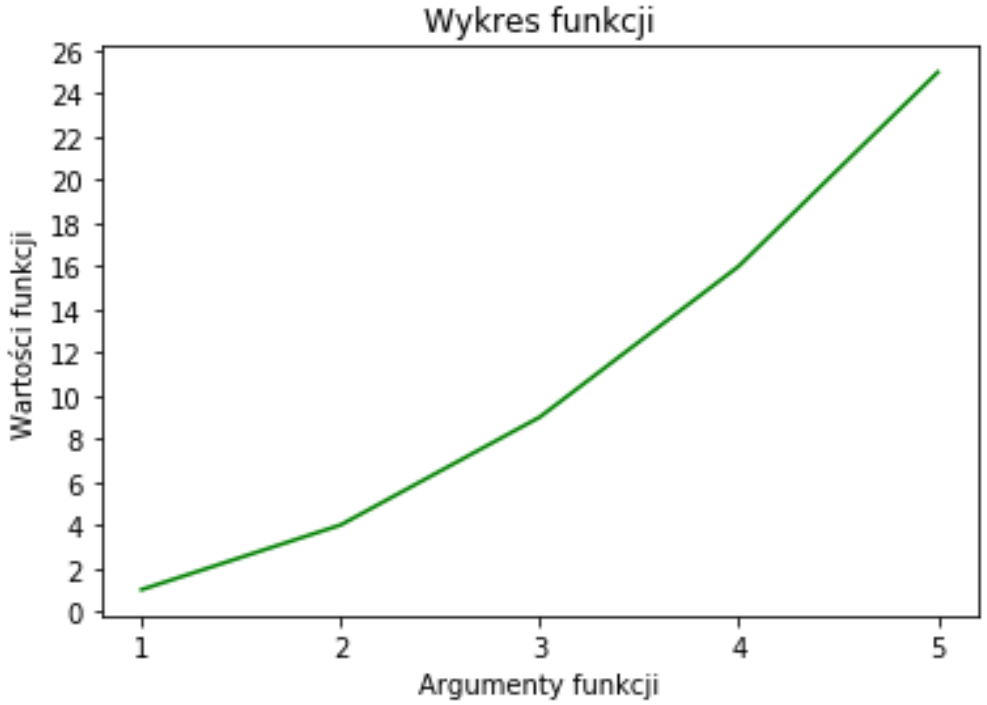
```
plt.plot(x, y,  color='g')
plt.xlabel('Argumenty funkcji')
plt.ylabel('Wartości funkcji')
plt.xlim(0,4)
plt.ylim(0,30)
plt.title('Wykres funkcji')
plt.show()
```



Uwaga – można tu też zastosować polecenie: plt.axis([0,4,0,30])

Ustalanie znaczników osi

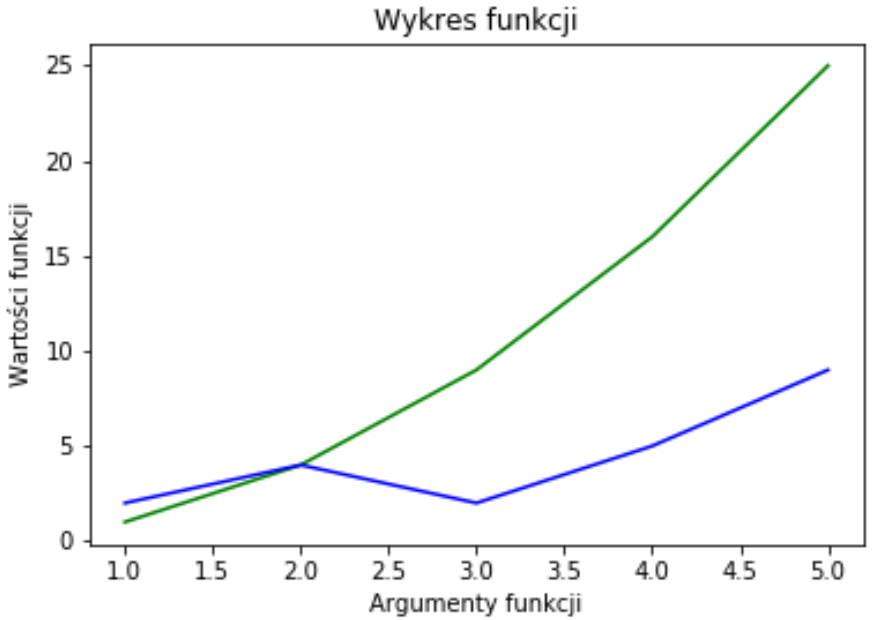
```
plt.plot(x, y,  color='g')
plt.xlabel('Argumenty funkcji')
plt.ylabel('Wartości funkcji')
plt.yticks(range(0,28,2))
plt.xticks(range(1,6,1))
plt.title('Wykres funkcji')
plt.show()
```



Uwaga – range(0,28,2) oznacza: zaczynamy od wartości 0, kończymy na 26, różnica między kolejnymi wartościami równa jest 2.

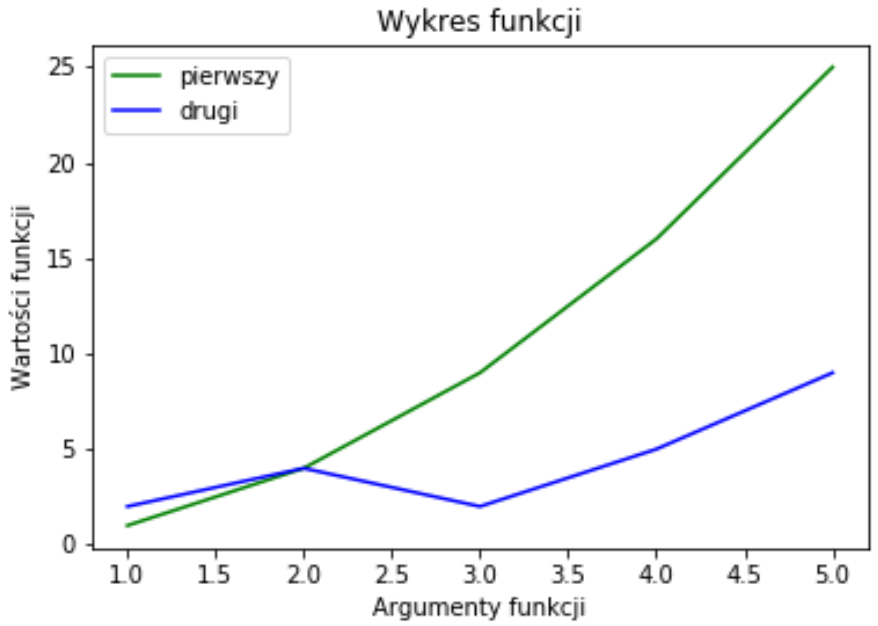
Dodajemy do rysunku drugi wykres

```
plt.plot(x, y,  color='g')
plt.plot([1,2,3,4,5], [2,4,2,5,9], color='b')
plt.xlabel('Argumenty funkcji')
plt.ylabel('Wartości funkcji')
plt.title('Wykres funkcji')
plt.show()
```



Dodajemy etykiety dla wykresów i legendę

```
plt.plot(x, y,  color='g', label='pierwszy')
plt.plot([1,2,3,4,5], [2,4,2,5,9], color='b', label='drugi')
plt.xlabel('Argumenty funkcji')
plt.ylabel('Wartości funkcji')
plt.title('Wykres funkcji')
plt.legend()
plt.show()
```

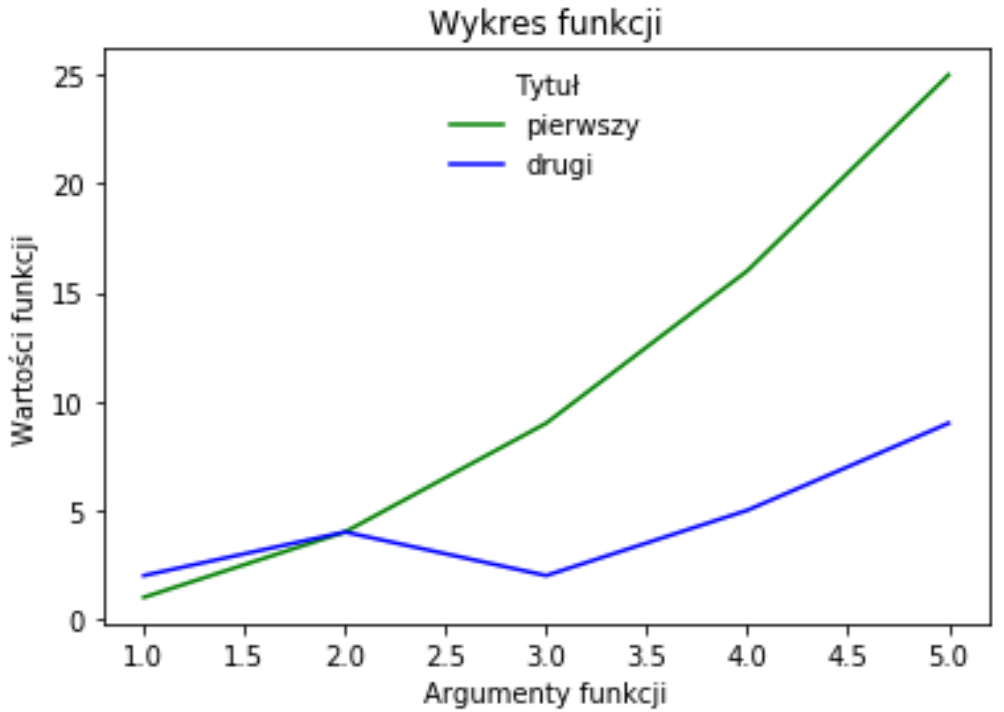


Legenda – opcje dodatkowe

Poniżej podajemy przykładowe dodatkowe opcje związane z legendą.

