Bootcamp Data Science
Zajęcia 1

Statystyka

Przemysław Spurek

Główne cele szkolenia

Główne cele szkolenia

- Wstęp do **Nauczania maszynowego**.

Główne cele szkolenia

- Wstęp do **Nauczania maszynowego**.
- Poznanie pojęć potrzebnych w Analizie Danych, Statystyce, Nauczaniu Maszynowym, Sztucznej inteligencji.

Główne cele szkolenia

- Wstęp do **Nauczania maszynowego**.
- Poznanie pojęć potrzebnych w Analizie Danych, Statystyce, Nauczaniu Maszynowym, Sztucznej inteligencji.
- Zrozumienie co to jest model i co oznacza modelowanie statystyczne.

Jakie są wasze CELE?

Dlaczego statystyka?

W życiu codziennym spotykamy się z sytuacjami, w których nie mamy pełnego dostępu do danych, gdzie dane są niekompletne. Czasami zdarzają się odwrotne sytuacje gdzie danych mamy bardzo dużo, ale większość z nich jest zbędna lub trudna do zinterpretowania.

Dlaczego statystyka?

W życiu codziennym spotykamy się z sytuacjami, w których nie mamy pełnego dostępu do danych, gdzie dane są niekompletne. Czasami zdarzają się odwrotne sytuacje gdzie danych mamy bardzo dużo, ale większość z nich jest zbędna lub trudna do zinterpretowania.

Bardzo często bazując na takich informacjach
musimy podejmować decyzje !!!

Przykłady:

Dlaczego statystyka?

W życiu codziennym spotykamy się z sytuacjami, w których nie mamy pełnego dostępu do danych, gdzie dane są niekompletne. Czasami zdarzają się odwrotne sytuacje gdzie danych mamy bardzo dużo, ale większość z nich jest zbędna lub trudna do zinterpretowania.

Bardzo często bazując na takich informacjach
musimy podejmować decyzje !!!

Przykłady:

- Czy powinienem gonić odjeżdżający już samochód?

Dlaczego statystyka?

W życiu codziennym spotykamy się z sytuacjami, w których nie mamy pełnego dostępu do danych, gdzie dane są niekompletne. Czasami zdarzają się odwrotne sytuacje gdzie danych mamy bardzo dużo, ale większość z nich jest zbędna lub trudna do zinterpretowania.

Bardzo często bazując na takich informacjach
musimy podejmować decyzje !!!

Przykłady:

- Czy powinienem gonić odjeżdżający już samochód?
- Które akcje powinienem kupić?

Dlaczego statystyka?

W życiu codziennym spotykamy się z sytuacjami, w których nie mamy pełnego dostępu do danych, gdzie dane są niekompletne. Czasami zdarzają się odwrotne sytuacje gdzie danych mamy bardzo dużo, ale większość z nich jest zbędna lub trudna do zinterpretowania.

Bardzo często bazując na takich informacjach
musimy podejmować decyzje !!!

Przykłady:

- Czy powinienem gonić odjeżdżający już samochód?
- Które akcje powinienem kupić?
- Czy powinienem zażyć pewne lekarstwo?

Dlaczego statystyka?

W życiu codziennym spotykamy się z sytuacjami, w których nie mamy pełnego dostępu do danych, gdzie dane są niekompletne. Czasami zdarzają się odwrotne sytuacje gdzie danych mamy bardzo dużo, ale większość z nich jest zbędna lub trudna do zinterpretowania.

Bardzo często bazując na takich informacjach
musimy podejmować decyzje !!!

Przykłady:

- Czy powinienem gonić odjeżdżający już samochód?
- Które akcje powinienem kupić?
- Czy powinienem zażyć pewne lekarstwo?
- Którego mężczyznę powinnam wziąć z męża?

Dlaczego statystyka?

Niektóre z tych pytań są poza zasięgiem królestwa statystyki ("Którego mężczyznę powinnam wziąć za męża?"), ponieważ zależą od zbyt wielu zmiennych.

Dlaczego statystyka?

Niektóre z tych pytań są poza zasięgiem królestwa statystyki ("Którego mężczyznę powinnam wziąć za męża?"), ponieważ zależą od zbyt wielu zmiennych.
Ale w wielu sytuacjach statystyka może pomóc wydobyć maksimum wiedzy z podanych informacji/danych i jasno wyjaśnić to, co wiemy.

Dlaczego statystyka?

Niektóre z tych pytań są poza zasięgiem królestwa statystyki ("Którego mężczyznę powinnam wziąć za męża?"), ponieważ zależą od zbyt wielu zmiennych.
Ale w wielu sytuacjach statystyka może pomóc wydobyć maksimum wiedzy z podanych informacji/danych i jasno wyjaśnić to, co wiemy.
Na przykład może zmienić nieokreślone stwierdzenia, takie jak

- “To lekarstwo może wywoływać nudności.”
- “Możesz umrzeć, jeśli nie przyjmujesz tego leku."

Dlaczego statystyka?

