Zaawansowana wizualizacja danych

Jarosław Jasiewicz, dr hab. prof. UAM
jarekj@amu.edu.pl

Tworzenie grafiki

Warstwa definicji

> Opis co zrobić z danymi w jakimś w miarę zrozumiałym języku (np. python)

Warstwa przetwarzania

> Tłumaczy język zrozumiały na obiekty graficzne (linie, poligony i ich zbiory)

Warstwa rysowania

> Nakłada obiekty graficzne na jakieś tło

Programowanie proceduralne

wszystkie parametry wykresu muszą być przekazane do procedury w jednym poleceniu

```
scatter(dane[0],dane[1],c="#aa1234",s=12,marker="o")
```

Po przekazaniu parametrów do procedury wykonywany jest wykres

programowanie obiektowe

Tworzony jest obiekt a następnie modyfikowany jest jego stan wewnętrzny. Po ustaleniu parametrów, obiekt przekazywany jest do dalszego przetwarzania

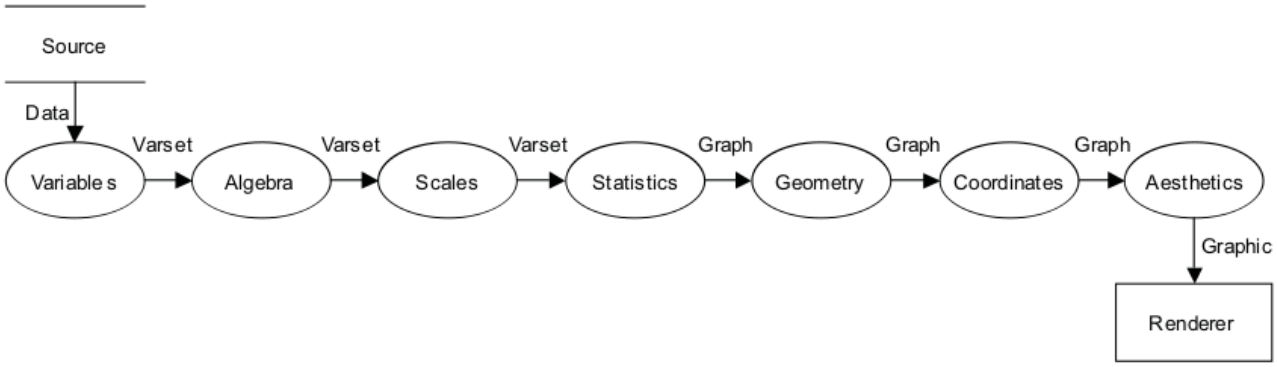
```
fig, ax = plt.subplot() # tworzenie obiektów figura i axes
ax.scatter(x[0],x[1]) # wywołanie metody scatter
ax.draw() # wywołanie fukcji rysującej (niejawne)
```

W przypadku ggplot jest to tylko imitowanie przepływu danych. R nie stosuje notacji *dot*.

Definiowanie wykresu

Z punktu widzenia użytkownika

Z punktu widzenia programu



Gramatyka grafiki

Niezależnie od implentacji istnieje konieczność przetłumaczenia opisu wykresu wykonanego w ramach języka programowania na język opisu geometrii, która zostanie użyta do narysowania wykresu na urządzeniu (np. PDF). Za tłumaczenie języka opisu danych na język opisu obiektów graficznych odpowiada gramatyka grafiki

Leland Wilkinson *Grammar of Graphics*

1. **DATA** : zbiór operacji wyodrębniającej zmienne z danych
2. **TRANS** : transformacje zmiennych (grupowanie, sortowanie)
3. **SCALE** : transformacje skali (log, logit, power)
4. **COORD** : wybór układu odniesienia (polar, merkator, inne),
5. **ELEMENT** : geometria (punkt, linia) i estetyki (color),
6. **GUIDE** : elementy wykresu (axes, legends, etc.).

Elementy graficzne wykresów

- punkty współrzędna X i Y
- łamane - uporządkowna lista współrzędnych x,y
- poligon
    - parametryczny: prostokąt, elipsa, linia
    - nieregularny
- macierz
- pole wektorowe

Zmienne wizualne

Zmienne posługujące się kontekstem wizualnym do przekazywania treści. Zmienne mogą być planarne (położenie) lub retinalne (pozostałe)

- **pozycja**: zmiana położenia
- **rozmiar**: zmiana długości, powierzchni lub gęstości
- **kształt**: nieskończenie wiele, ale niw wszystkie rozróżnialne
- **jasność (value)**: gradnient od najaśniejszego do najciemiejszego

- **kolor (walor)**
- **tekstura**: zróżnicowanie wzorców
- **orientacja**: zróżnicowanie wyrównania obiektów

Rola elementów graficznych

Punkty

> Reprezentują położenie, nie mają długości ani obszaru. Rozmiar, kolor, znak, jasność są atrybutami niezależnymi od położenia i reprezenują niezależne zmienne

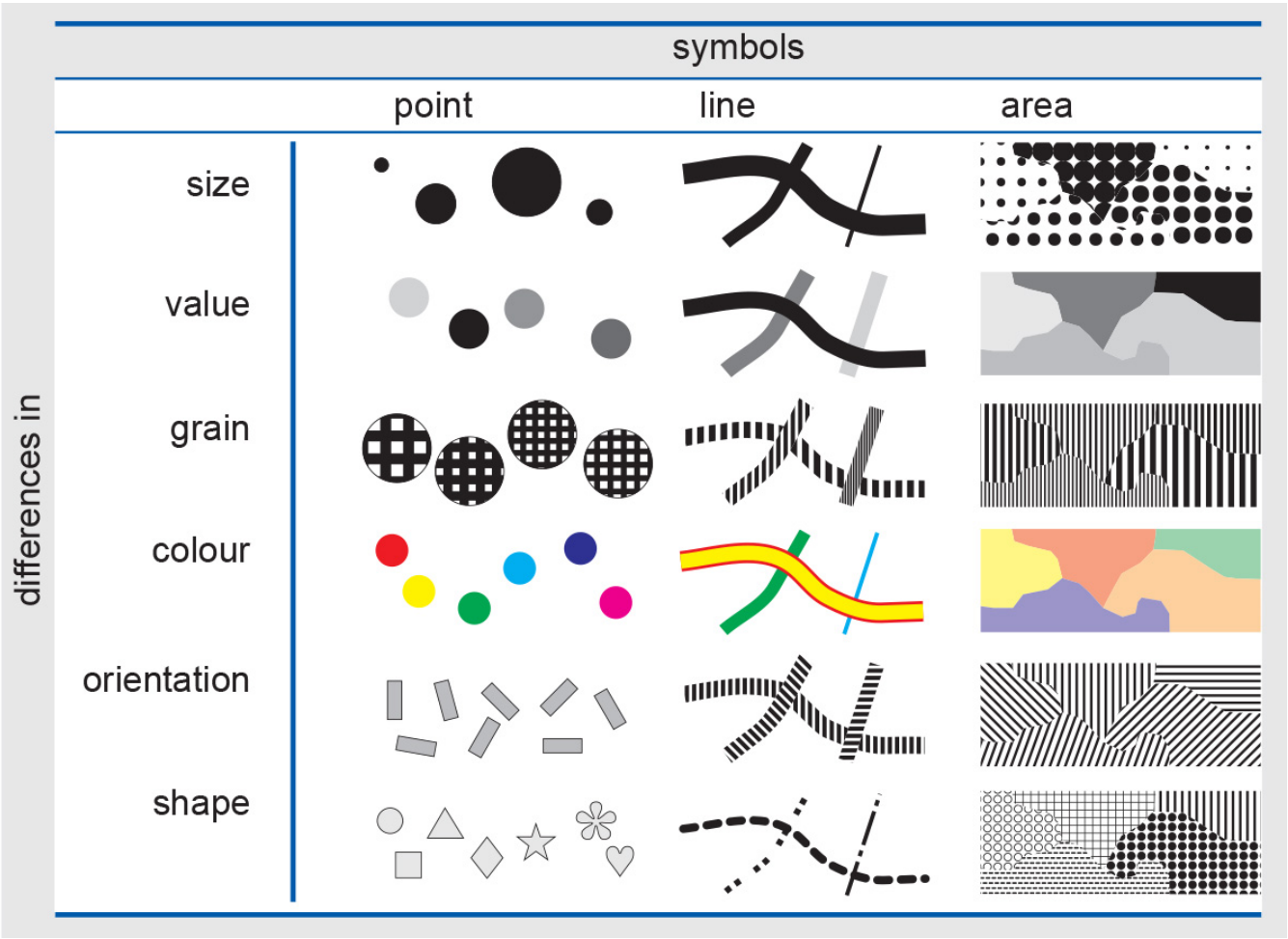
Linie

> Reprezentuje zjawiska o mierzalnej długości, ale nie ma powierzchni. Grubość linii podobnie jak powierzchnia punktu jest atrybutem.