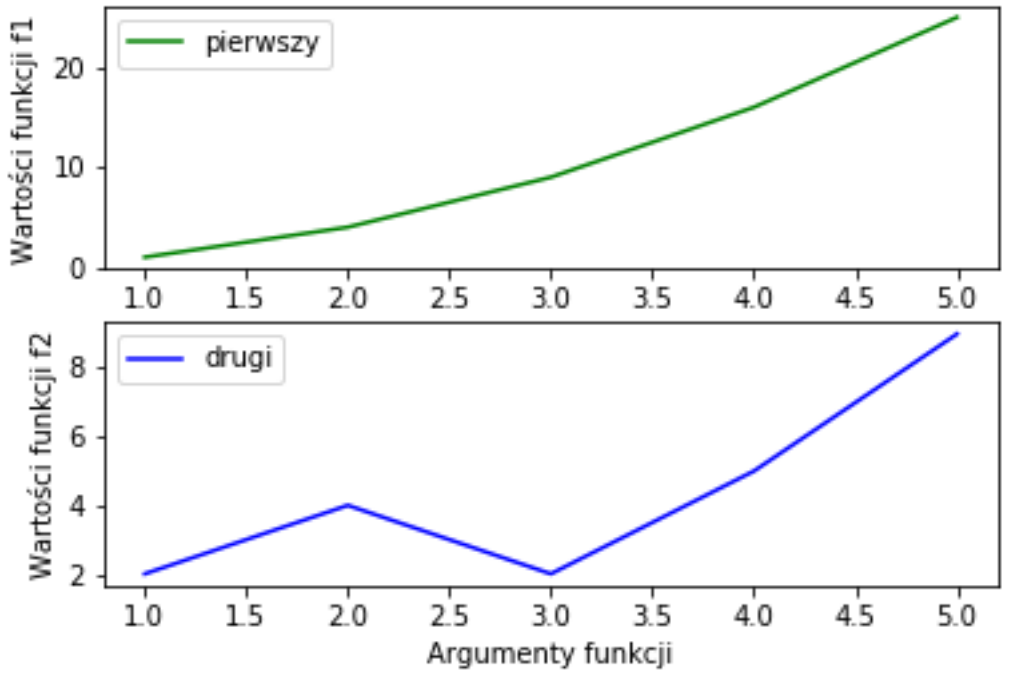
Uwaga – w legendzie pojawią się tylko odniesienia do tych linii, dla których określony został argument „label”.

```
plt.plot(x, y,  color='g', label='pierwszy')
plt.plot([1,2,3,4,5], [2,4,2,5,9], color='b', label='drugi')
plt.xlabel('Argumenty funkcji')
plt.ylabel('Wartości funkcji')
plt.title('Wykres funkcji')
plt.legend(frameon=False,
                loc='upper center',
                title='Tytuł')
plt.show()
```



Większa liczba wykresów

```
plt.subplot (211)
plt.plot(x, y,  color='g', label='pierwszy')
plt.ylabel('Wartości funkcji f1')
plt.legend()
plt.subplot (212)
plt.plot([1,2,3,4,5], [2,4,2,5,9],
            color='b', label='drugi')
plt.xlabel('Argumenty funkcji')
plt.ylabel('Wartości funkcji f2')
plt.legend()
plt.show()
```

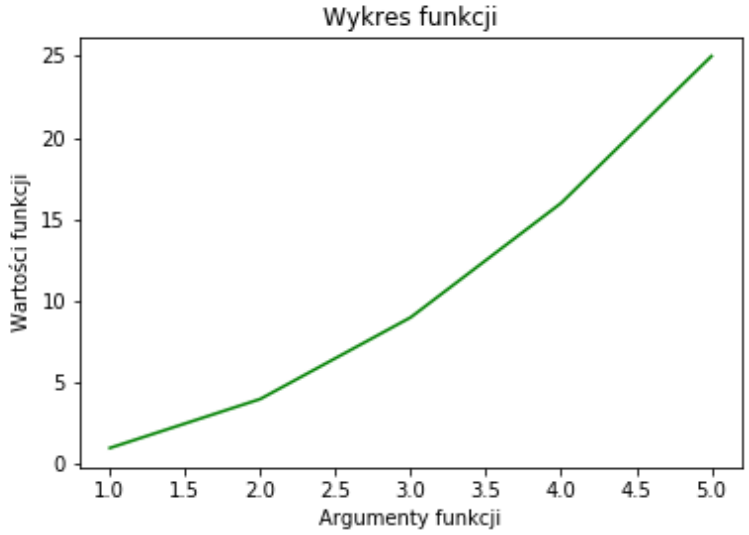


Argumentem funkcji subplot() są trzy liczby. Pierwsza określa liczbę wierszy, a druga liczbę kolumn, w których grupowane będą wykresy. Trzecia określa, który wykres aktualnie opisujemy.

Alternatywna metoda generowania wykresów

- Zaprezentujemy również alternatywną metodę tworzenia wykresów w matplotlib.
- W tym przypadku tworzymy najpierw obiekty fig (cały wykres) i ax (osie), które są z początku są puste.
- Efekt będzie taki sam, jak poprzednio

```
fig = plt.figure()  #w obiekcie fig zapisany będzie nasz wykres
ax = plt.axes()     #obiekt ax zawiera oś (na początku pusta)
ax.plot(x, y,  color='g')   #używamy funkcji „ax.plot”
ax.set_xlabel('Argumenty funkcji')
ax.set_ylabel('Wartości funkcji')
ax.set_title('Wykres funkcji')
plt.show()
```



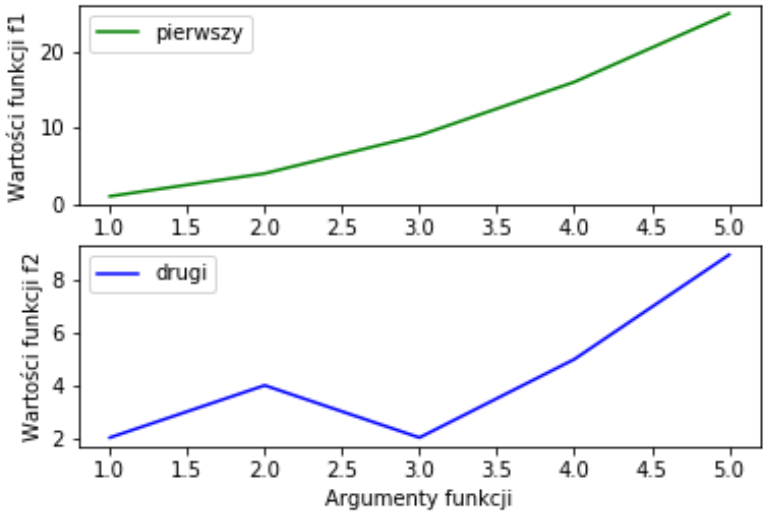
Alternatywna metoda generowania wykresów

- Tym razem zastosujemy drugą metodę do generowanie większej liczby wykresów (efekt znów jest taki sam jak poprzednio)

```
fig, ax = plt.subplots(2)

ax[0].plot(x, y,  color='g', label='pierwszy')
plt.ylabel('Wartości funkcji f1')
plt.legend()

ax[1].plot([1,2,3,4,5], [2,4,2,5,9], color='b', label='drugi')
plt.xlabel('Argumenty funkcji')
plt.ylabel('Wartości funkcji f2')
plt.legend()
plt.show()
```



Inne opcje dodatkowe

- Zmiana stylu wykresu

plt.style.use()

- Dodatenie elementu tekstowe na polu wykresu

plt.text(), plt.annotate()

- Dodanie strzałki (może być użyta np. razem z tekstem)

plt.arrow()

Zapisujemy nasz wykres do pliku

- Stosując funkcję „plt.figure” przed zdefiniowaniem wykresu możemy zapisać nasz wykres w zmiennej. Poniżej zapisujemy wykres w zmiennej „fig”:

```
fig = plt.figure()
plt.plot([1,2,3,4,5], [2,4,2,5,9], color='b', label='drugi')
plt.show()
```

- Wtedy możemy m.in. zapisać wykres w pliku:

```
fig.savefig('my_figure.png')
```

- Wykresy można zapisywać w różnych formatach, listę wszystkich dostępnych wyświetli następująca komenda:

```
fig.canvas.get_supported_filetypes()
```

3. Wizualizacja jednej zmiennej

3.1. Zmienna ilościowa:

- 3.1.1. Histogram
- 3.1.2. Wykres pudełkowy (box plot)

3.2. Zmienna jakościowa

- 3.2.1. Wykres słupkowy (bar chart)
- 3.2.2. Wykres kołowy (pie chart)

3.1.1. Histogram

- Za pomocą histogramu wizualizujemy dane ilościowe. Histogram pozwala zapoznać się z rozkładem wartości danej cechy. Dokładniej, za pomocą histogramu możemy przekonać się dla ilu obserwacji zmienna ta przyjmuje wartości z określonych przedziałów.
- Elementami histogramu są prostokąty. Każdy prostokąt (słupek) reprezentuje obserwacje, dla których wartość odpowiedniej zmiennej leży w określonym przedziale. Wysokość słupka reprezentuje liczbę obserwacji, które mieszczą się w tym przedziale.

Histogram – przykładowe argumenty

1. Najprostszy przypadek, w którym podajemy tylko zbiór danych.

```
plt.hist(dane)
plt.show()
```

2. Domyślna liczba słupków w Matplotlib to 10, aby zmienić tę wartość należy zastosować argument „bins”. Przy pomocy argumentu „histtype” możemy z kolei określić kształt wykresu (np. wartość „step” oznacza, że widzimy tylko kontury słupków).

```
plt.hist(dane, bins=15, histtype='step')
plt.show()
```

Histogram – przykładowe argumenty (2)

2(cd)

W argumencie „bins” możemy też podać dokładne granice zakresów za pomocą funkcji range, albo listy wartości :

plt.hist(dane, bins=range(0,20,2))

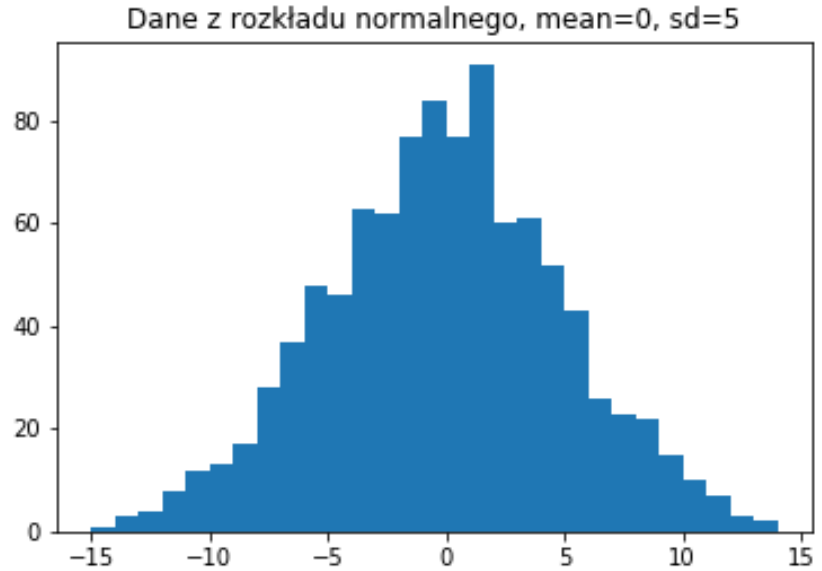
plt.hist(dane, bins=[0,5,10,15,20])

3. Kolor słupków kontrolujemy za pomocą argumentu „color”, a kolor ich konturu za pomocą argumentu „edgecolor”:

plt.hist(dane, color='steelblue', edgecolor='k')