Niektóre z tych pytań są poza zasięgiem królestwa statystyki ("Którego mężczyznę powinnam wziąć za męża?"), ponieważ zależą od zbyt wielu zmiennych.
Ale w wielu sytuacjach statystyka może pomóc wydobyć maksimum wiedzy z podanych informacji/danych i jasno wyjaśnić to, co wiemy.
Na przykład może zmienić nieokreślone stwierdzenia, takie jak

- “To lekarstwo może wywoływać nudności.”
- “Możesz umrzeć, jeśli nie przyjmujesz tego leku."

na określone

- “Trzy pacjentki na tysiąc doświadczają nudności podczas przyjmowania tego leku.”
- “Jeśli nie przyjmujesz tego leku, istnieje 95% szans na to, że umrzesz."

“German Tank Problem”

Bez statystyk interpretacja danych może doprowadzić do błędnych wniosków, a tym samym podejmowanych decyzji. Weźmy na przykład szacowaną liczbę niemieckich czołgów produkowanych w ciągu II Wojny Światowej, znany również jako “German Tank Problem”.

- `https://en.wikipedia.org/wiki/German_tank_problem`
- `https://www.youtube.com/watch?v=Fzfne9vT624`

Dlaczego statystyka?

Ogólnie statystyka może pomóc w:

Dlaczego statystyka?

Ogólnie statystyka może pomóc w:

- wyjaśnianiu pytań;

Dlaczego statystyka?

Ogólnie statystyka może pomóc w:

- wyjaśnianiu pytań;
- zidentyfikowaniu zmiennych i miar tych zmiennych, które odpowiadają na zadane pytanie;

Dlaczego statystyka?

Ogólnie statystyka może pomóc w:

- wyjaśnianiu pytań;
- zidentyfikowaniu zmiennych i miar tych zmiennych, które odpowiadają na zadane pytanie;
- określeniu wymaganej wielkości próbki;

Dlaczego statystyka?

Ogólnie statystyka może pomóc w:

- wyjaśnianiu pytań;
- zidentyfikowaniu zmiennych i miar tych zmiennych, które odpowiadają na zadane pytanie;
- określeniu wymaganej wielkości próbki;
- opisie różnic;

Dlaczego statystyka?

Ogólnie statystyka może pomóc w:

- wyjaśnianiu pytań;
- zidentyfikowaniu zmiennych i miar tych zmiennych, które odpowiadają na zadane pytanie;
- określeniu wymaganej wielkości próbki;
- opisie różnic;
- określeniu ilościowych miar dotyczących szacowanych parametrów;

Dlaczego statystyka?

Ogólnie statystyka może pomóc w:

- wyjaśnianiu pytań;
- zidentyfikowaniu zmiennych i miar tych zmiennych, które odpowiadają na zadane pytanie;
- określeniu wymaganej wielkości próbki;
- opisie różnic;
- określeniu ilościowych miar dotyczących szacowanych parametrów;
- zaproponowaniu prognozy na podstawie danych.

Mała mobilizacja do pracy

Jednym z ojców statystyki - jak wielu innych rzeczy - jest znany matematyk C.F. Gauss, który powiedział o swojej pracy:

"Ich habe Fleissig sein müssen; Wer e gleichfalls ist, wird eben, weit kommen."

"I had to work hard; if you work hard as well, you, too, will be successful."

Mała mobilizacja do pracy

Jednym z ojców statystyki - jak wielu innych rzeczy - jest znany matematyk C.F. Gauss, który powiedział o swojej pracy:

"Ich habe Fleissig sein müssen; Wer e gleichfalls ist, wird eben, weit kommen."

"I had to work hard; if you work hard as well, you, too, will be successful."

Zachęcam do ciężkiej pracy na zajęciach i wykonania wszystkich zadań własnoręcznie.

Dostępne materiały w zakresie statystyki

W internecie znajdziesz bardzo obszerne informacje o statystyce najczęściej w języku angielskim:

- http://www.statsref.com/
- http://www.vassarstats.net/
- http://www.biostathandbook.com/
- http://onlinestatbook.com/2/index.html
- http://www.itl.nist.gov/div898/handbook/index.htm
- http://wazniak.mimuw.edu.pl/index.php?title=Rachunek_prawdopodobie%C5%84stwa_i_statystyka
- https://www.statlect.com/fundamentals-of-statistics/

Python - podstawy

Na zajęciach powtórzymy sobie podstawowe informacje o Pythonie.
Jeżeli szukasz dodatkowych informacji, to możesz je znaleźć tutaj:

- Python Scientific Lecture Notes (`http://scipy-lectures.github.com`)
- NumPy dla użytkowników Matlaba (`http://mathesaurus.sourceforge.net/matlab-numpy.html`)
- The Python tutorial (`http://docs.python.org/3/tutorial`)

W internecie dostępnych jest też kilka książek:

- `https://python.swaroopch.com/`
- `https://learnpythonthehardway.org/book/`
- `http://greenteapress.com/wp/think-python/`
- `https://www.kevinsheppard.com/images/0/09/Python_introduction.pdf`
- `http://camdavidsonpilon.github.io/Probabilistic-Programming-and-Bayesian-Methods-for-Hackers`

Python - podstawy

https://github.com/przem85/statistic_4/blob/master/D01_Z01_Hello_World.ipynb

```
print('Hello World')
```

Zadanie

Napiszmy skrypt w Pythonie, który wypisze kwadraty cyfr od zera do pięciu.