Powierzchnie

> Reprezentuje zjawiska o mierzalnej powierzchni. Obszar może zmieniać położenie ale nie może zmieniać kształtu ani orietnacji

Zmienne wizualne a geometria

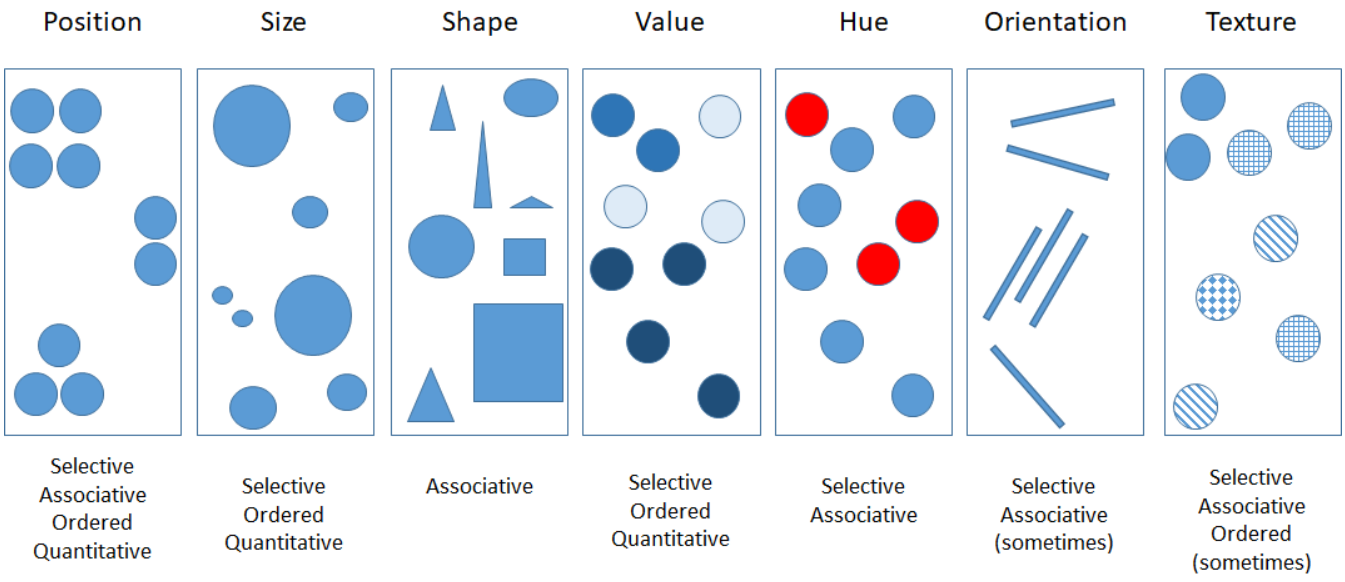


Rola zmiennych wizualnych

- **selekcja**: czy zmiana wartości oddziela od pozostałych obiektów

- **łączenie**: czy wartości zmiennych tworzą grupy i heirarchie
- **ilość**: czy forma zmiennej przekazuje informacje o ilości
- **porządek**: czy wartości zmiennej wywołują efekt porządkowania
- **skala zmian**: ile zmian jest wymaganych aby wartość zmiennej pokazała nieciągłość

Rola poszczególnych zmiennych



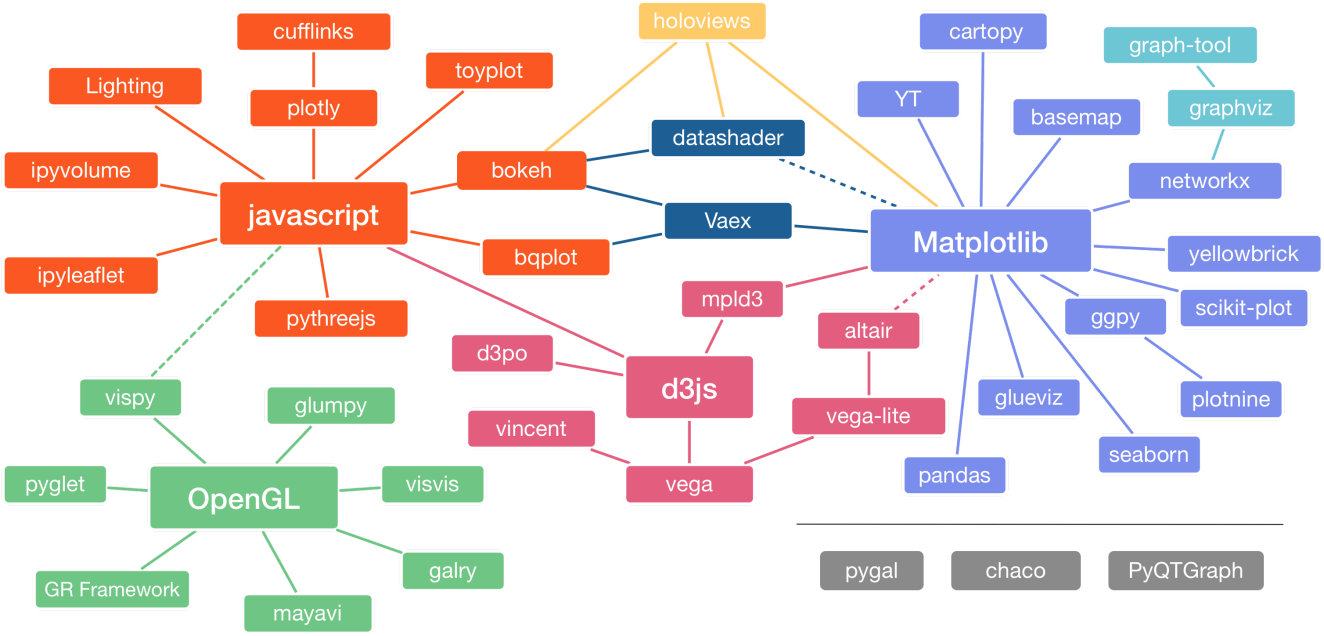
Tworzenie wizualizacji

1. Wymaga zrozumienia relacji informacji i zdanań
2. Utworzenie rzutowania pomiędzy danymi a zmienną wizualną (jaka zmienna reprezentuje jaką informację)
3. Prezentacja wizualizacji
4. Weryfikacja użyteczności

Weryfikacja użyteczności

1. Czy ma być drukowana bw/kolor?
2. Czy ma komunikować ogólną informację czy szczeguły
3. Z jakej odległości ma być oglądana
4. Czy oświetlenie może wpływać na odbiór
5. Rodzaj odbiorcy

System graficzny Pythona



Matplotlib

Rozwijany jako graficzne rozwinięcie Numpy. Zorientowny obiektowo z implementacją proceduralną **pylab** wzorowany na MATLAB. Część suity **scipy umbrella**

- Zalety: dostępny wszędzie gdzie dostępne Numpy, liczne backendy, rozbudowany, wysoce konfigurowalny, liczne specjalistyczne nakładki
- Wady: trudny w obsłudze, nieprzystępny, wiele przestarzałych rozwiązań, nieprzystępna dokumentacja

Oparte na Java Scipt: Plotly i Bokeh

Rozwijany komercyjnie jako biblioteka JS. Część dużej suity data science. Ograniczenia komercyjne. Python tworzy kod JS

- Zalety: interaktywność, duży wybór wykresów i możliwości konfiguracji, bradzo dobra dokumentacja
- Wady: ograniczony do rozwiązań przeglądarkowych, nie dostępny w rozwiązaniach osadzonych i innych backendach, wersja w Python to tylko wrapper.

Oparte na d3JS: Altair

Implementacja w języku Python oparta na gramatyce wizualizacji **Vega** i **Vega lite** w JavaScript. Uzywa składni JSON.

- Zalety: Oparty o gramatykę wizualizacji i JSON
- Wady: takie same jak w przypadku **Plotly** + trudny w obsłudze i mało popularny

Oparte na openGL: Mayavi

Wykorzystuje OpenGL (VTK), przeznaczony do wizualizacji naukowych i 3D

Holoviews

Używa wyjścia matplotlib (wizualizacje statyczne) i Bokheh/Plotly (interaktywne)

- Zalety: zalety obu systemów
- Wady: wady i ograniczenia obu systemów

Oparty na SVG: PyGAL

Opisać

Alternatywne API matplotlib

Matplotlib dostarcza przede wszystkim tworzywo do budowania zaawansowanych systemów. Alternatywne API generujące obiekty Matplotlib

Plotnine

> Nakładka imitująca składnię *ggplot*.

Seaborn

> Zaawansowany system tworzenia wykresów używający składni proceduralnej. (Będzie na zajęciach)

Seaborn.object

> Eksperymentalny system wprowadzony w wersji 0.12 używający notacji funkcyjnej i gramatyki grafiki (rozwojowy)

Wybrane dedykowane specjalistyczne nakładki

Pandas i XArray

> Tabelaryczny przetwarzania danych. Posiada funcję *plot()*, pozwalającą na tworzenie ograniczonych wykresów

Yellowbrick

> System analizy i wizualizacji oparty na sci-kit learn. Przede wszystkim wykresy używane w data science