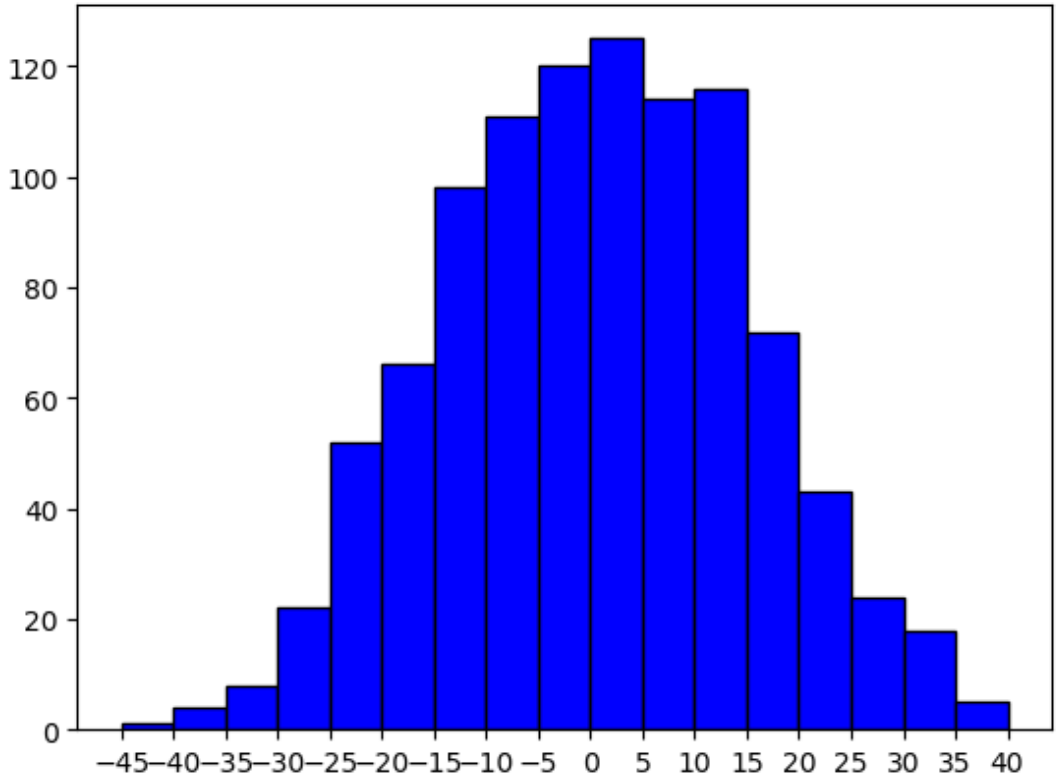
Histogram - przykład

```
dane = np.random.normal(0,5,1000)
plt.hist(dane, bins=range(-15,15))
plt.title('Dane z rozkładu normalnego, mean=0, sd=5')
plt.show()
```



Histogram – przykład (2)

- dane = np.random.normal(0,15,1000)
- plt.hist(dane, bins=range(-45,45,5), color='b',edgecolor='black')
- plt.xticks(range(-45,45,5))
- plt.show()

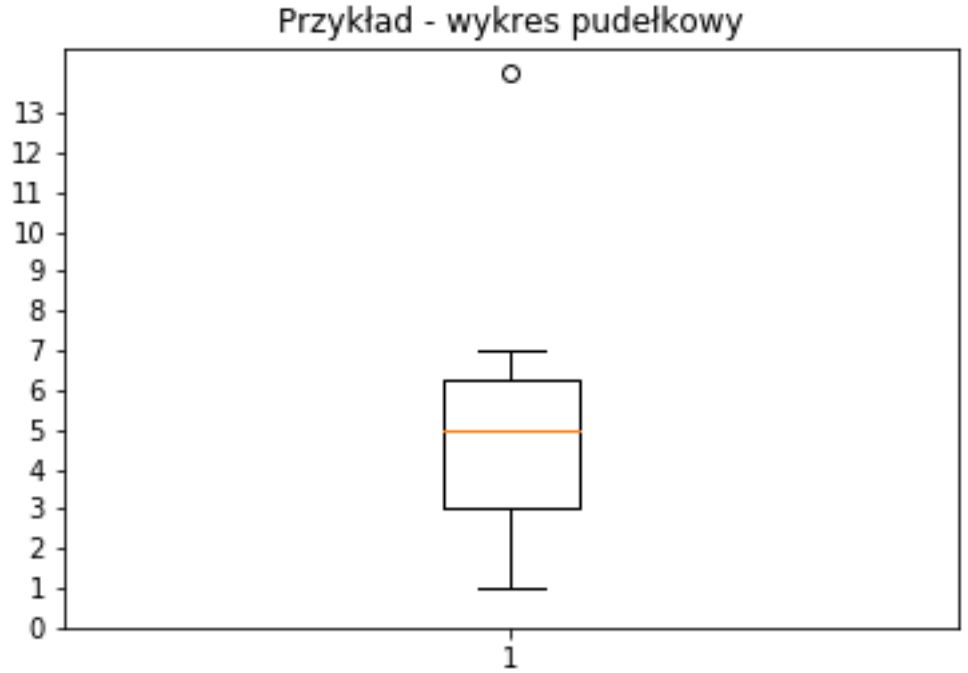


3.1.2.Wykres pudełkowy (*box plot*)

- Za pomocą wykresu pudełkowego wizualizujemy pięć liczb charakteryzujących zestaw danych. Liczby te poznaliśmy na poprzednim wykładzie, są to: minimum, pierwszy kwartyl, mediana, trzeci kwartyl i maksimum. Nazywa się je czasem „piątką Tukeya”, w j. angielskim – „*five-number summary*”. Funkcja tworząca wykres pudełkowy w Matplotlib dodatkowo oddziela (za pomocą odpowiedniego algorytmu) graficznie wartości odstające od pozostałych danych.
- Na wykresie czerwona linia w środku prostokąta odpowiada medianie, poziome krawędzie prostokąta („pudełka”) odpowiadają pierwszemu i trzeciemu kwartylowi a krańce linii „wychodzących” z pudełka – maksimum i minimum. Wartości odstające reprezentowane są przez niewielkie okręgi.

Wykres pudełkowy – przykład

```
plt.boxplot([1,2,3,3,4,5,5,5,6,7,7,14])
plt.title('Przykład - wykres pudełkowy')
plt.yticks(range(0,14))
plt.show()
```



3.2.1. Wykres kolumnowy (*bar chart*)

- Za pomocą wykresu kolumnowego przedstawiamy wartości liczbowe dla osobnych kategorii. Wysokość słupka odpowiada liczbie, która charakteryzuje daną kategorię.
- Uwaga – w przypadku „wykresu kolumnowego” słupki danych są zawsze pionowe. Ogólniejszą nazwą jest „wykres słupkowy” – wtedy dopuszcza się również, że słupki danych umieszczone są w poziomie. Używa się również nazwy „wykres paskowy” (jako synonim „wykresu słupkowego”).

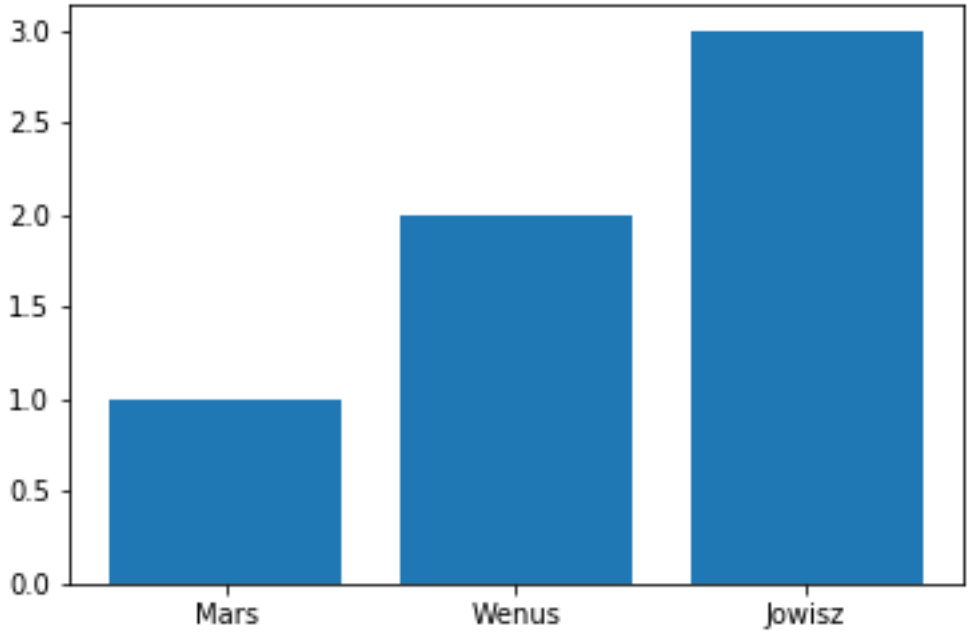
Wykres kolumnowy (*bar chart*)

- Aby stworzyć wykres kolumnowy w Matplotlib musimy najpierw odpowiednio przygotować dane. Dokładniej, musimy stworzyć dwa zestawy danych (mogą być to listy, macierze, albo elementy typu pd.Series):

- lista wartości, które mają pojawić się na osi X (mogą być numeryczne lub tekstowe). Dla każdej wartości pojawi się odpowiadająca jej kolumna.
- lista wartości liczbowych, które będą odpowiadały wysokości kolumn (a więc odwzorowane będą na osi Y).

Wykres kolumnowy – prosty przykład

```
#najpierw tworzymy trzy listy z wartościami
x = ['a', 'b', 'c']
y = [1,2,3]
podpisy = ['Mars','Wenus','Jowisz']
plt.bar(x,y)
plt.xticks(x,podpisy)
plt.show()
```



Uwaga – w etykietach kolumn domyślnie pojawią się kategorie z listy x ('a', 'b', 'c'). Jeśli chcemy, żeby na wykresie pojawiły się inne etykiety, możemy zrobić to za pomocą drugiego argumentu funkcji plt.xticks – tutaj użyliśmy do tego specjalnie stworzonego wektora tekstowego „podpisy”.

Wykres kolumnowy – przykład 2

- Analizując jedną zmienną kategorialną możemy użyć wykresu kolumnowego do przedstawienia jak często występują w zmiennej poszczególne kategorie.
- W tym celu należy odpowiednio przygotować dane.