Zadanie

Napiszmy skrypt w Pythonie, który wypisze kwadraty cyfr od zera do pięciu.

```
def squared(x):
    return x**2
for ii in range(6):
    print(ii, squared(ii))

print('Done')
```

Python - typy danych

https://github.com/przem85/bootcamp/blob/master/statistics/D01_Z02.ipynb

Tuple ( )

Kolekcja elementów różnego typu. Tuple są "niezmienne", tzn. nie można ich modyfikować po utworzeniu.

```
import numpy as np
myTuple = ('abc', np.arange(0,3,0.2), 2.5)
myTuple[2]
```

List []

Kolekcja elementów tego samego typu. Listy są "zmienne"', tzn. ich elementy mogą być modyfikowane. Dlatego listy są zazwyczaj używane do zbierania elementów tego samego typu (liczby, łańcuchy). Zauważmy, że operator "+"łączy listę.

```
import numpy as np
myList = ['abc', 'def', 'ghij']
myList.append('klm')
myList

myList2 = [1,2,3]
myList3 = [4,5,6]
myList2 + myList3
```

Python - typy danych

Arrays []

Tablice służą do pracy z danymi liczbowymi. Python operuje również na obiektach typu `matrix`. Zaleca się jednak stosowanie tablic, ponieważ wiele bibliotek wymaga obiektów typu `array`

Zauważmy, że listy i tablice 1-d różnią się:

- wektory nie można transponować!
- operator "+" dodaje odpowiednie elementy
- operator ".dot" wykonuje mnożenie skalarne

```
import numpy as np
myList2 = [1,2,3]
myList3 = [4,5,6]

myArray2 = np.array(myList2)
myArray3 = np.array(myList3)
myArray2 + myArray3

myArray2.dot(myArray3)
```

Dictionary { }

Słowniki są nieuporządkowanymi strukturami danych (kluczowymi wartościami), w których do elementów odwołujemy się za pomocą: `dict['key']`. Słowniki można utworzyć za pomocą polecenie `dict` lub używając nawiasów klamrowych `{...}` .

```
myDict = dict(one=1, two=2, info='some information')
myDict2 = {'ten':1, 'twenty':20,
          'info':'more information'}
print(myDict['info'])
print(myDict.keys())
```

Python - typy danych

Odwoływanie się do elementów w listach, krotkach czy tablicach jest bardzo proste:

```
a[start:end] # items start through end-1
a[start:] # items start through the rest of the array
a[:end] # items from the beginning through end-1
a[:] # a copy of the whole array
```

Istnieje również wartość kroku, która może być użyta z dowolnym z powyższych podejść:

```
a[start:end:step] # start through not past end, by step
```

Python - typy danych

Zadanie

Napiszmy skrypt który wypisuje:

- pierwsze dwa elementy tablicy,
- wszystkie po za pierwszymi dwoma elementami tablicy,
- tablicę bez ostatnich dwóch elementów.
- elementy znajdujące się na parzystych indeksach.

Python - typy danych

Zadanie

Napiszmy skrypt który wypisuje:

- pierwsze dwa elementy tablicy,
- wszystkie po za pierwszymi dwoma elementami tablicy,
- tablicę bez ostatnich dwóch elementów.
- elementy znajdujące się na parzystych indeksach.

```
import numpy as np
# create a range
x = np.arange(0, 10, 1) # arguments: start, stop, step
print(x)
print(x[0:2])
print(x[:-2])
print(x[2:])
print(x[::2])
```

Python - typy danych

Najważniejsze punkty do zapamiętania to:

- indeksowanie zaczyna się od 0, a nie 1
- wartość `:end` reprezentuje pierwszą wartość, która nie znajduje się w wynikowej tablicy.
- w konsekwencji różnica między `end` a `start` zawiera wybrane elementy (jeśli krok wynosi 1, domyślny).

Python - typy danych

Polecenia najczęściej używane do generowania liczb:

- np.zeros – generator tablicy zawierającej zera

```
import numpy as np
print(np.zeros(3))
print(np.zeros( (2,3) ))
```

- np.ones – generator tablicy zawierającej jedynki

```
import numpy as np
print(np.ones(3))
print(np.ones( (2,3) ))
```

- np.random.randn – generator tablicy zawierającej próbkę z rozkładu normalnego o średniej zero i wariancji 1

```
import numpy as np
print(np.random.randn(10))
```

Python - typy danych

Polecenia najczęściej używane do generowania liczb:

- np.arange – generator szeregu liczb. Możemy podać trzy parametry `start`, `end`, `steppingInterval`. Zauważ, że tablica nie zwiera wartość końcowej (chociaż wygląda to dziwnie, to ma tę zaletę, że kolejne sekwencje mogą być łatwo sklejane.

```
import numpy as np
print(np.arange(3))
print(np.arange(1,3,0.5))
xLow = np.arange(0,3,0.5)
xHigh = np.arange(3,5,0.5)
print(xLow)
print(xHigh)
print(np.concatenate((np.arange(1,3,0.5),
          np.arange(3,5,0.5))))
print(np.arange(1,5,0.5))
```

Python - typy danych

Polecenia najczęściej używane do generowania liczb:

- np.linspace – Generuje liniowo rozmieszczone liczby.

```
import numpy as np
print(np.linspace(0,10,6))
print(np.linspace(2.0, 3.0, num=5, endpoint=False))
print(np.linspace(2.0, 3.0, num=5, retstep=True))
```

- np.array – Generuje tablicę numpy z danych liczbowych.

```
Amat = np.array([ [1, 2], [3, 4] ])
print(Amat)
```

Python - typy danych

Jest kilka specyficznych własności języka python:

- Wektory są po prostu "listami list". Dlatego też pierwszy element zwraca pierwszy wiersz.

```
Amat = np.array([ [1, 2], [3, 4] ])
print(Amat)
print(Amat[0])
```

- Wektor nie jest tym samym co macierz jednowymiarowa, np. wektory nie mogą być transponowane, a macierze mogą.