Cartopy

> System przeznaczony do wizualizacji kartograficznych

NetworkX

> System przeznaczony do analizy i wizualizacji sieci i grafów

SciPy

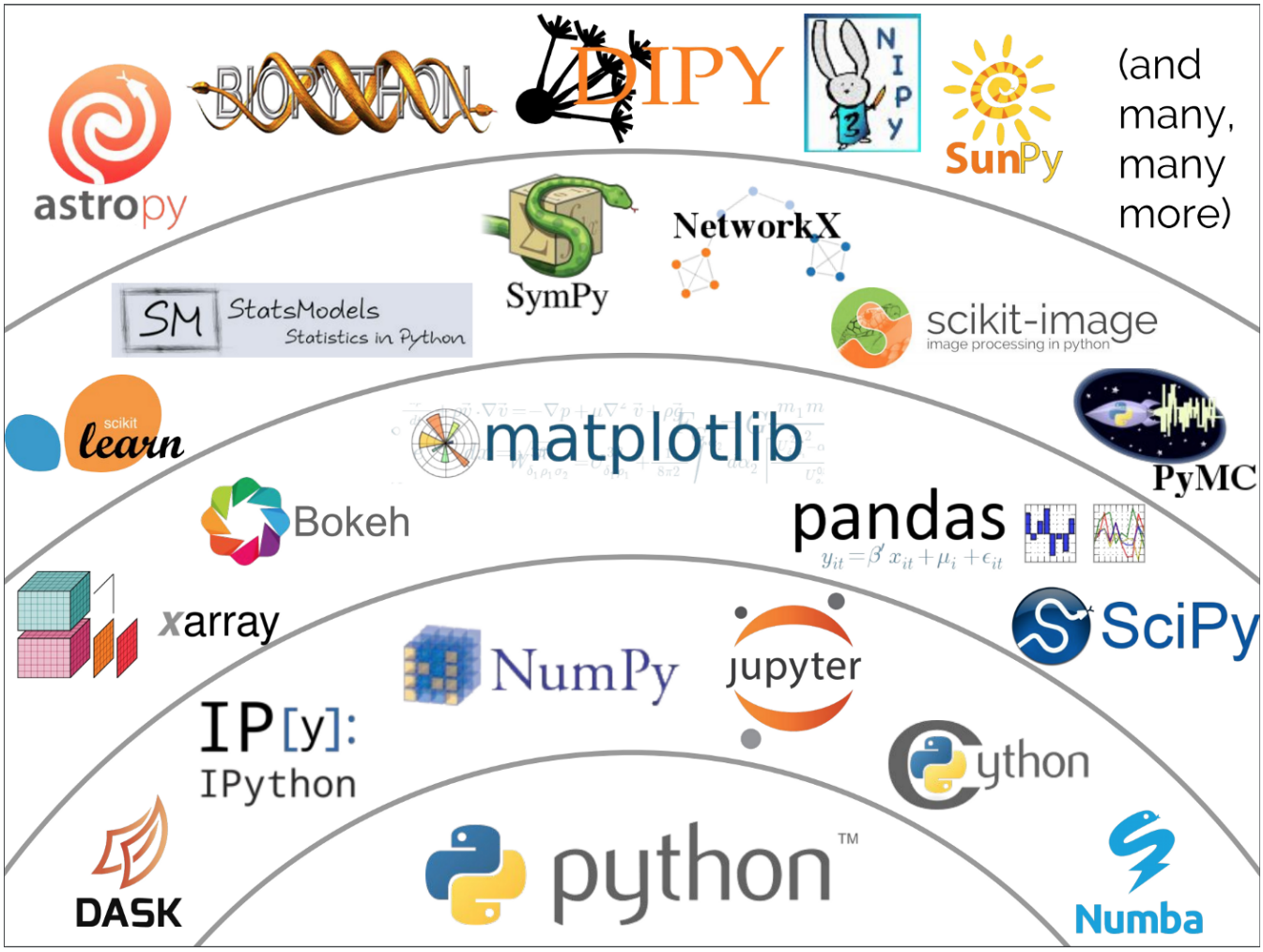
> Dendrogramy

Statsmodel

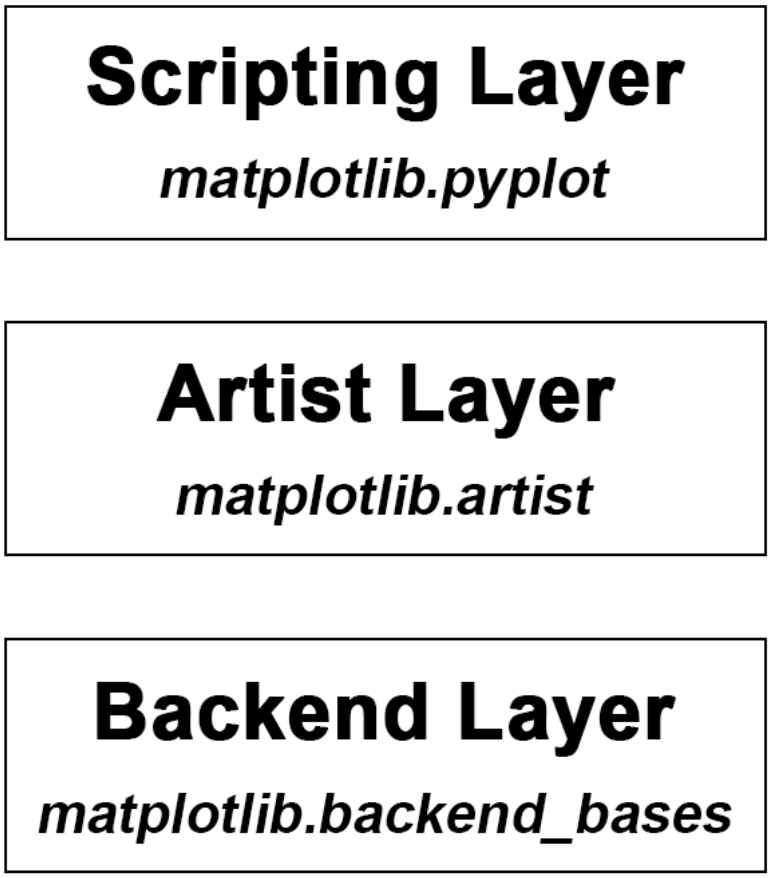
> Alternatywne wykresy podstawowe, wybrane złożone wykresy np. Treemap, mozaikowe dedykowane

> Biblioteki dedykowane do pojedynczych wykresów, np *squiarify*

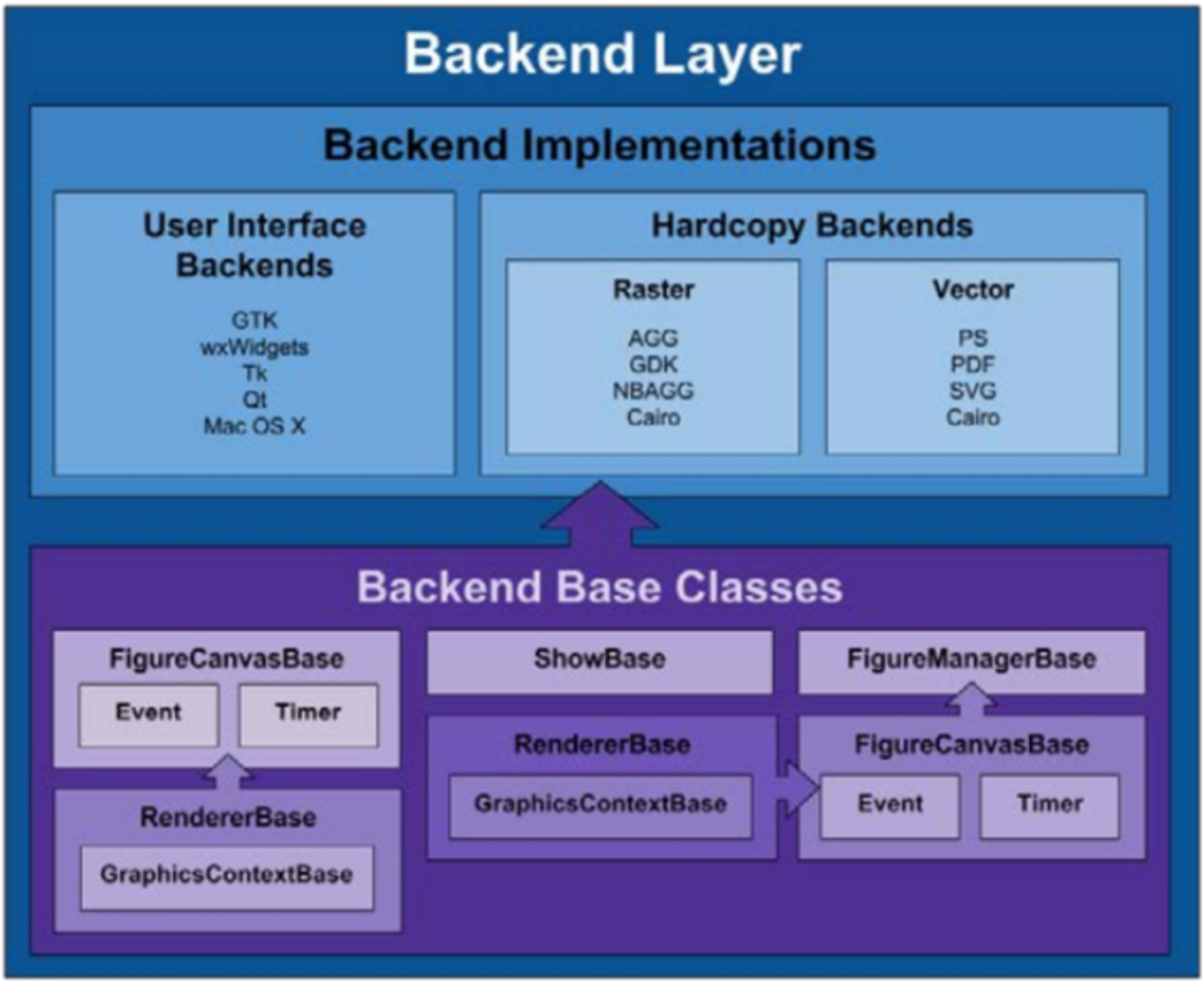
Ekosystem naukowy Pythona



Warstwy przetwarzania danych w matplotlib



Backend Layer



FigureCanvas:

> Deifniuje obszar rysowania (kartka)