Wykres kolumnowy – przykład 2

- Analizując jedną zmienną kategorialną możemy użyć wykresu kolumnowego do przedstawienia jak często występują w zmiennej poszczególne kategorie.
- W tym celu należy odpowiednio przygotować dane.
- Przypomnijmy zbiór z poprzednich zajęć – stworzymy wykres kolumnowy z częstościami wystąpień poszczególnych kategorii w zmiennej „zawód”.
- Dane można przygotować na różne sposoby, my zaprezentujemy jeden z nich (na kolejnym slajdzie)
- Najpierw wczytujemy dane:

```
excel = pd.ExcelFile('Wyk4_dane.xlsx')
df = excel.parse('aggregation_example')
```

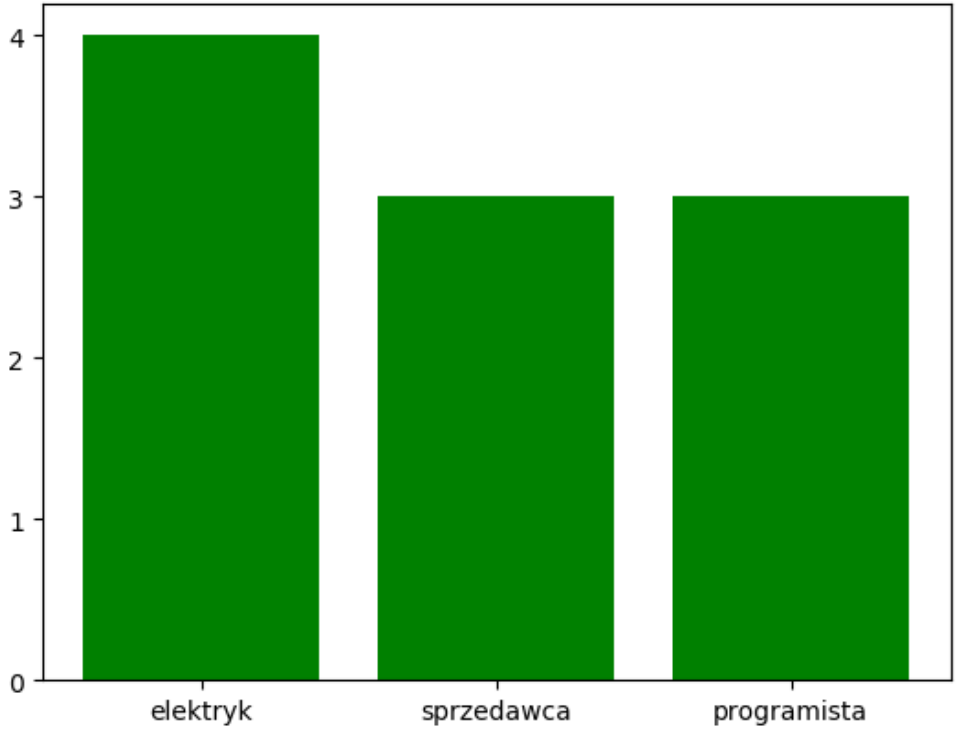
imię	płeć	zawód	wiek	staż
Marcin	M	elektryk	36	23
Ewa	K	programista	32	2
Ola	K	programista	25	12
Tomek	M	sprzedawca	24	1
Ania	K	elektryk	29	29
Elwira	K	sprzedawca	55	48
Grzegorz	M	elektryk	33	32
Jan	M	sprzedawca	27	9
Filip	M	elektryk	51	23
Róża	K	programista	49	52

```
czestosci = df.zawód.value_counts()
data = czestosci.reset_index()
print(data)
```

```
In [535]: print(data)
          index  zawód
0     elektryk       4
1   sprzedawca       3
2  programista       3
```

```
czestosci = df.zawód.value_counts()
data = czestosci.reset_index()
print(data)
```

```
In [535]: print(data)
          index  zawód
0     elektryk       4
1    sprzedawca      3
2   programista      3
```




```
data.columns = ["zawód", "licznik"]

plt.bar(data.zawód, data.licznik, color='green')
plt.yticks(range(0,5))
plt.show()
```

3.2.2. Wykres kołowy (*pie chart*)

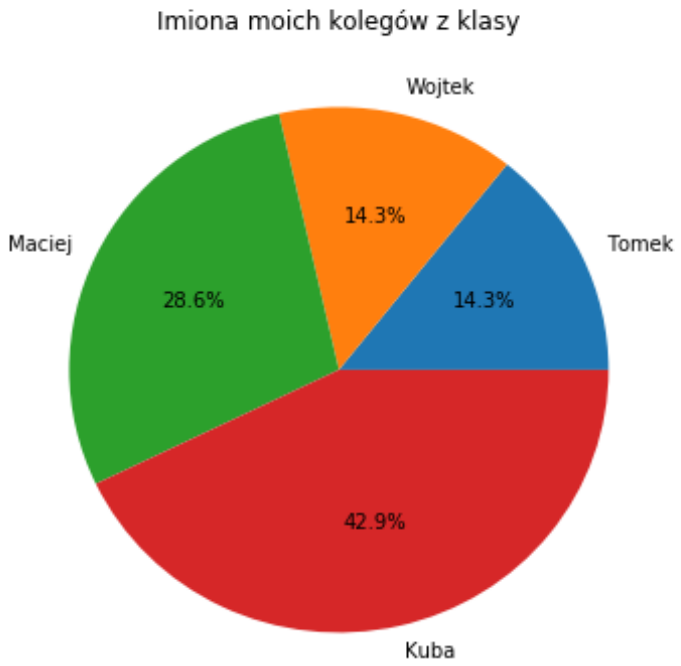
- Inny sposób wizualizacji danych kategorialnych to wykres kołowy. W Matplotlib tworzymy go za pomocą polecenia „plt.pie”. Wykres taki zostanie stworzony, jeśli podamy po prostu ciąg liczb, z których każda będzie odpowiadać jednemu kawałkowi wykresu kołowego (pole odpowiedniego wycinka koła będzie odpowiadać udziałowi danej liczby w sumie liczb).
- Jednak aby wykres był czytelny, należy również podać nazwy kategorii, które będą odpowiadać wycinkom koła (i tym samym wartościom z listy danych). Do tego używany jest argument „labels”.

Wykres kołowy (*pie chart*)

- Można również dodać argument „autopct” ustalając jego wartość np. na „%1.1f%%”, aby dodać do wykresu wartości procentowe odpowiadające odpowiednim wynikom.
- Warto także ustalić wielkość wykresu za pomocą polecenia „plt.figure” (tak, aby koło było faktycznie kołem, a nie elipsą), dodawanego przed poleceniem „plt.pie” (patrz przykład na następnym slajdzie).
- Uwaga – wielu analityków uważa, że należy unikać wykresów kołowych ze względu na percepcyjne trudności z porównywaniem wielkości poszczególnych wycinków koła (gdy kąty różnią się o niewielką wartość, trudno rozstrzygnąć, który jest większy; w przypadku wykresu kolumnowego jest to zazwyczaj łatwiejsze).

Wykres kołowy – przykład

```
plt.figure(1 , figsize =(6 ,6))
plt.pie([1,1,2,3], labels=['Tomek','Wojtek','Maciej','Kuba'], autopct='%1.1f%%')
plt.title('Imiona moich kolegów z klasy')
plt.show()
```



4. Wizualizacja dwóch (i więcej) zmiennych

4.1. Zmienne ilościowe:

- 4.1.1. Wykresy funkcji.
- 4.1.2. Wykres rozrzutu.

4.2. Kombinacja zmiennych jakościowych i ilościowych.

- 4.2.1. Wykres rozrzutu.
- 4.2.2. Wykres kolumnowy.
- 4.2.3. Podwójny histogram.
- 4.2.4. Wielokrotne wykresy dotyczące zmiennej ilościowej dla każdej z kategorii zmiennej jakościowej.

4.3. Szeregi czasowe

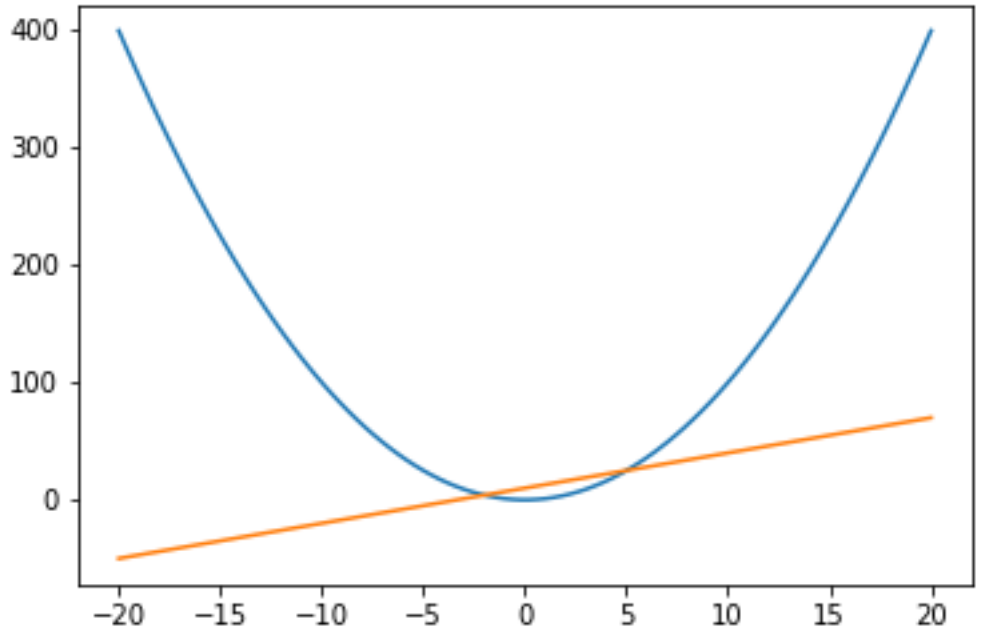
Wizualizacja dwóch zmiennych numerycznych - komentarz

- Jeśli użyjemy polecenia „plt.plot”, punkty (odpowiadające sobie pary wartości z dwóch podanych list) połączone będą prostą. Taki wykres nadaje się dobrze np. do rysowania wykresów funkcji.
- Jeśli jednak pomiędzy dwoma zestawami danych nie ma zależności funkcyjnej (a tak jest najczęściej w przypadku „rzeczywistych” danych), taki wykres staje się najczęściej zupełnie nieczytelny. Należy wtedy stworzyć tzw. *wykres rozrzutu*, tzn. wykres, na którym zaznaczone są tylko punkty odpowiadające odpowiednim parom wartości (bez prostej je łączącej).
- Wykres rozrzutu nadaje się bardzo dobrze do wyrobienia sobie ogólnego poglądu na strukturę relacji pomiędzy dwiema zmiennymi. Uzupełnia on w ten sposób często wiedzę dotyczącą korelacji, która jest pojedynczą wartością liczbową i często nie daje wyczerpującej informacji o związku między dwoma zmiennymi.