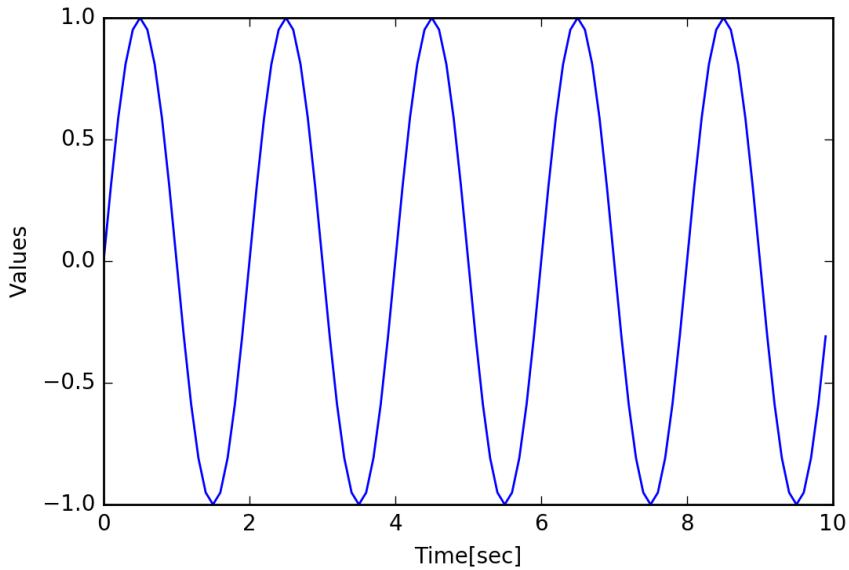
```
x = np.arange(3)
print(x)
print(type(x))
Amat = np.array([ [1,2], [3,4] ])
print(Amat)
print(type(Amat))
print(x.T == x)
print(Amat.T == Amat)
```

Python - podstawy

https://github.com/przem85/bootcamp/blob/master/statistics/D01_Z03.ipynb

Zadanie

Napiszmy skrypt w Python, który wygeneruje punkty w kształcie sinusa i je wyrysuje.



Pandas: Data Structures for Statistics

Pandas – jest pakietem Python, stosowanym w statystyce. Zapewnia on struktury danych odpowiednie do analizy statystycznej oraz udostępnia funkcje ułatwiające wprowadzanie danych, organizację danych i manipulowanie danymi (`http://pandas.pydata.org/`).

Pakiet importujemy za pomocą komendy:

```
import pandas as pd
```

- W statystycznej analizie danych, struktury danych okazały się niezwykle przydatne. Aby obsługiwać etykietowane dane w Pythonie, Pandas wprowadził obiekty typu **DataFrame** (ramka danych).
- Ramka danych jest dwuwymiarową strukturą o kolumnach zawierających potencjalnie różne typy danych. Można o nich myśleć, jak o arkuszu kalkulacyjnym lub tabeli SQL.

DataFrame

https://github.com/przem85/bootcamp/blob/master/statistics/D01_Z04.ipynb

Zacznijmy od konkretnego przykładu i stwórzmy ramkę danych (DataFrame) z trzema kolumnami: "Time", "x" i "y":

```
import numpy as np
import pandas as pd
t = np.arange(0,10,0.1)
x = np.sin(t)
y = np.cos(t)
df = pd.DataFrame({'Time':t, 'x':x, 'y':y})
print(df)
```

DataFrame

- Aby wybrać jedną kolumnę możemy odnieść się do niej za pomocą nazwy lub indeksu.

```
print(df.Time)
print(df['Time'])
```

DataFrame

- Aby wybrać jedną kolumnę możemy odnieść się do niej za pomocą nazwy lub indeksu.

```
print(df.Time)
print(df['Time'])
```

- Aby wybrać dwie kolumny.

```
data = df[['Time', 'y']]
```

DataFrame

- Aby wybrać jedną kolumnę możemy odnieść się do niej za pomocą nazwy lub indeksu.

```
print(df.Time)
print(df['Time'])
```

- Aby wybrać dwie kolumny.

```
data = df[['Time', 'y']]
```

- Aby wyświetlić pierwsze/ostatnie wiersze

```
data.head()
data.tail()
```

DataFrame

- Aby wybrać jedną kolumnę możemy odnieść się do niej za pomocą nazwy lub indeksu.

```
print(df.Time)
print(df['Time'])
```