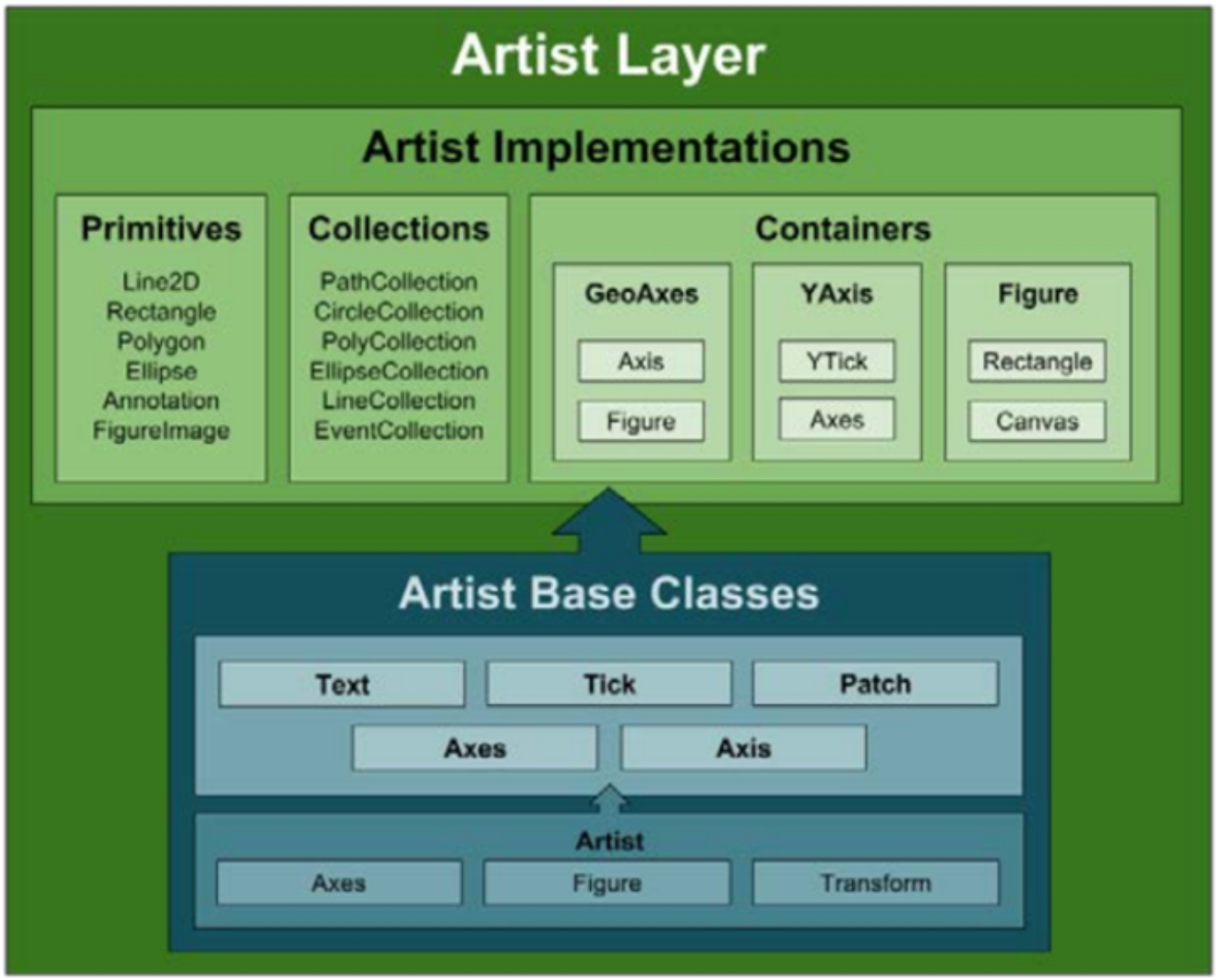
Renderer:

> Narzędzie rysujące (ołówek)

Event:

> Reakcja na działania użytkownika (kiedy rysować)

Artist layer



Obsługuje zamianę opisu wykresów na rysowane elementy

Prymitywy

> linie, prostokąty, elipsy, texty obrazy

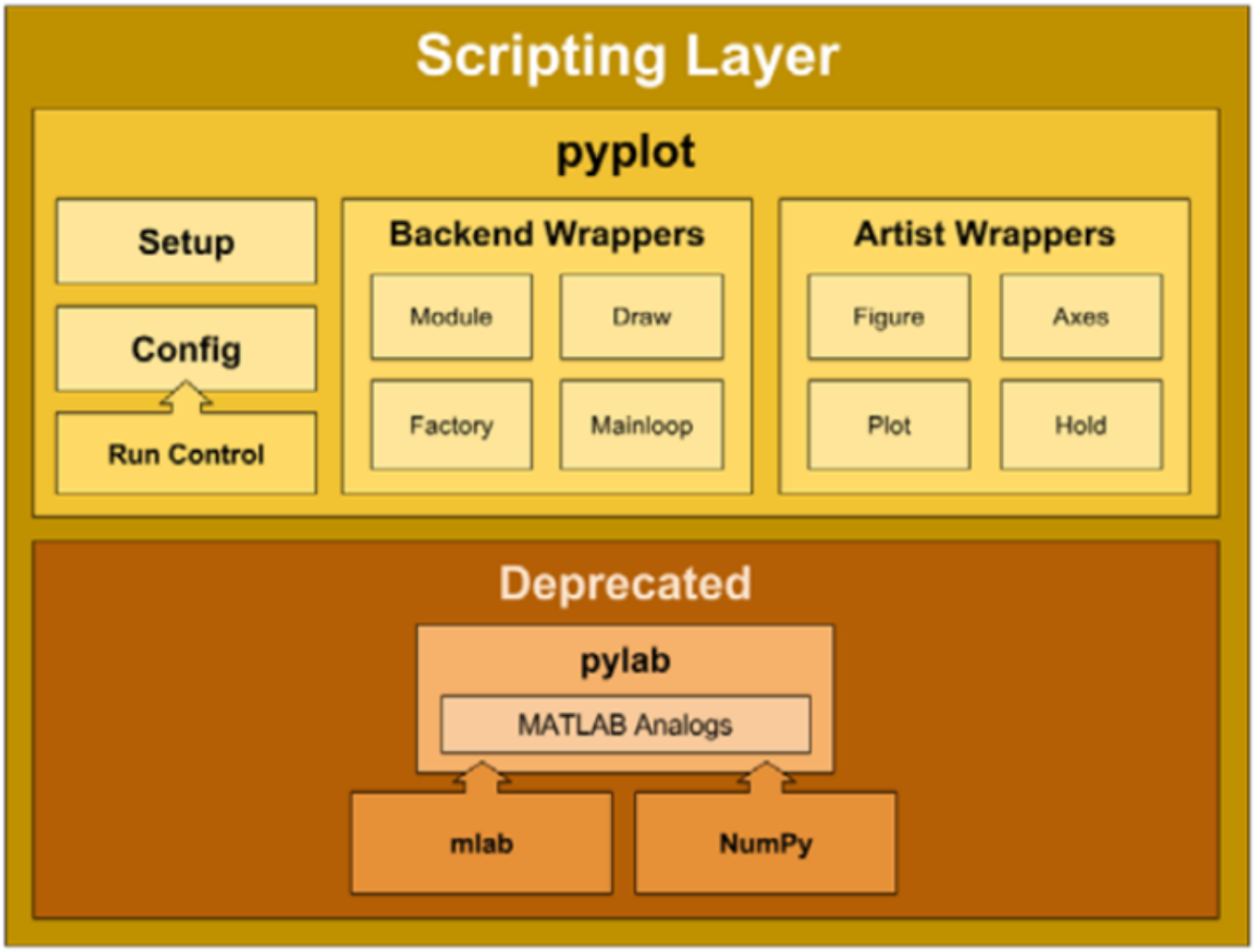
Kolekcje

> złożone zarówno z prymitywów jak i innych klekcji

Typy złożone

> Figura, wykres (axes), osie (axis), znaczniki (ticks)

Scripting layer



Język opsiujący figurę

Manager figury

> rozmiar, ilość i układ wykresów, itp

Przetwarzanie danych

> funkcje rysujące zwracające Artist

Wybór backednu

Sposób definiowania wykresu:

```
fig, ax = plt.subplot() # manager figury
ax.scatter(x[0],x[1]) # skrypt przetwarzania danych
fig.savefig("wykres.pdf") # wybór backendu (pdf)
```

Implementacja

matplotlib jest zaimplementowany w ramach paradygmatu obiektowego (parogramowanie imperatywne), ale jego uproszczona wersja (pylab) imituje działanie procedralne.

Każdy parametr w wykresie musi być określony. Jeżeli nie jest (z reguły *nie* jest) określony przez użytkownika, pobierana jest wartość. Domyślna. Każdy system graficzny posiada jakieś źródło wartości domyślnych. W matplotlib jest to **rcParams** (Run-time configuration). a następnie wykonywane jest rendering.