4.1.1. Wykres funkcji

```
x = np.linspace(-20,20,100)
y1 = pow(x,2)
y2 = 3*x+10

plt.plot(x,y1)
plt.plot(x,y2)
plt.show()
```

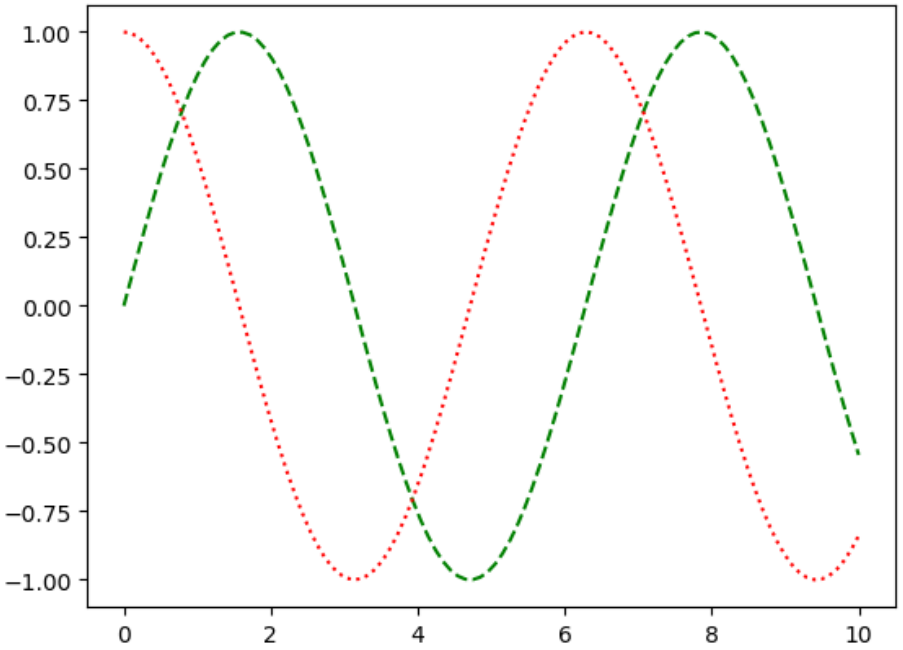


Style - linie

```
x = np.linspace(0, 10, 100)
plt.plot(x, np.sin(x), linestyle = '--', color = 'g')
plt.plot(x, np.cos(x), linestyle= ':', color = 'r')
plt.show()
```

Uwaga – ten sam efekt uzyskamy używając „skrótowych” poleceń:

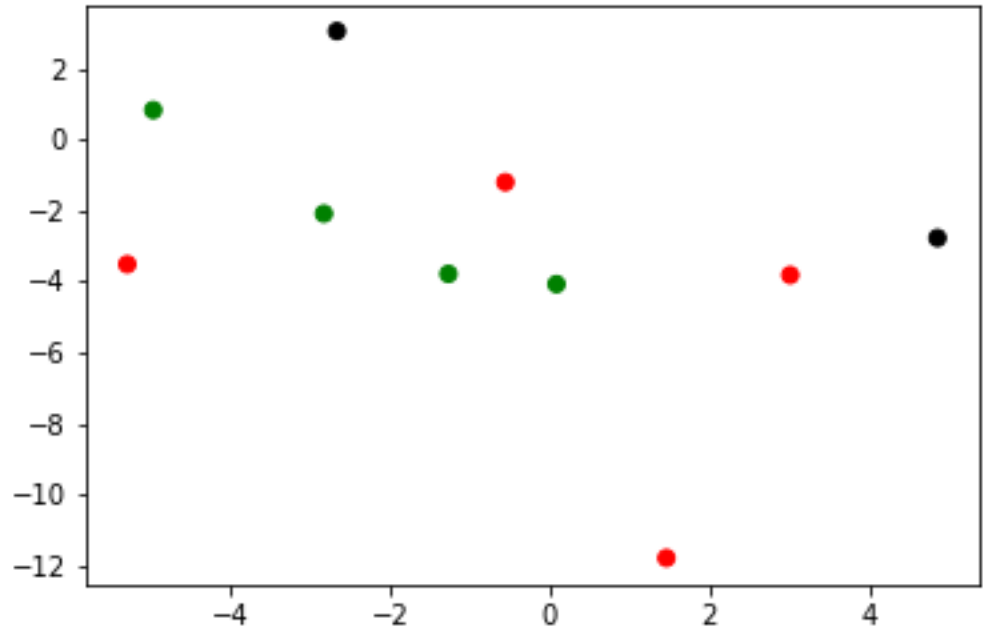
```
plt.plot(x, np.sin(x), '--g')
plt.plot(x, np.cos(x), ':r')
```



4.1.2. Wykres rozrzutu (*scatterplot*)

```
x = np.random.normal(0,5,10)
y = np.random.normal(0,5,10)
col = ['g','g','g','g','k','k','r','r','r','r']

plt.scatter(x,y,c=col)
plt.show()
```



Uwaga – w tym przypadku zmienne x oraz y są wartościami numerycznymi (ilościowymi), natomiast za pomocą koloru możemy reprezentować zmienne kategorialne (jakościowe).

Wykres rozrzutu – dalsze opcje

1. Do wykresu rozrzutu możemy również dodać zmienną za pomocą argumentu „s” (size).

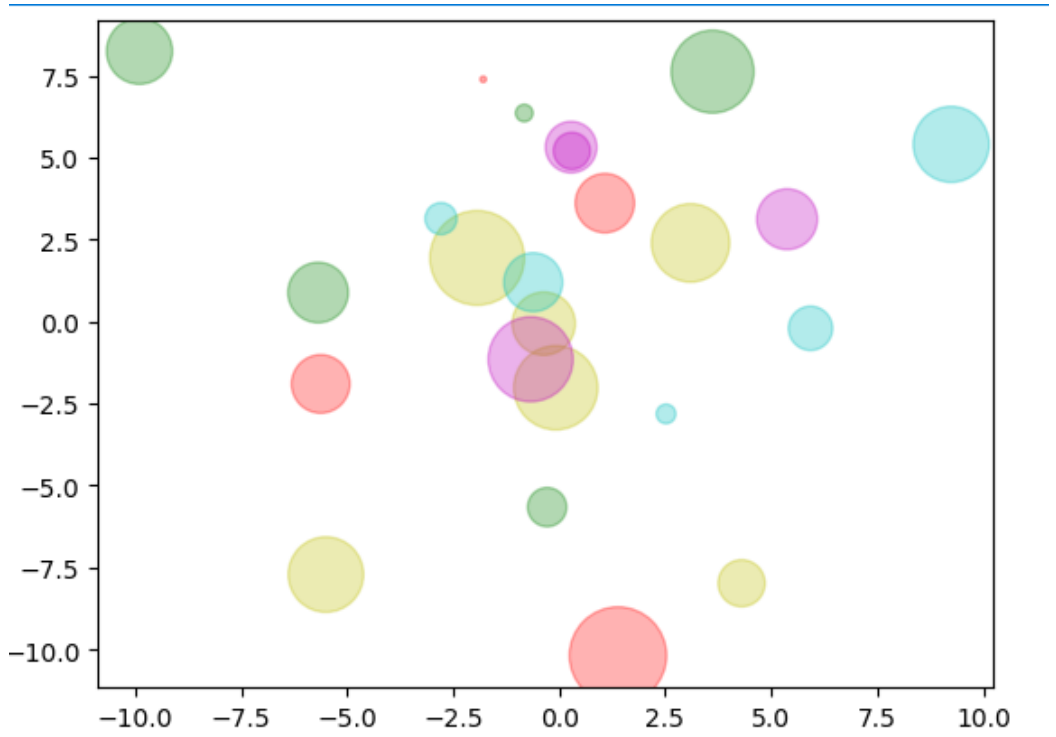
```
plt.scatter(x, y, s=size_data)
```

2. Za pomocą argumentu „alpha” możemy zwiększać „przezroczystość” poszczególnych punktów. Przydaje się to w szczególności, gdy punktów na wykresie jest bardzo dużo.

```
plt.scatter(x, y, c=col, alpha=0.8)
```

Wykres rozrzutu – przykład2

```
x = np.random.normal(0,5,50)
y = np.random.normal(0,5,50)
colors = ['g']*10 + ['y']*10+ ['r']*10 + ['c']*10 + ['m']*10
sizes = 150 * np.random.normal(0,5,50)
plt.scatter(x, y, c=colors, s=sizes, alpha=0.3)
plt.show()
```



4.2.2. Wykres kolumnowy dla 2 zmiennych

- Na wykresie słupkowym (kolumnowym) można przedstawić więcej zmiennych niż jedną na różne sposoby, oto dwa przykłady:

1. Każdy słupek odpowiada kategoriom pierwszej zmiennej (jakościowej), natomiast wysokość słupka odpowiada zagregowanej wartości opartej na innej zmiennej (ilościowej).

2. Dla każdej kategorii danej zmiennej na osi X pokazujemy więcej słupków, przy czym każdy z nich odpowiada kategorii innej zmiennej (również jakościowej). Mogą one być ustawione obok siebie, lub „jeden na drugim”. Każdy ze słupków może przedstawiać np. agregacje wartości trzeciej zmiennej (ilościowej).

Aby wykonać wykres pierwszego typu (słupki odpowiadają kategoriom pierwszej zmiennej, wysokość słupka odpowiada zagregowanej wartości drugiej zmiennej) trzeba znów przygotować odpowiednio dane.

Tym razem użyjemy innego (również prostego) przykładu. Znajduje się on w drugiej zakładce używanego już wcześniej pliku:

```
excel = pd.ExcelFile('Wyk5_dane.xlsx')

df = excel.parse('example2')
```

imię	płeć	zawód	zarobki
Marcin	M	elektryk	2000
Ewa	K	programista	3900
Ola	K	programista	4300
Tomek	M	sprzedawca	3000
Ania	K	elektryk	2900
Elwira	K	sprzedawca	2600
Grzegorz	M	elektryk	3100
Jan	M	sprzedawca	2400
Filip	M	elektryk	2900
Zbigniew	M	programista	3000
Maria	K	programista	3700
Wojtek	M	elektryk	2500
Ula	K	sprzedawca	2500
Róża	K	programista	2500

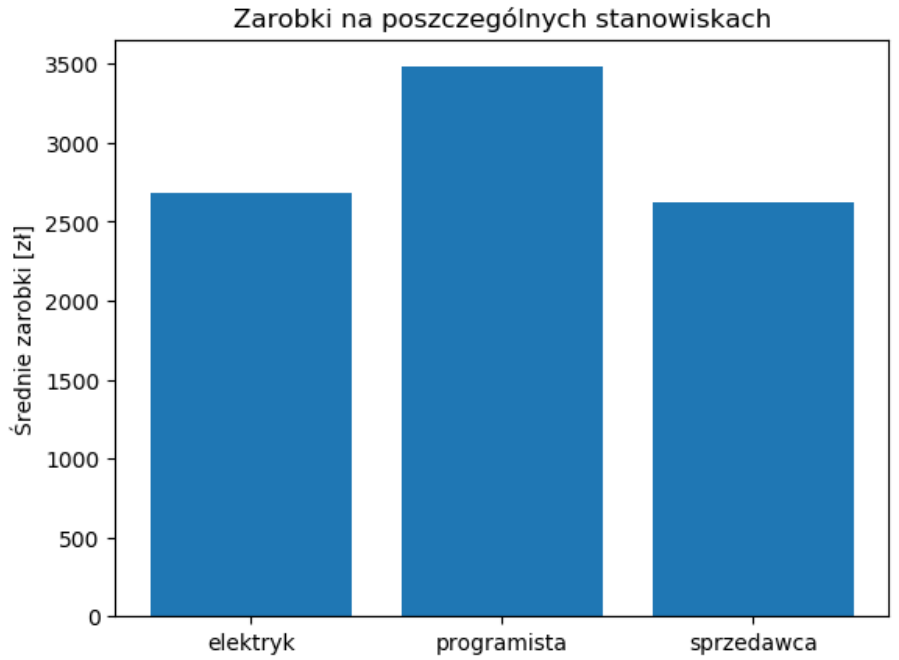
```
x = df.groupby('zawód')['zarobki'].mean()
data = x.reset_index()
print(data)
```

```
          zawód  zarobki
0     elektryk      2680
1  programista      3480
2   sprzedawca      2625
```

```
x = df.groupby('zawód')['zarobki'].mean()
data = x.reset_index()
print(data)
```

```
          zawód  zarobki
0      elektryk     2680
1   programista     3480
2    sprzedawca     2625
```

```
plt.bar(data.zawód,data.zarobki)
plt.ylabel('Średnie zarobki [zł]')
plt.title('Zarobki na poszczególnych stanowiskach')
plt.show()
```



- Wykonanie wykresu drugiego typu nastręcza nieco więcej trudności. Spróbujmy wykonać dla zarobków pogrupowanych po płci i zawodzie.

```
df.groupby(['zawód','płeć'])['zarobki'].mean()
```

```
zawód       płeć
elektryk    K       2900
            M       2625
programista K       3600
            M       3000
sprzedawca  K       2550
            M       2700
Name: zarobki, dtype: int64
```

- Wykonanie wykresu drugiego typu nastręcza nieco więcej trudności. Spróbujmy go wykonać dla zarobków pogrupowanych po płci i zawodzie.