- Aby wybrać dwie kolumny.

```
data = df[['Time', 'y']]
```

- Aby wyświetlić pierwsze/ostatnie wiersze

```
data.head()
data.tail()
```

- Aby wyświetlić wiersze od 5 do 10 (10-4 =6.)

```
data[4:10]
```

DataFrame

- Aby wyświetlić wybrane kolumny oraz wiersze

```
df[['Time', 'y']][4:10]
```

DataFrame

- Aby wyświetlić wybrane kolumny oraz wiersze

```
df[['Time', 'y']][4:10]
```

- Można używać też standardowej notacji

```
df.iloc[4:10, [0,2]]
```

DataFrame

- Aby wyświetlić wybrane kolumny oraz wiersze

```
df[['Time', 'y']][4:10]
```

- Można używać też standardowej notacji

```
df.iloc[4:10, [0,2]]
```

- Możemy uzyskać bezpośredni dostęp do danych konwertując obiekt do numpy array

```
data.values
```

DataFrame

DataFrame (ramka danych) z pakietu `pandas` jest bardzo podobnym obiektem do tablicy w numpy (numpy array), ale **filozofie** stojące za nimi są kompletnie różne.

- **numpy** – koncentruje się na wierszach (np. `data[0]` jest pierwszym wierszem)
- **pandas** – koncentruje się na kolumnach (np. `df['values'][0]` jest pierwszym elementem kolumny o nazwie `'values'`)

Wczytywanie danych

Może to być zaskakujące, ale importowanie danych w prawidłowym formacie i sprawdzanie błędnych lub brakujących wpisów jest często jedną z najbardziej czasochłonnych części analizy danych.

Gdy dane są dostępne jako pliki ASCII, zawsze należy zacząć od sprawdzenia kilku szczegółów! W szczególności należy sprawdzić:

Wczytywanie danych

Może to być zaskakujące, ale importowanie danych w prawidłowym formacie i sprawdzanie błędnych lub brakujących wpisów jest często jedną z najbardziej czasochłonnych części analizy danych.

Gdy dane są dostępne jako pliki ASCII, zawsze należy zacząć od sprawdzenia kilku szczegółów! W szczególności należy sprawdzić:

- Czy dane mają nagłówek i/lub stopkę?

Wczytywanie danych

Może to być zaskakujące, ale importowanie danych w prawidłowym formacie i sprawdzanie błędnych lub brakujących wpisów jest często jedną z najbardziej czasochłonnych części analizy danych.

Gdy dane są dostępne jako pliki ASCII, zawsze należy zacząć od sprawdzenia kilku szczegółów! W szczególności należy sprawdzić:

- Czy dane mają nagłówek i/lub stopkę?
- Czy na końcu pliku są puste wiersze?

Wczytywanie danych

Może to być zaskakujące, ale importowanie danych w prawidłowym formacie i sprawdzanie błędnych lub brakujących wpisów jest często jedną z najbardziej czasochłonnych części analizy danych.

Gdy dane są dostępne jako pliki ASCII, zawsze należy zacząć od sprawdzenia kilku szczegółów! W szczególności należy sprawdzić:

- Czy dane mają nagłówek i/lub stopkę?
- Czy na końcu pliku są puste wiersze?
- Czy przed pierwszym elementem występuje jakiś "biały znak"?

Wczytywanie danych

Może to być zaskakujące, ale importowanie danych w prawidłowym formacie i sprawdzanie błędnych lub brakujących wpisów jest często jedną z najbardziej czasochłonnych części analizy danych.

Gdy dane są dostępne jako pliki ASCII, zawsze należy zacząć od sprawdzenia kilku szczegółów! W szczególności należy sprawdzić:

- Czy dane mają nagłówek i/lub stopkę?
- Czy na końcu pliku są puste wiersze?
- Czy przed pierwszym elementem występuje jakiś "biały znak"?
- Czy na końcu każdej linii występuje jakiś "biały znak"?

Wczytywanie danych

Może to być zaskakujące, ale importowanie danych w prawidłowym formacie i sprawdzanie błędnych lub brakujących wpisów jest często jedną z najbardziej czasochłonnych części analizy danych.

Gdy dane są dostępne jako pliki ASCII, zawsze należy zacząć od sprawdzenia kilku szczegółów! W szczególności należy sprawdzić:

- Czy dane mają nagłówek i/lub stopkę?
- Czy na końcu pliku są puste wiersze?
- Czy przed pierwszym elementem występuje jakiś "biały znak"?
- Czy na końcu każdej linii występuje jakiś "biały znak"?
- Czy dane są oddzielone tabulatorami i/lub spacjami?