> Żaden parametr nie może pozostać nieokreślony

Sposób definiowania parametrów domyślnych w matplotlib

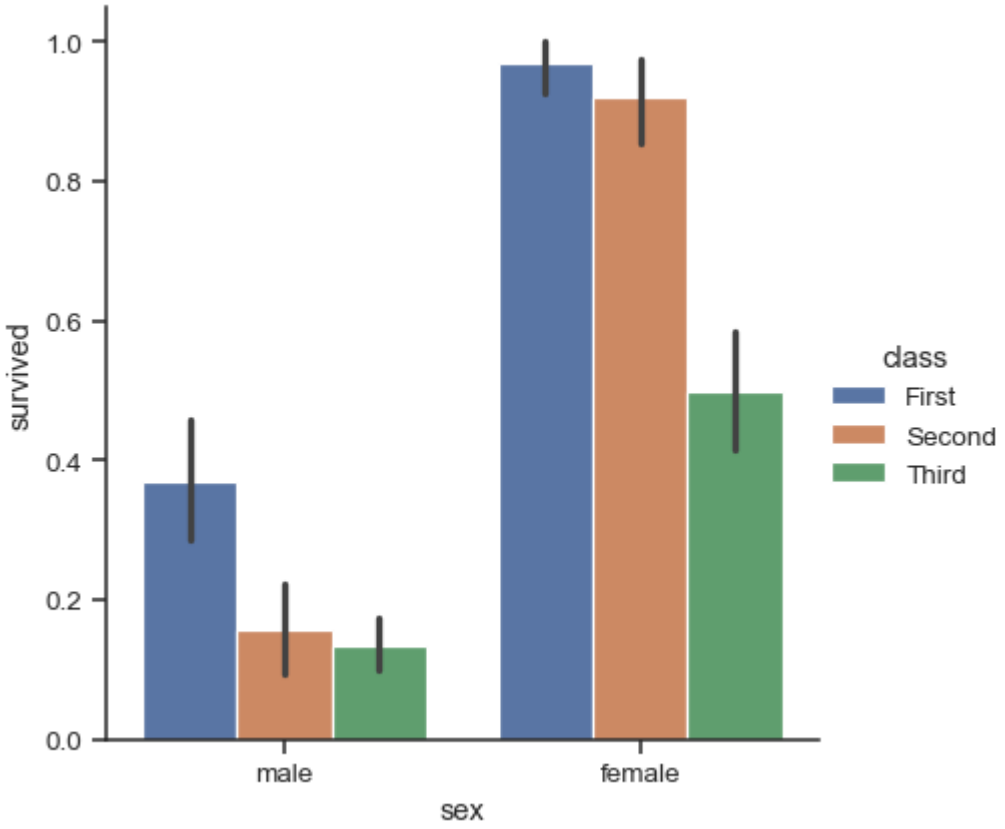
- **stała** - np. kolor tła lub osi
- **zależność funkcyjna** - np. wielkość czcionki zależy od rozmiaru wykresu
- **wartość cykliczna (cycler)** - np. kolor kolejnego obiektu jest pobierany z listy cyklicznej
- **wartość losowa** - pobierana losowo, najczęściej z predefiniowanej listy (nie występują w matplotlib, ale są powszechne np w Excelu)
- **wyliczana** - system graficzny próbuje znaleść optymalną wartość parametru na podstawie analizy danych (z reguły zawodzi)

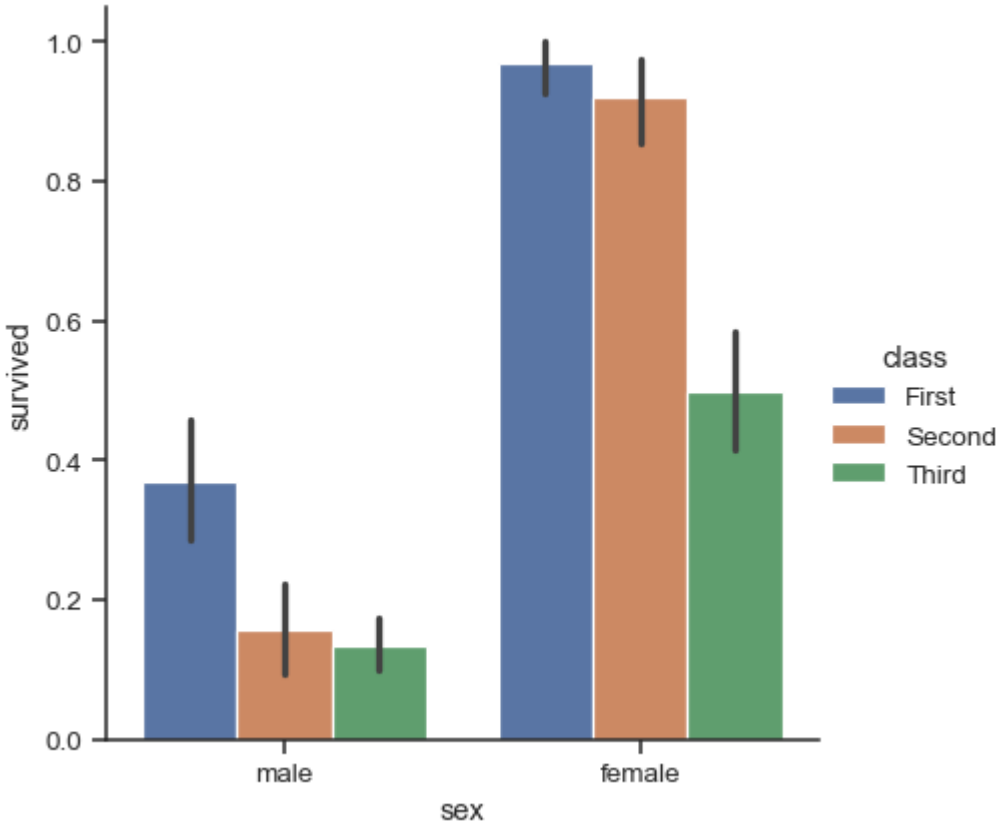
Taksonomia wykresów - wizualizacja wartości

1. Wartości kategoryzowane
2. Wartości ciągłe (1D i 2D)
3. Wartości kategoryzowane i ciągłe
4. Relacje miedzy wartościami
5. Wykresy wielozmienne
6. Wykresy grupujące

Wartości kategoryzowane

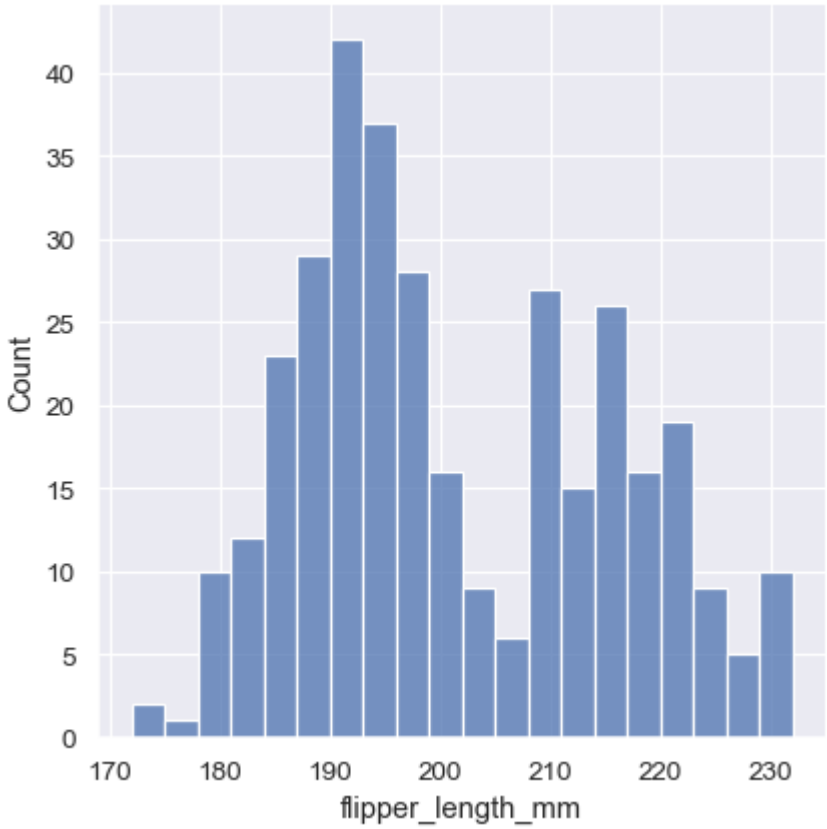
Wykresy mogą pokazywać tylko liczność kategorii