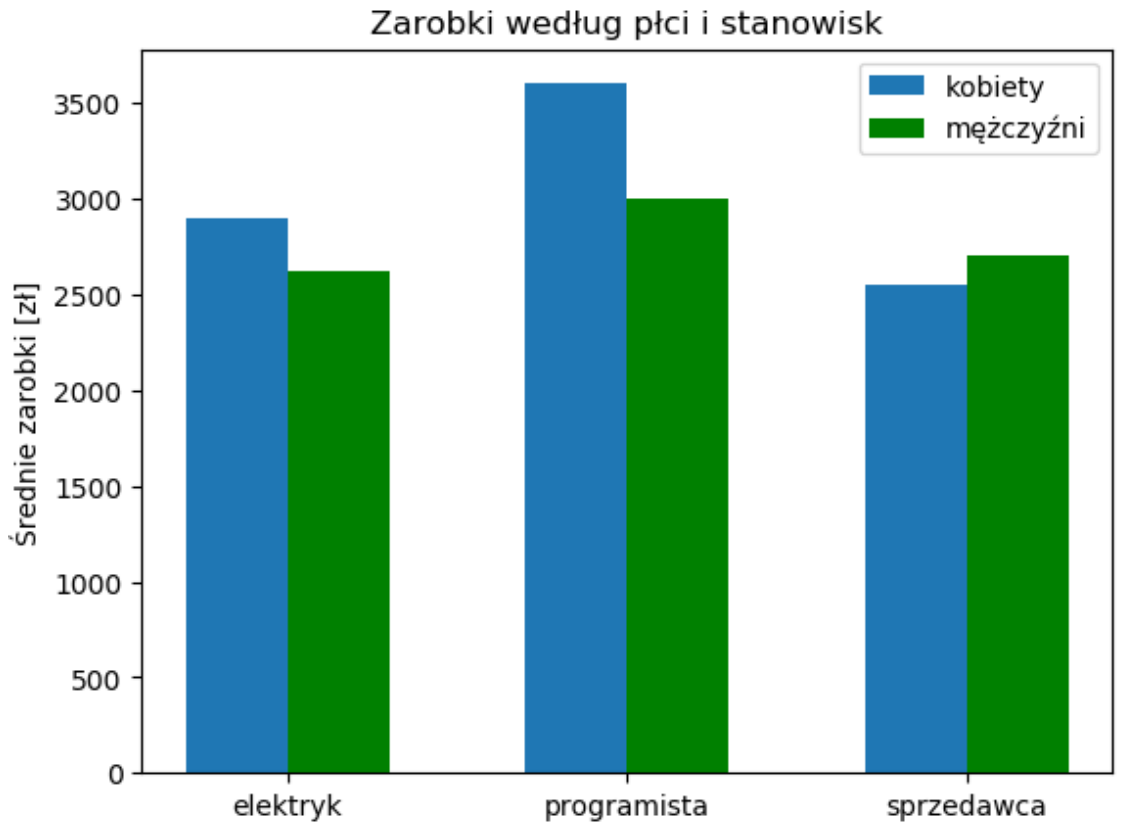
```
df.groupby(['zawód','płeć'])['zarobki'].mean()
```

```
zawód        płeć
elektryk     K       2900
             M       2625
programista  K       3600
             M       3000
sprzedawca   K       2550
             M       2700
Name: zarobki, dtype: int64
```

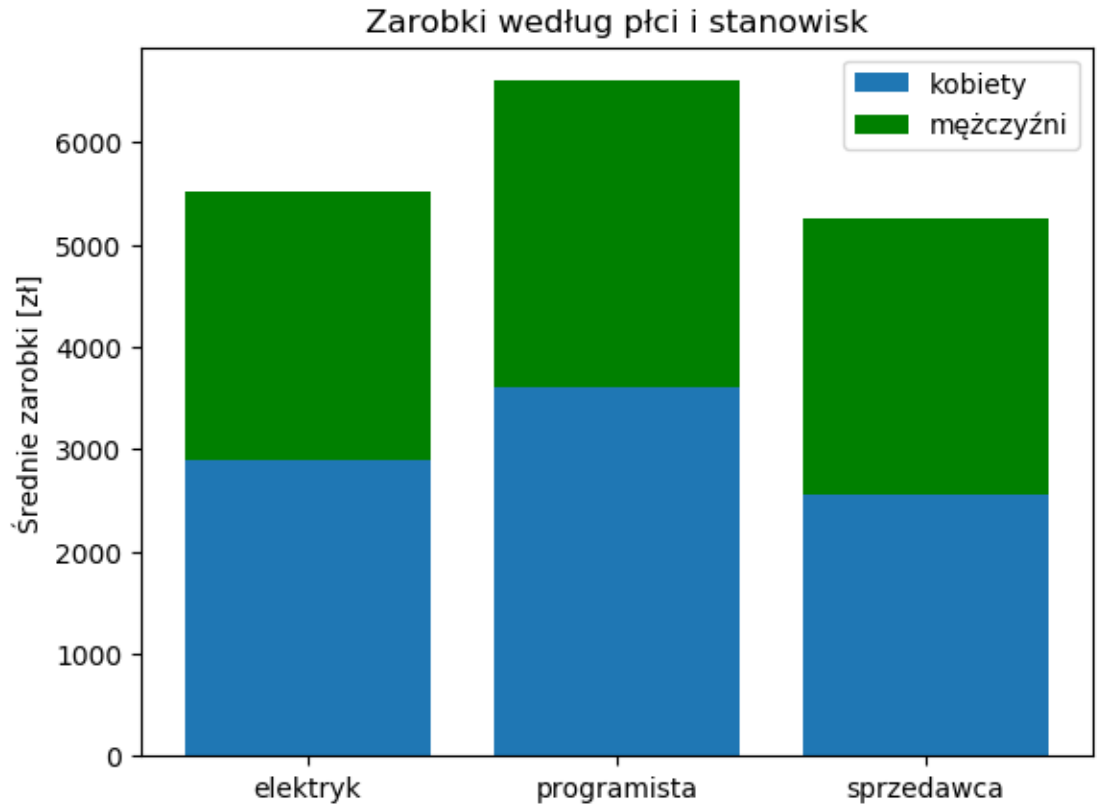
- Na podstawie pogrupowanych wartości stworzymy dane dla wykresu:

```
kobiety = [2900, 3600, 2550]        #zarobki kobiet według zawodów
mężczyźni = [2625, 3000, 2700]   #zarobki mężczyzn według zawodów
zawód = ['elektryk', 'programista', 'sprzedawca']
ind = np.arange(3)  #w tej zmiennej zaznaczymy ile będzie głównych
                    znaczników na osi X
w = 0.3                    #musimy też ustalić szerokość kolumn
```

```
plt.bar(ind, kobiety, width=w, label='kobiety')
plt.bar(ind + w, mężczyźni, width=w, label='mężczyźni', color='g') #kolumny z drugiej
                                    serii danych musimy przesunąć o odpowiednią wartość „w prawo”
plt.xticks(ind + w/2, zawód)
plt.ylabel('Średnie zarobki [zł]')
plt.title('Zarobki według płci i stanowisk')
plt.legend()
plt.show()
```



```
w = 0.8
plt.bar(ind, kobiety, width=w, label='kobiety')
plt.bar(ind, mężczyźni, width=w, label='mężczyźni', color='g, bottom=kobiety)
plt.xticks(ind, zawód)
plt.ylabel('Średnie zarobki [zł]')
plt.title('Zarobki według płci i stanowisk')
plt.legend()
plt.show()
```



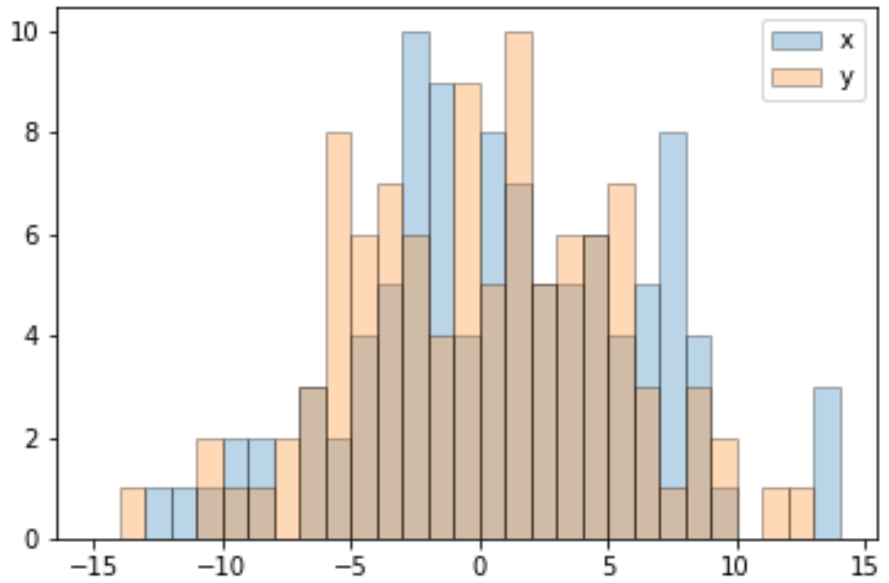
Uwaga – na wykresie „skumulowanym” trudniej jest odczytać czy zarobki mężczyzn i kobiet istotnie się różnią. „Widać” za to, że programiści zarabiają więcej od elektryków i sprzedawców (niezależnie od płci)