Wczytywanie danych

https://github.com/przem85/bootcamp/blob/master/statistics/D01_Z05.ipynb

Przykład 1

0,1.0,1.3,0.6
1,2.0,2.1,0.7
2,3.0,4.8,0.8
3,4.0,3.3,0.9

```
data = np.loadtxt('data.txt', delimiter=',')
print(data)
```

```
[[ 0.   1.   1.3  0.6]
 [ 1.   2.   2.1  0.7]
 [ 2.   3.   4.8  0.8]
 [ 3.   4.   3.3  0.9]]
```

Wczytywanie danych

Przykład 1

0,1.0,1.3,0.6
1,2.0,2.1,0.7
2,3.0,4.8,0.8
3,4.0,3.3,0.9

```
df = pd.read_csv('data.txt', header=None)
df.head()
```

	0	1	2	3
0	0	1.0	1.3	0.6
1	1	2.0	2.1	0.7
2	2	3.0	4.8	0.8
3	3	4.0	3.3	0.9

```
df.values
```

Wczytywanie danych

Przykład 1

0,1.0,1.3,0.6
1,2.0,2.1,0.7
2,3.0,4.8,0.8
3,4.0,3.3,0.9

```
df = pd.read_csv('data.txt')
df
```

	0	1.0	1.3	0.6
0	1	2.0	2.1	0.7
1	2	3.0	4.8	0.8
2	3	4.0	3.3	0.9

```
df.values
```

Wczytywanie danych

Przykład 2

ID,Weight,Value
1.0,1.3,0.6
2.0,2.1,0.7
3.0,4.8,0.8
4.0,3.3,0.9

```
data = np.loadtxt('data2.txt', delimiter=',')
data
```

Wczytywanie danych

Przykład 2

ID,Weight,Value
1.0,1.3,0.6
2.0,2.1,0.7
3.0,4.8,0.8
4.0,3.3,0.9

```
df = pd.read_csv('data2.txt', header=None)
df
```

	0	1	2
0	ID	Weight	Value
1	1.0	1.3	0.6
2	2.0	2.1	0.7
3	3.0	4.8	0.8
4	4.0	3.3	0.9

```
df.values
```

Wczytywanie danych

Przykład 2

ID,Weight,Value
1.0,1.3,0.6
2.0,2.1,0.7
3.0,4.8,0.8
4.0,3.3,0.9

```
df = pd.read_csv('data2.txt')
df
```

	ID	Weight	Value
0	1.0	1.3	0.6
1	2.0	2.1	0.7
2	3.0	4.8	0.8
3	4.0	3.3	0.9

```
df.values
```

Wczytywanie danych

Przykład 3

ID, Weight, Value
1, 1.3, 0.6
2, 2.1, 0.7
3, 4.8, 0.8
4, 3.3, 0.9
blad
Ja, 2017

```
df = pd.read_csv('data3.txt',engine='python'
        ,skipfooter=3, delimiter='[ ,]*')
```

	ID	Weight	Value
0	1.0	1.3	0.6
1	2.0	2.1	0.7
2	3.0	4.8	0.8
3	4.0	3.3	0.9

```
df.values
```

Wczytywanie danych

Ostatnia zmienna, `separator = '[,] *'` to wyrażenie regularne (regular expression), określające, że do oddzielenia wartości wpisu można użyć "co najmniej jednej spacji i/lub przecinków".

Wczytywanie danych

Ostatnia zmienna, `separator = '[,] *'` to wyrażenie regularne (regular expression), określające, że do oddzielenia wartości wpisu można użyć "co najmniej jednej spacji i/lub przecinków".

- Praca z danymi tekstowymi często wymaga użycia prostych wyrażeń regularnych.
- Wyrażenia regularne to bardzo potężny sposób wyszukiwania i/lub manipulowania ciągami tekstowymi.
- W internecie można znaleźć bardzo dużo informacji na ten temat:
  - `https://www.debuggex.com/cheatsheet/regex/python`
  - `http://www.regular-expressions.info/`

Wczytywanie danych

https://github.com/przem85/bootcamp/blob/master/statistics/D01_Z06.ipynb

Przykłady użycia wyrażeń regularnych:

Wczytywanie danych

https://github.com/przem85/bootcamp/blob/master/statistics/D01_Z06.ipynb

Przykłady użycia wyrażeń regularnych:

- Wyodrębnianie kolumn z pewnymi wzorami nazw z DataFrame. W poniższym przykładzie wyodrębnimy wszystkie kolumny zaczynające się od `Vel`:

```
data = np.round(np.random.randn(100,7), 2)
df = pd.DataFrame(data, columns=['Time','PosX',
        'PosY', 'PosZ', 'VelX', 'VelY', 'VelZ'])
vel = df.filter(regex='Vel*')
vel.head()
```

Wczytywanie danych

Przykład w Jupyter

https://github.com/przem85/bootcamp/blob/master/statistics/D01_Z09.ipynb

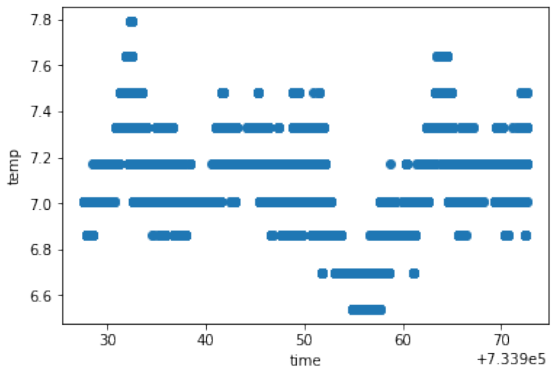
Wczytywanie danych

https://github.com/przem85/bootcamp/blob/master/statistics/D01_Z07.ipynb

Pobierzmy plik: https://github.com/PyHOGS/pyhogs-code/blob/master/data/CA2009.mat

Zadanie

Proszę odczytać dane 'time' oraz 'temp' oraz je narysować.