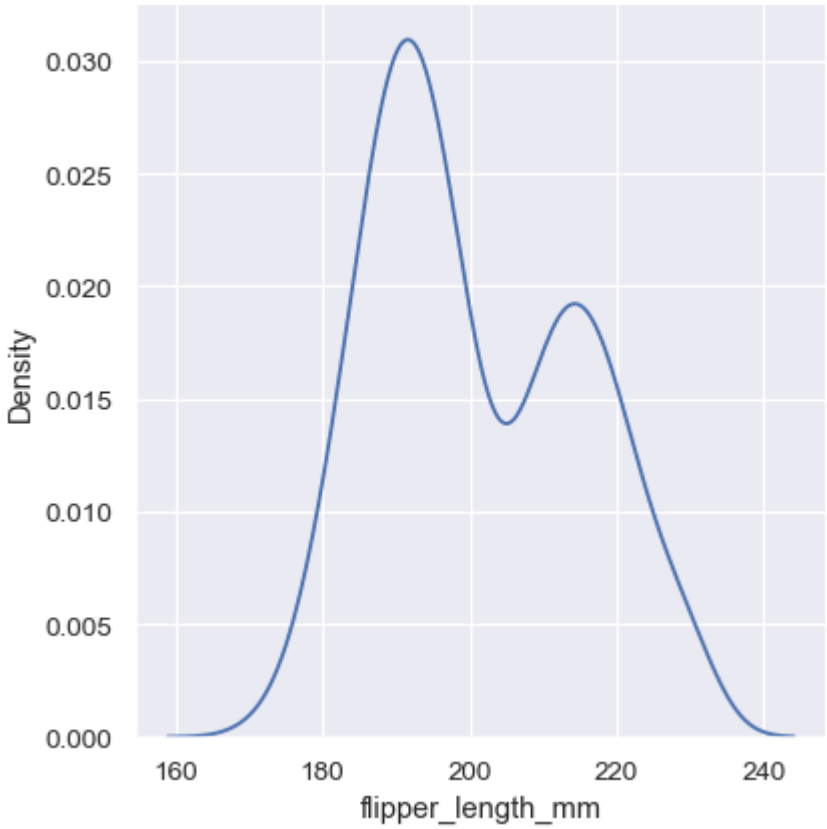


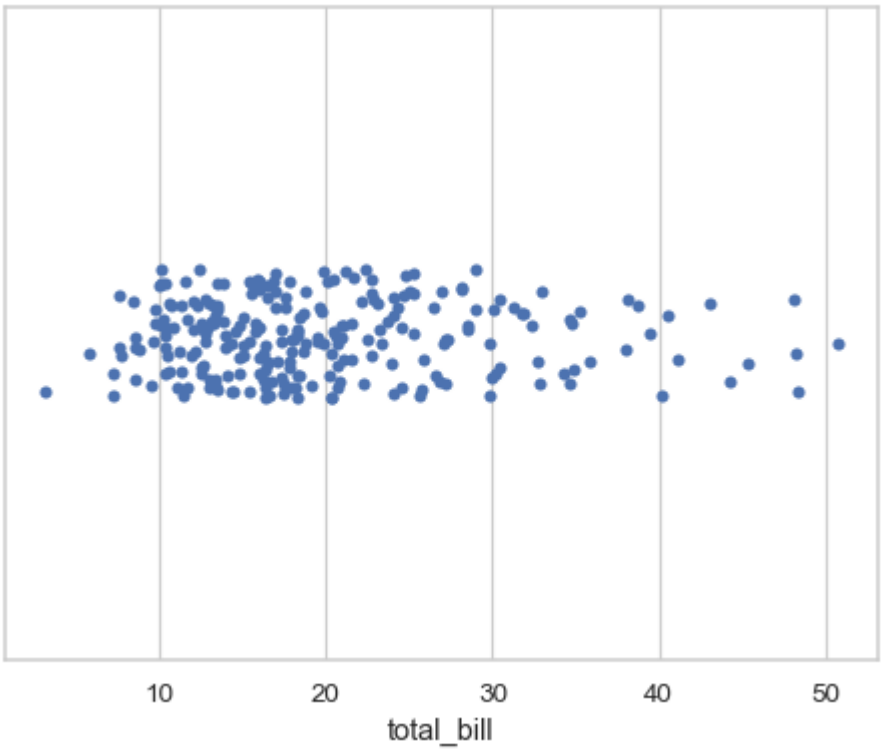


Wartości ciągłe

Wartości ciągłe prezentowane są w formach ilustrujących rozkłady statystyczne

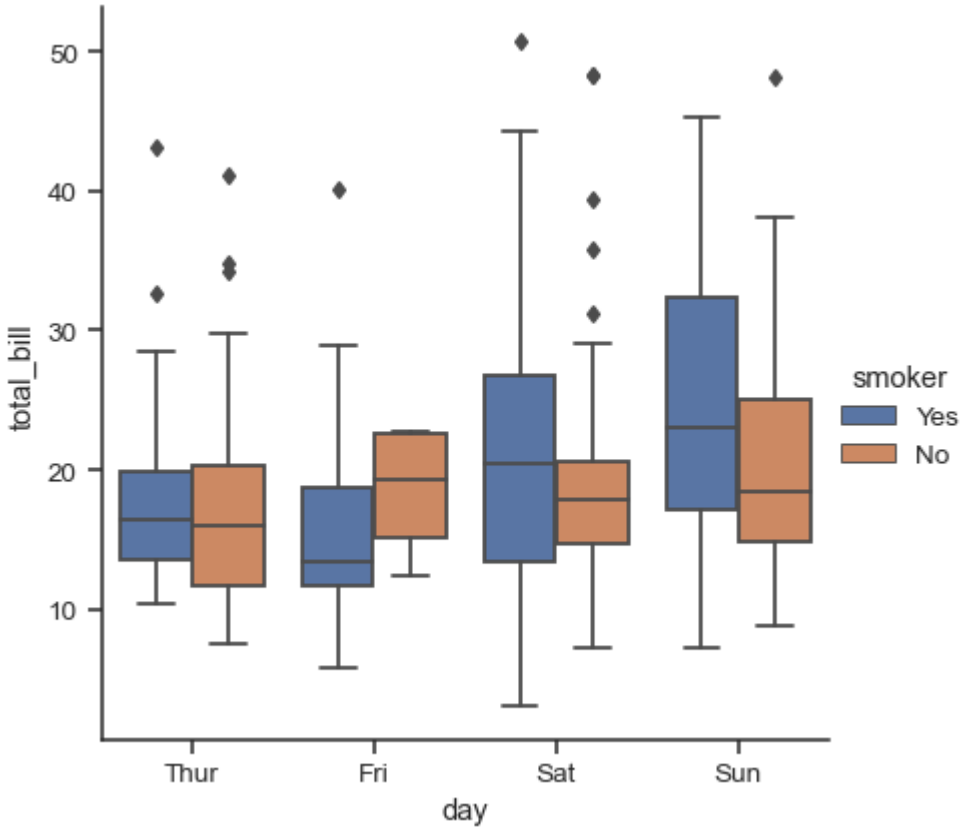

Count
0
5
10
15
20
25
30
35
40
170
180
190
200
210
220
230
flipper_length_mm

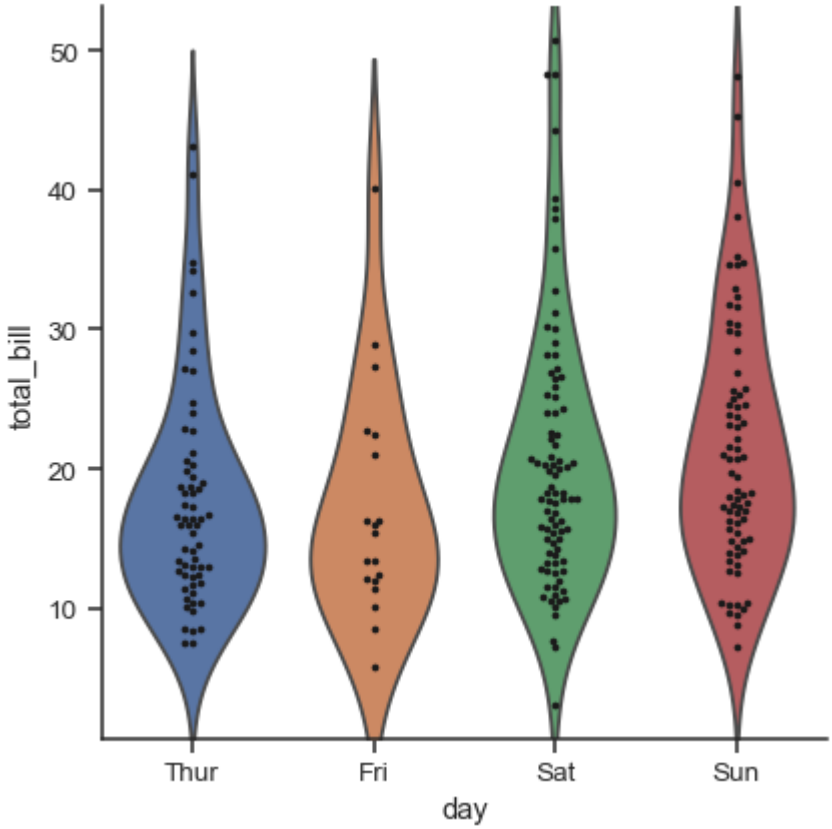

Density
0.000
0.005
0.010
0.015
0.020
0.025
0.030
160
180
200
220
240
flipper_length_mm