4.2.3. Podwójny histogram

- Dwie zmienne numeryczne można też przedstawić za pomocą dwóch nałożonych na siebie histogramów. Wtedy jednak muszą być mierzone na tej samej skali i mieć relatywnie podobne średnie i odchylenia standardowe (aby wizualne porównanie pomiędzy nimi było możliwe).

```
x = np.random.normal(0,5,100)
y = np.random.normal(0,5,100)
bins=range(-15,15)

plt.hist(x, bins, alpha=0.3, label='x', edgecolor='k')
plt.hist(y, bins, alpha=0.3, label='y', edgecolor='k')
plt.legend()
plt.show()
```



4.2.4. Wielokrotne wykresy dotyczące zmiennej ilościowej dla każdej z kategorii zmiennej jakościowej.

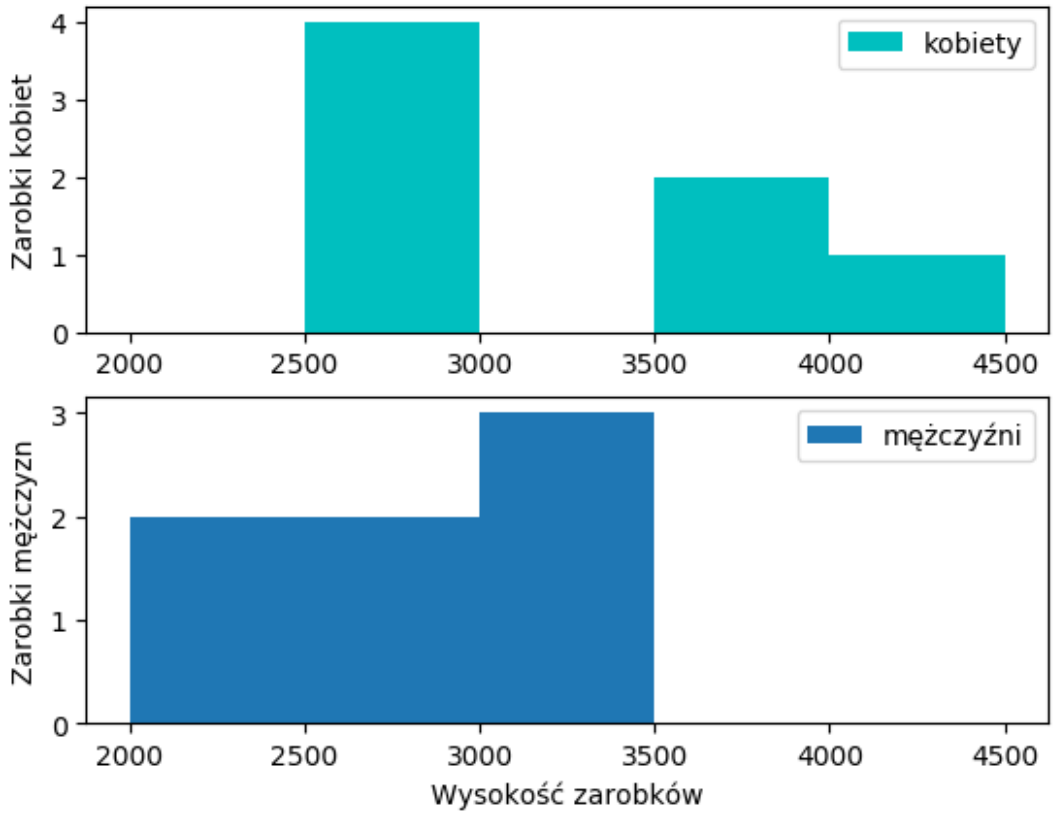
1. Wreszcie, można zestawić ze sobą zmienną kategorialną i numeryczną przez wykonanie wykresu dla zmiennej numerycznej (np. histogramu) osobno dla każdej z kategorii zmiennej kategorialnej.
2. Można tę metodę rozwinąć i np. przedstawić dwie zmienne kategorialne i dwie numeryczne: tworzymy wtedy np. siatkę złożoną z kombinacji wartości dwóch zmiennych kategorialnych i dla każdej kombinacji tworzymy wykres rozrzutu, obrazujący dwie zmienne numeryczne.

Przykład1

Skorzystamy z tego samego zbioru, co ostatnio. Przestawimy histogramy dla zarobków osobno dla kobiet i mężczyzn. Aby to zrobić, musimy przygotować najpierw dane:

```
kobiety = df.zarobki[df.płeć=='K']
mężczyźni = df.zarobki[df.płeć=='M']
```

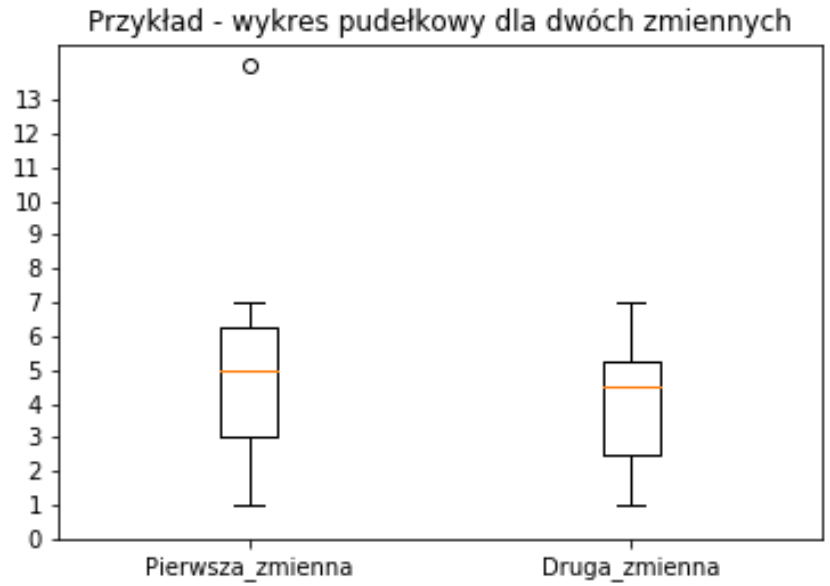
```
plt.subplot (211)
plt.hist(kobiety, bins=[2000,2500,3000,3500,4000,4500], label='kobiety', color='c')
plt.ylabel('Zarobki kobiet')
plt.legend()
plt.subplot (212)
plt.hist(mężczyźni, bins=[2000,2500,3000,3500,4000,4500], label='mężczyźni')
plt.ylabel('Zarobki mężczyzn')
plt.xlabel('Wysokość zarobków')
plt.legend()
plt.show()
```



Przykład2

Warto też w niektórych przypadkach stworzyć serię wykresów pudełkowych, np. dla różnych zmiennych albo różnych kategorii jednej zmiennej.

```
plt.boxplot([[1,2,3,3,4,5,5,5,6,7,7,14], [1,1,3,4,4,5,5,5,6,7,7,1]])
plt.title('Przykład - wykres pudełkowy dla dwóch zmiennych')
plt.xticks([1,2],['Pierwsza_zmienna','Druga_zmienna'])
plt.yticks(range(0,14))
plt.show()
```



4.3. Wizualizacja szeregów czasowych – wstępne informacje

- Bardzo często zdarza się, że chcemy zwizualizować zmienność jakiejś własności w czasie. Do tego celu idealnie nadaje się wykres liniowy. Na osi X mamy wtedy daty, a na osi Y kolejne pomiary interesującej nas cechy.
- Aby sporządzić tego typu wykres w Pythonie należy najpierw wczytać dane w taki sposób, aby daty były umieszczone w etykietach wierszy.

```
data = pd.read_csv('file.csv', parse_dates=['date'], index_col='date')
```

- Powyższe polecenie wczyta plik w taki sposób, że zmienna o nagłówku „date” będzie wczytana jako obiekt typu „datetime” (argument „parse_dates”), oraz umieszczona w etykietach wierszy (argument „index_col”).
- Następnie należy po prostu uruchomić poniższe polecenie. Przez „col” rozumiemy tutaj zmienną, w której zawarta jest cecha, która zmienia się w czasie (np. populacja, zysk)

```
plt.plot(data.index, data.col)
```

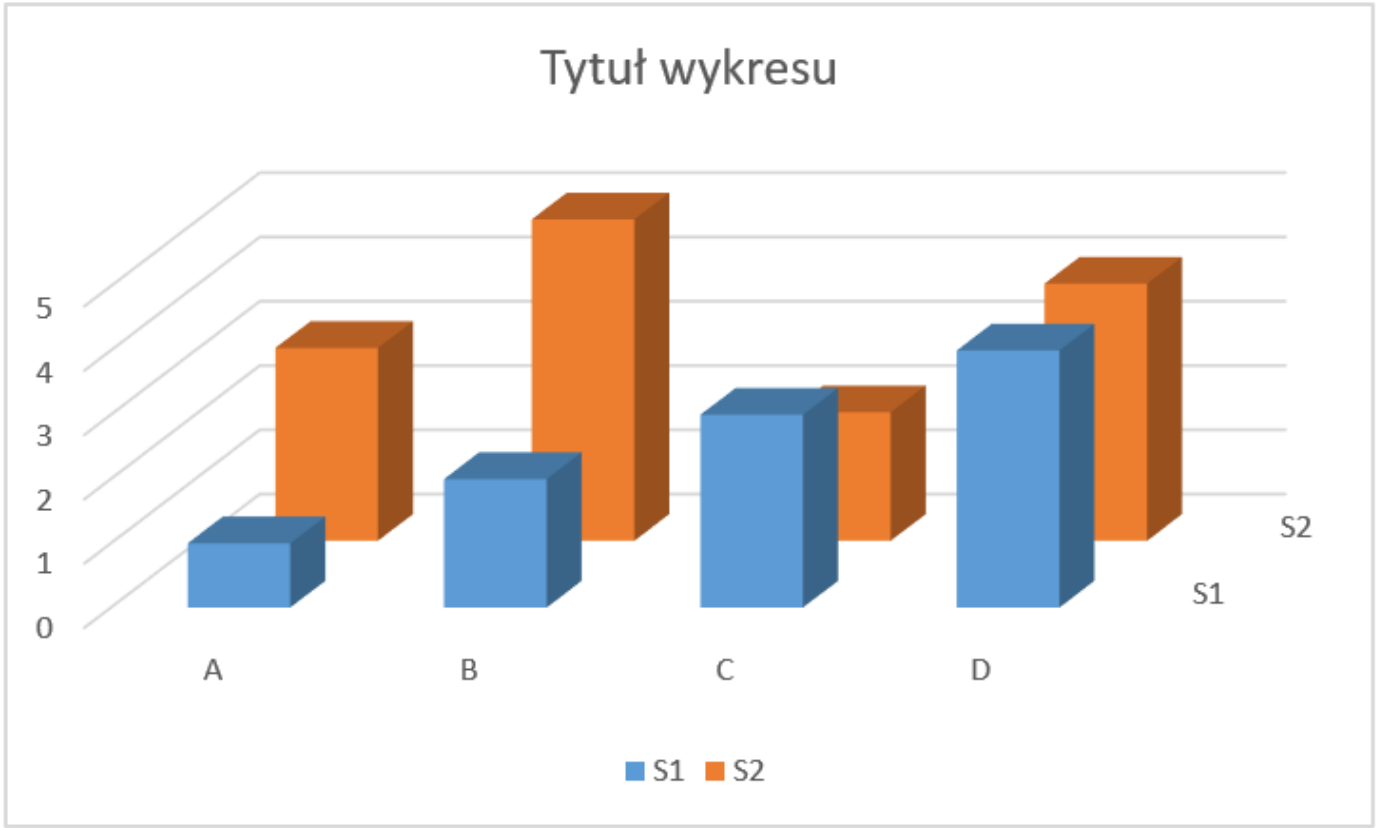
- Bardziej szczegółowe informacje – np. na kursach data.camp!