Wczytywanie danych

```
print("t['data'][0,0]['temp'] is a: "
        + dtype_shape_str(t['data'][0,0]['time']))
print("t['data'][0,0]['temp'][0,0] is a: "
        + dtype_shape_str(t['data'][0,0]['time'][0,0]))

time = t['data'][0,0]['time'][0,0]
temp = t['data'][0,0]['temp'][0,0]
plt.plot(time,temp,'o')
plt.xlabel('time')
plt.ylabel('temp')
plt.show()
```

Wczytywanie danych

https://github.com/przem85/bootcamp/blob/master/statistics/D01_Z10.ipynb

Zadanie

Wczytaj i przetwórz dane do postaci numerycznej.

Wizualizacja danych

- Jednym z pierwszych kroków w analizie danych jest ich wizualizacja. Wyświetlanie danych jest pomocne w znajdowaniu elementów ekstremalnych/odstających, które często są spowodowane błędami w gromadzeniu danych.
- Wybór odpowiedniej procedury statystycznej zależy od typu danych. Dane mogą być:
  - kategoryczne
  - liczbowe.

Wizualizacja danych

Dane kategoryczne:

- Dane typu Boolean to dane, które mogą mieć tylko dwie możliwe wartości, np. Kobieta/mężczyzna, palacz/osoba niepaląca, prawda/fałsz.
- Dane kategoryczne to takie, w których występują więcej niż dwie kategorie, np. żonaty/singiel/rozwiedziony.
- Dane typu porządkowego, w przeciwieństwie do danych nominalnych są uporządkowane i mają logiczną sekwencję, np. bardzo niewielu/kilku/niektórzy/wielu/bardzo wielu.

Wizualizacja danych

Dane liczbowe:

- Dane ciągłe to takie, które mogą przyjąć dowolną wartość rzeczywistą.
- Dane dyskretne to takie, które przyjmują tylko wartości całkowite, np. ilość dzieci: 0, 1, 2, 3, 4, 5, 6, ...

Wizualizacja danych

https://github.com/przem85/bootcamp/blob/master/statistics/D01_Z08.ipynb

Wczytaj dane z pliku: http://ww2.amstat.org/publications/jse/datasets/babyboom.dat.txt

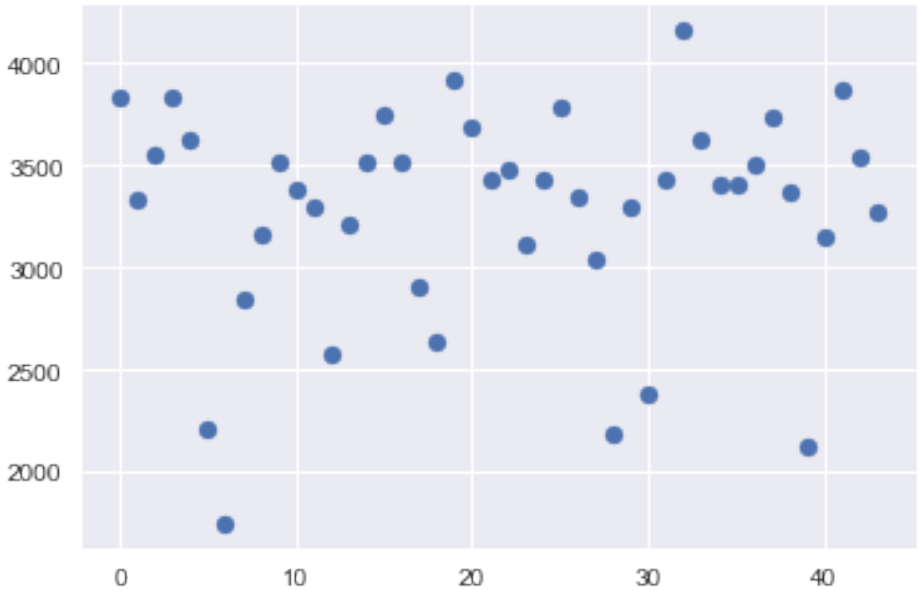
- Wczytujemy dane

```
inFile = 'http://ww2.amstat.org/publications/jse/data
data = pd.read_csv(inFile, sep='[ ]*', header=None,
          engine='python',
          names= ['sex', 'Weight', 'Minutes'])
df = data[['Minutes', 'sex', 'Weight']]
```

Wizualizacja danych

- Narysujmy dane `df.Weight.values` jako "szereg czasowy":