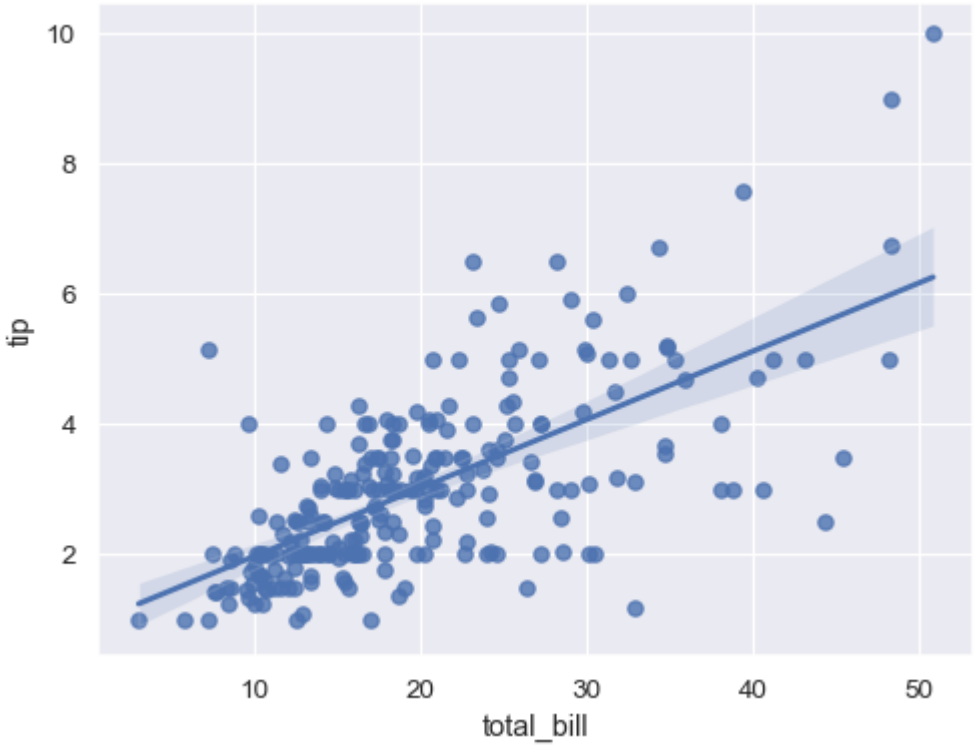
Zmienne ciągłe w obrębie kateogrii

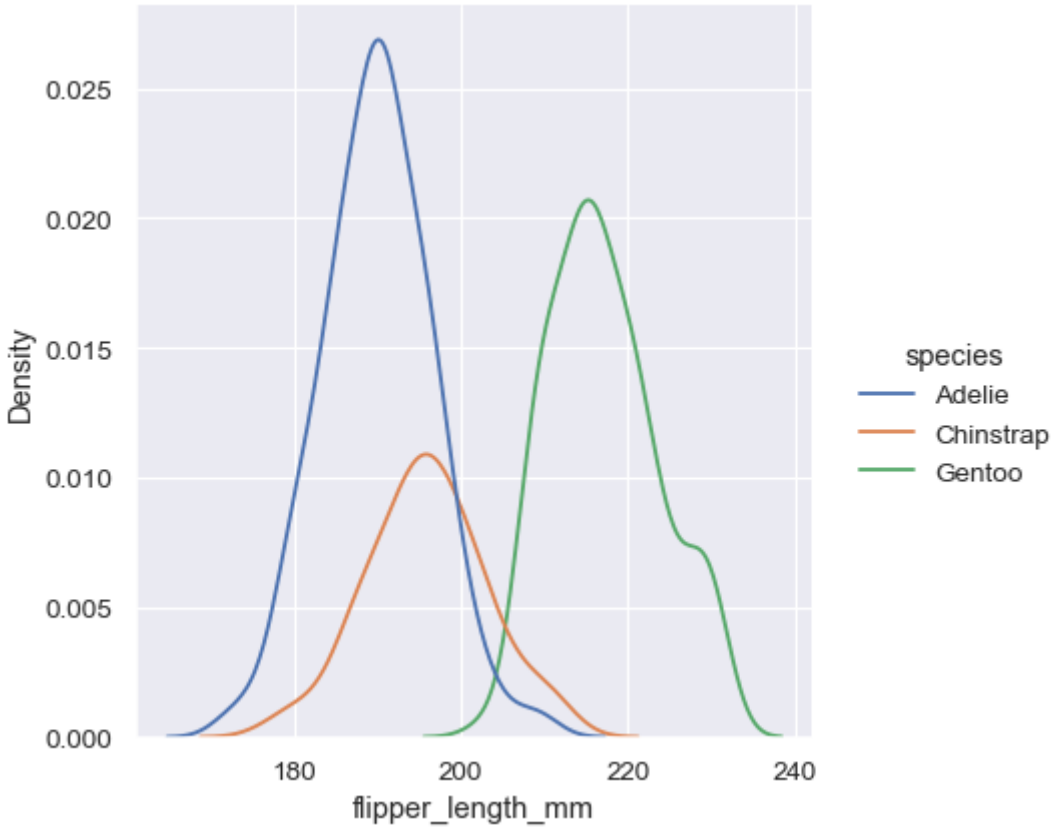
Stosuje się wykresy dla rozkładu podzielone na kategorie

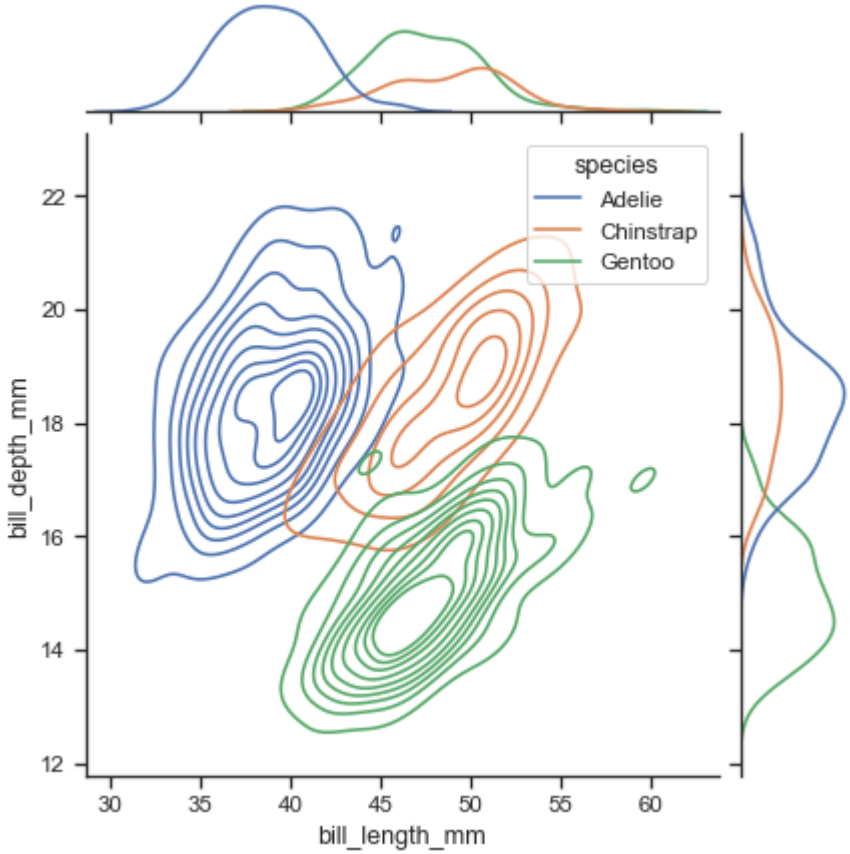




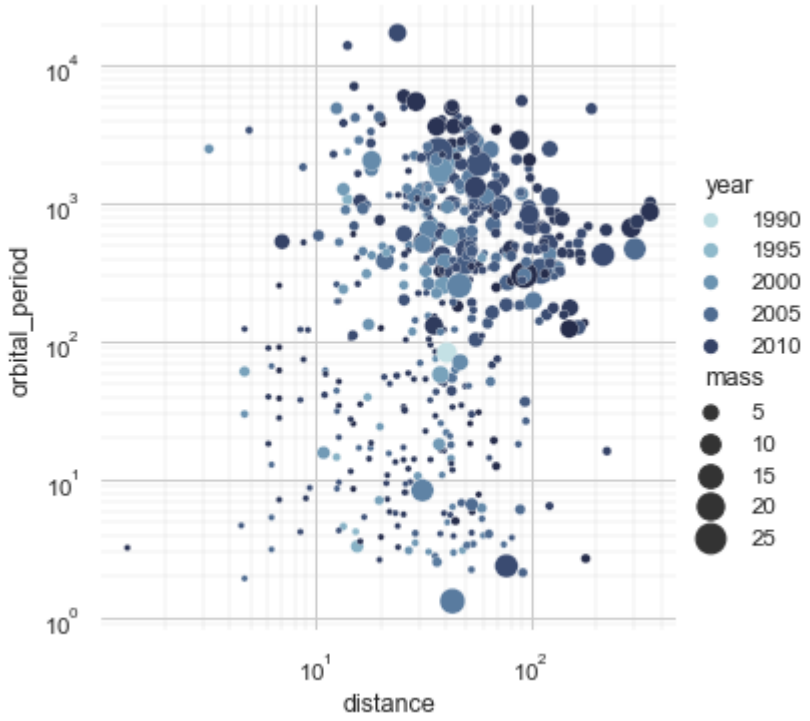
Relacje między wartościami

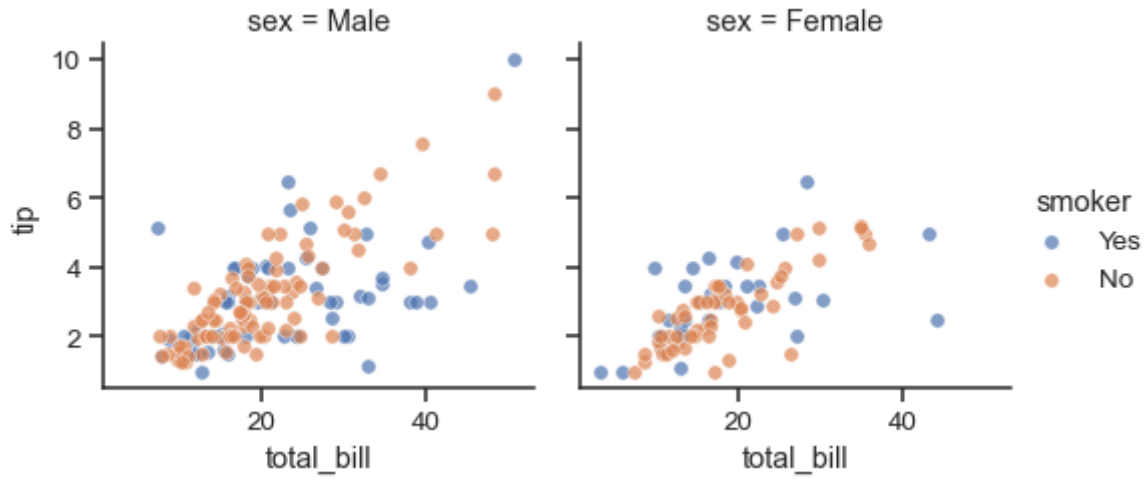

tip
total_bill
10
8
6
4
2
10
20
30
40
50


Density
flipper_length_mm
0.025
0.020
0.015
0.010
0.005
0.000
180
200
220
240
species
Adelie
Chinstrap
Gentoo