5. Wybrane błędy w sporządzaniu wizualizacji

1. Brak skali na osi pionowej lub brak osi pionowej.
2. Skala nie zaczyna się od 0.
3. Słupki nie odpowiadają wartościom liczbowym.
4. Trójwymiarowe wykresy kołowe i kolumnowe.
5. Niewłaściwy opis lub legenda.
6. Nieczytelny wykres – nadmiar elementów na diagramie.
7. Nieczytelny wykres – zbyt mała czcionka.

Przykład – nieczytelność wykresów 3-wymiarowych

	S1	S2
A	1	3
B	2	5
C	3	2
D	4	4



Polecane kursy na platformie data.camp

- Intermediate Python
- Introduction to Data Visualisation with Matplotlib
- Introduction to Data Visualisation in Python

Dla chętnych (dodatkowo):

1. Zapoznanie się z rozdziałami powyższych kursów dotyczącymi szeregów czasowych (time-series).
2. Zapoznanie się z innym pakietem Seaborn, również służącym do wizualizacji danych. Seaborn jest zbudowany na Matplotlib ale ma szersze możliwości.

- Introduction to Data Visualisation with Seaborn

- Warto również zajrzeć do dokumentacji matplotlib.

http://matplotlib.org/1.5.3/index.html

http://matplotlib.org/examples/index.html

https://matplotlib.org/3.1.0/tutorials/introductory/pyplot.html

Ćwiczenie 1.

Skopiuj kod z wykładu do Spydera i spróbuj wygenerować wykresy przedstawione na slajdach. Wygeneruj również „na próbę” dalsze wykresy, manipulując dowolnie wybranymi argumentami.

Ćwiczenie 2.

Użyj funkcji plt.plot aby stworzyć następujące wykresy:

1. Wykres funkcji y=2x+5, podając jako dane dwie listy: argumenty będące liczbami naturalnymi z przedziału [0,10], oraz listę wartości funkcji dla tych argumentów. Do wykresu dodaj dowolne etykiety osi oraz tytuł.
2. Trzy wykresy, łącząc je w jeden obraz za pomocą funkcji plt.subplot. Dane przygotuj analogicznie w powyższym podpunkcie. Powinny one zawierać wykresy następujących funkcji:
   a) y=x+1
   b) y=5x
   c) y=3x+3

   Do wykresu dodaj legendę, dla każdej funkcji określ również inny kolor.

Ćwiczenie 3.

Otwórz zbiór „USstates.txt”. Zawiera on dane dotyczące 50 stanów USA, zbierane w latach 70-tych. Bardziej szczegółowe informacje o zbiorze (w tym o znaczeniu zmiennych) można znaleźć na stronie: https://www.rdocumentation.org/packages/datasets/versions/3.6.1/topics/state

Wykonaj następujące wizualizacje:

1. Wykres pudełkowy dla wybranych czterech zmiennych numerycznych.
2. Histogramy dla zmiennych „Population”, „Income” i „Illiteracy”. W każdym przypadku dobierz szerokość kolumn za pomocą argumentu „bins” i dostosuj znaczniki na osi X tak, aby wykres był jak najbardziej komunikatywny.

Ćwiczenie 3 (cd).

3. Utwórz nową zmienną – powinna zawierać wartość „pop_high” dla stanów o populacji większej lub równej medianie z populacji, oraz wartość „pop_low” w odwrotnym przypadku. Utwórz teraz w Pythonie (za pomocą funkcji „subplot”) rysunek zawierający dwa histogramy dla zmiennej „Income” – każdy dla jednej z wartości nowo utworzonej zmiennej kategorialnej. Powtórz tę czynność dla zmiennej „Murder”.
4. Stwórz wykres rozrzutu dla każdej możliwej pary zmiennych spośród zmiennych „Population”, „Income”, „Illiteracy” oraz „Life exp”. Policz również korelacje pomiędzy parami zmiennych, np. korzystając z funkcji „data.corr()” (pod „data” wstaw nazwę Twojego zbioru). Jak korelacje mają się do wykresów rozrzutu?
5. Zwizualizuj na jednym wykresie 3 zmienne:
   a) Stwórz wykres rozrzutu dla zmiennych „Murder” i „Illiteracy”. Dodatkowo, niech zmienna „Income” będzie reprezentowana przez wielkość punktów.
   b) Na tym samym wykresie rozrzutu dodaj utworzoną w punkcie 3 zmienną, której dwie kategorie powinny być reprezentowane przez dwa kolory.

Ćwiczenie 4

Otwórz zbiór zawarty w pliku „Wyk5_dane.xlsx” w zakładce „example2” i stwórz następujące wizualizacje:

1. Histogram dla zmiennej „zarobki”, wykonany na dwa sposoby:
   a. Przedziały o długość 500, rozpoczynając od wartości 2000.
   b. Liczba przedziałów ustalona na 5.

Do każdej kolumny histogramu dodaj obramowanie w kolorze czarnym.

2. Wykres kołowy dla zmiennej „zawód”. Upewnij się, że w wycinkach koła podane są odpowiadające im wartości procentowe.

Ćwiczenie 5.

Otwórz plik „countries_of_the_world.xlsx” z wybranymi danymi demograficznymi i geograficznymi dotyczącymi wszystkich państw świata (państwa stanowią więc obserwacje, a każdy wiersz odpowiada jednym państwu). Zbiór został pobrany ze strony www.kaggle.com (i nieco zmodyfikowany), informacje o nim można uzyskać na stronie https://www.kaggle.com/search?q=countries+of+the+world.

Korzystając ze zbioru odpowiedz na zamieszczone poniżej pytania. Wykonaj zadanie zarówno z wykorzystaniem Excela jak i Pythona.

1. Jaka jest średnia populacja w każdym z regionów (zmienna „Region”)?
2. Dla ilu krajów występują braki danych w zmiennej „Literacy (%)”
3. W którym kraju występuje największa gęstość zaludnienia („Pop.density..”)?
4. Ile jest krajów z populacją powyżej 20 mln w każdym z regionów?
5. Jaka jest średnia i odchylenie standardowe ze zmiennej „Agriculture” (odsetek osób zatrudnionych w rolnictwie) dla agregatu wszystkich krajów z kontynentów amerykańskich?

Ćwiczenie 5 (cd).

5. Stwórz histogram dla zmiennej „GDP ($ per capita)” (uwaga na braki danych!). Każdy słupek powinien odpowiadać przedziałowi o długości 10000, dodaj również czarne obramowanie słupków. Dodaj tytuł („Countries of the world - GDP/capita”) oraz opis osi X („GDP [$ per capita]”).

6. Stwórz wykres pudełkowy dla zmiennej „GDP ($ per capita)”. Które kraje są wartościami odstającymi oznaczonymi na wykresie?

7. Stwórz dwa wykresy kolumnowe:

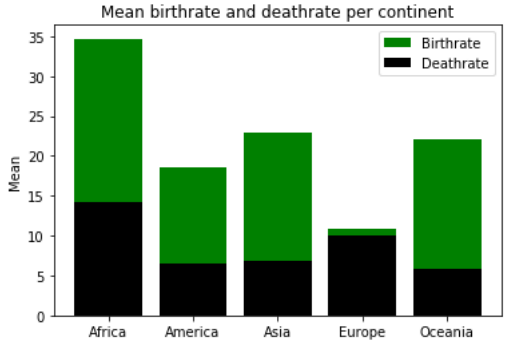
- na osi X powinny znaleźć się kategorie zmiennej „Continents” a na osi Y liczbę krajów z każdego kontynentu
- na osi X kategorie zmiennej „Continents”, długość każdej kolumny odpowiada średniej ze zmiennej „GDP ($ per capita)”

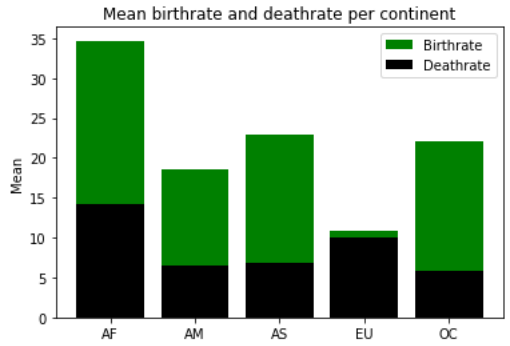
Ćwiczenie 5 (cd).

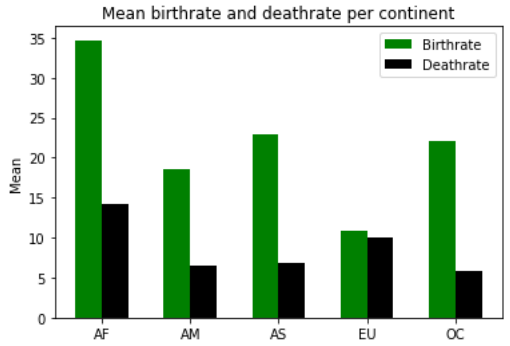
8. Stwórz wykresy kolumnowe, na których zawarte będą dwie serie danych: pierwsza powinna zawierać średnie wartości ze zmiennych „Deathrate” i „Birthrate” dla poszczególnych kontynentów (zmienna „Continents”):

- pierwszy wykres powinien być wykresem skumulowanym, a podpisy powinny pochodzić ze zmiennej „Continents”
- na drugim wykresie zmień podpisy na osi X
- na trzecim wykresie słupki dla poszczególnych kategorii powinny być ustawione „obok” siebie

Wykresy powinny również zawierać legendę i opisy osi Y (patrz poniżej)







Ćwiczenie 5 (cd).

9.

a) Stwórz wykres rozrzutu dla zmiennych „GDP ($ per capita)” oraz „Literacy (%)”. Dodaj odpowiednie opisy osi.

b) Do powyższego wykresu dodaj trzeci parametr – wielkość punktów (argument „s”), dla zmiennej „Population”. Musisz podzielić zmienną przez dużą wartość (np. 3000000), aby wielkość punktów nie przesłoniła całego wykresu.

W materiałach w MS Teams zawarty jest kod służący do wygenerowania wizualizacji do tego ćwiczenia. Do podpunktu 9 został dodany przykładowy kod pozwalający dodać do wykresu tekstu oraz czwartej zmiennej reprezentowanej przez kolor.