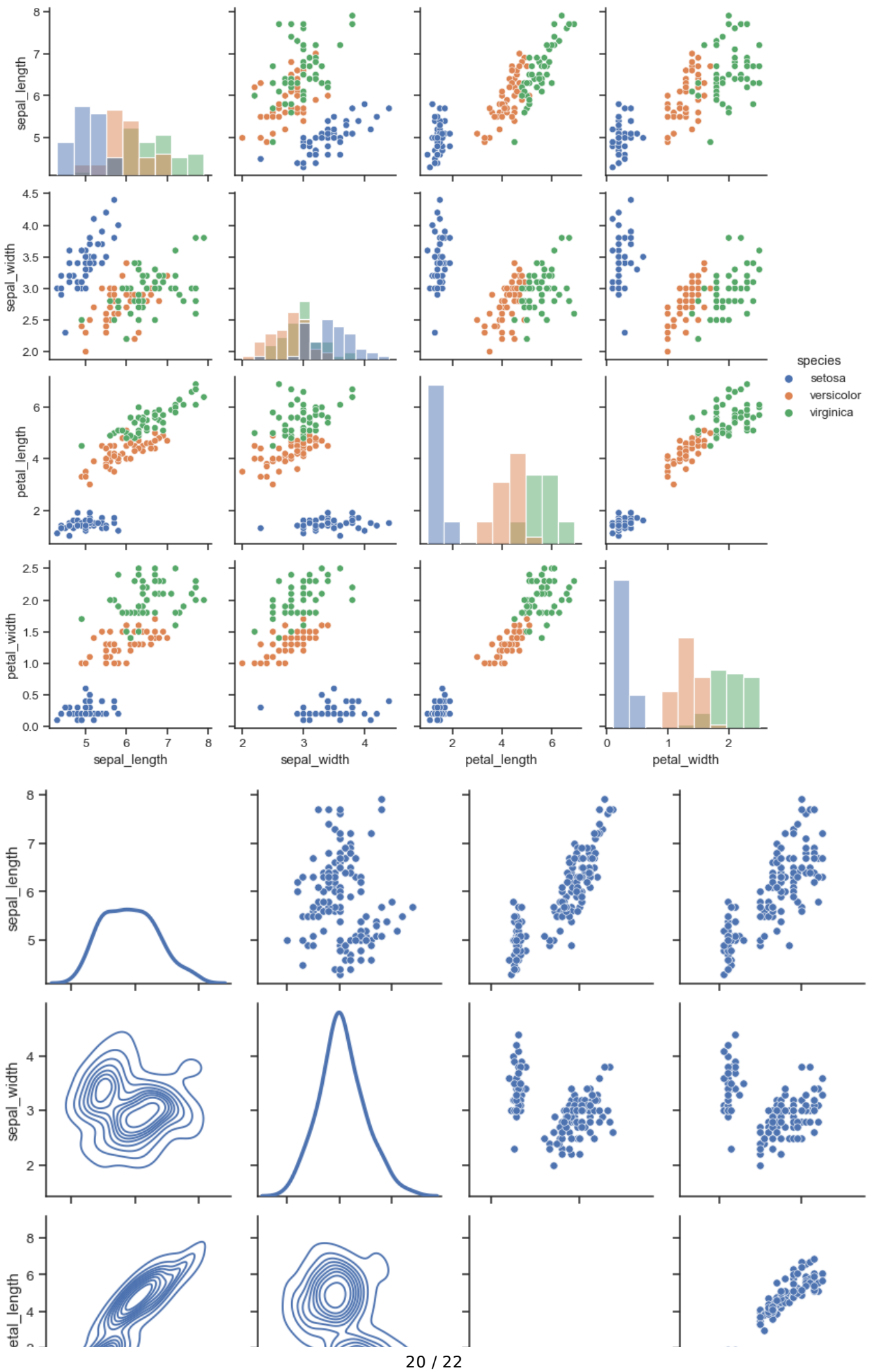


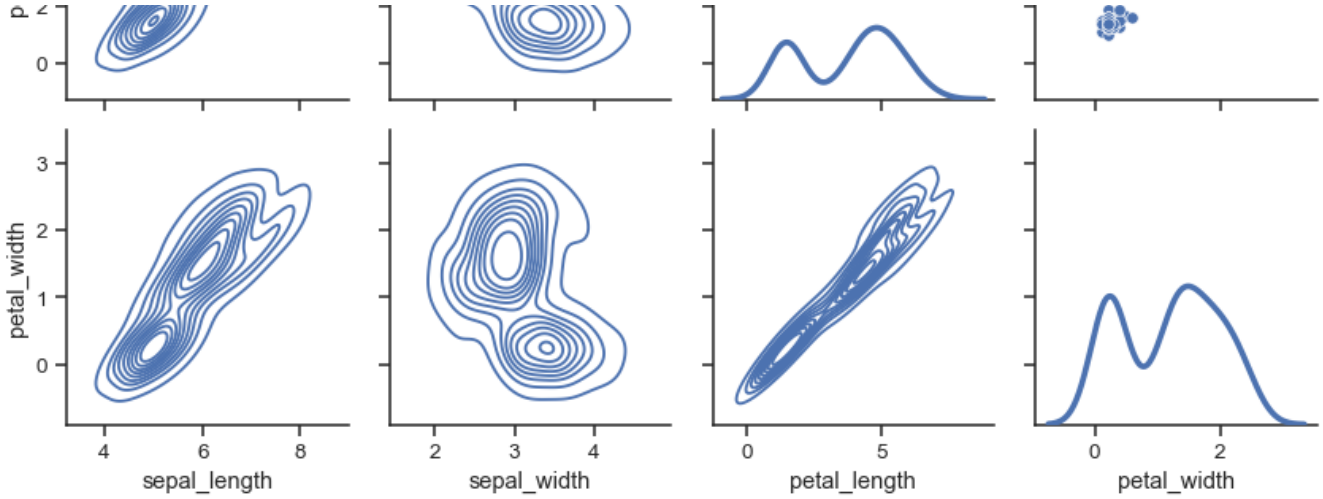
Wiele zmiennych na wykresie



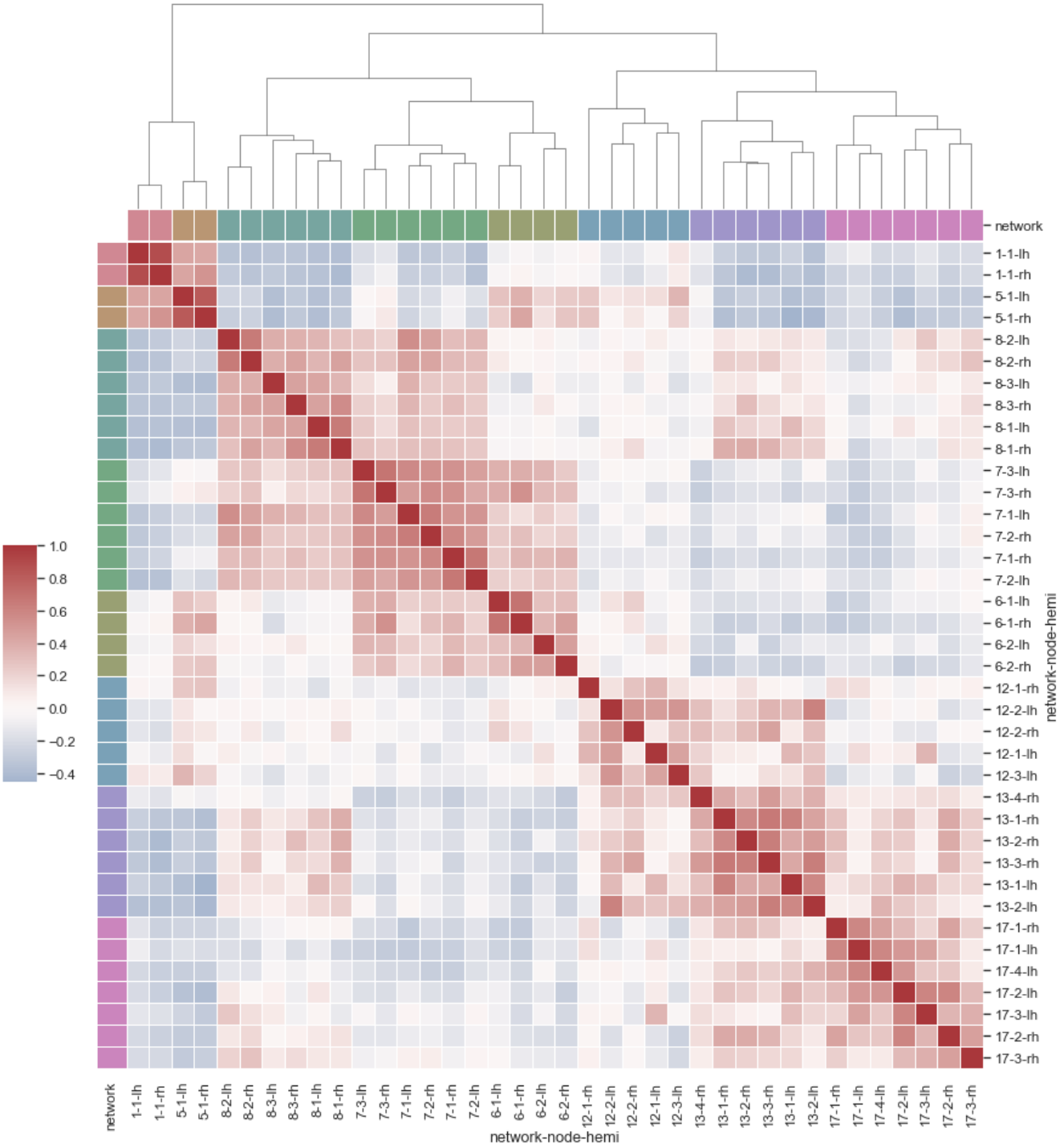


Wykresy wielozmienne (pairplots)


sepal_length
sepal_width
petal_length
petal_width
species
setosa
versicolor
virginica



Wykresy złożone


network
1-1-lh
1-1-rh
5-1-lh
5-1-rh
8-2-lh
8-2-rh
8-3-lh
8-3-rh
8-1-lh
8-1-rh
7-3-lh
7-3-rh
7-1-lh
7-2-rh
7-1-rh
7-2-lh
6-1-lh
6-1-rh
6-2-lh
6-2-rh
12-1-rh
12-2-lh
12-2-rh
12-1-lh
12-3-lh
13-4-rh
13-1-rh
13-2-rh
13-3-rh
13-1-lh
13-2-lh
17-1-rh
17-1-lh
17-4-lh
17-2-lh
17-3-lh
17-2-rh
17-3-rh
network-node-hemi
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4