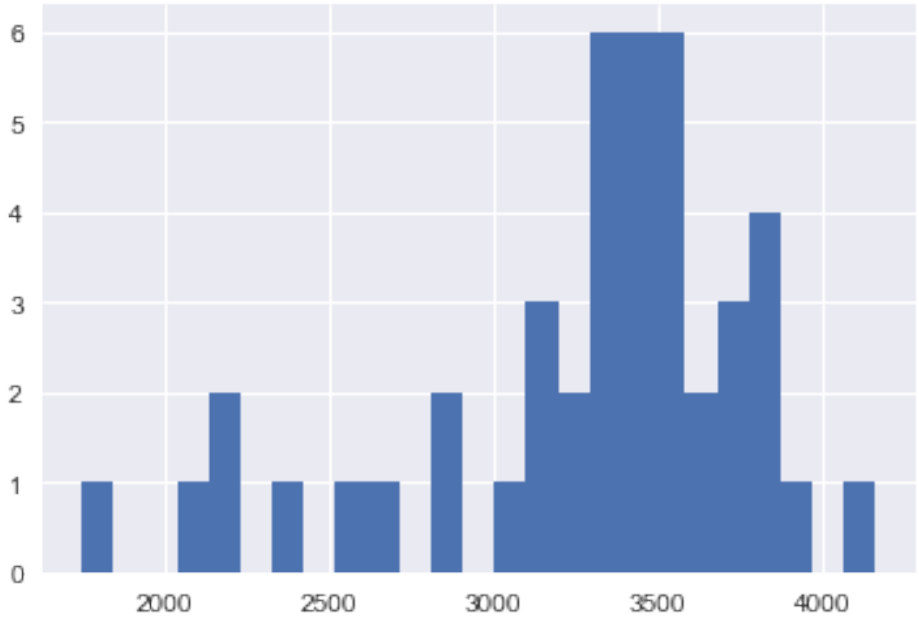
```
x=df.Weight.values
plt.scatter(np.arange(len(x)), x)
plt.show()
```



Wizualizacja danych

- Narysujmy histogram dla `df.Weight.values`:

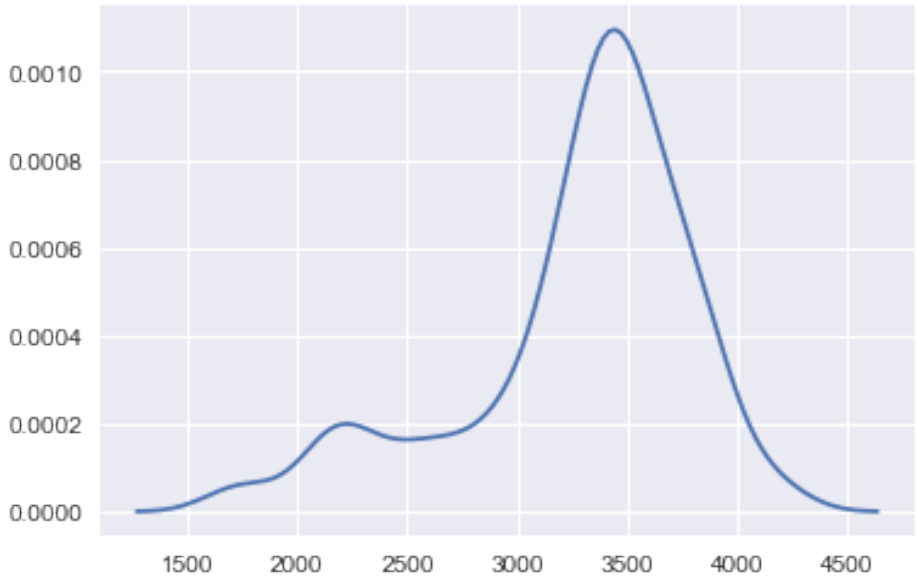
```
plt.hist(x, bins=25)
plt.show()
```



Wizualizacja danych

- Narysujmy gęstość dla `df.Weight.values` (estymacja jądrowa):

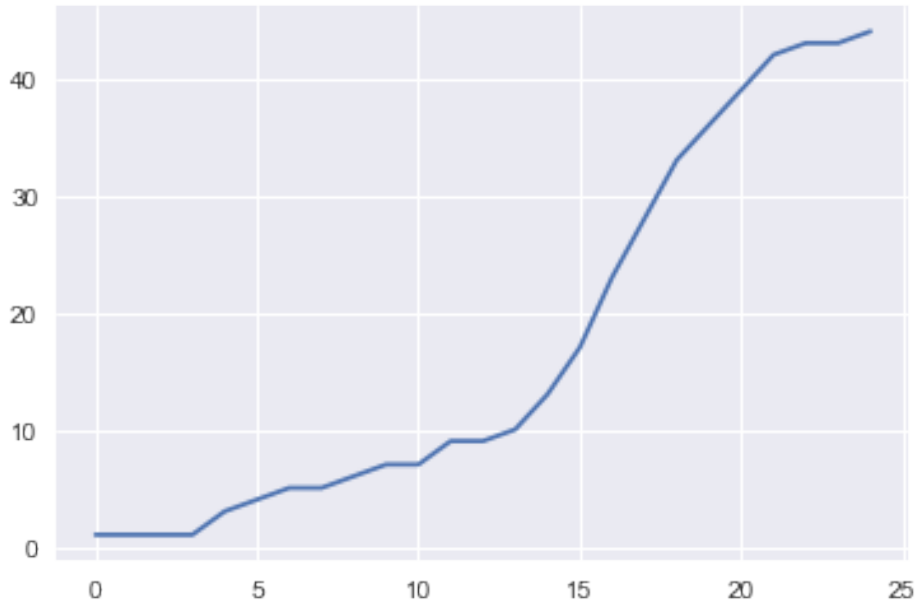
```
sns.kdeplot(x)
plt.show()
```



Wizualizacja danych

- Narysujmy dystrybuantę empiryczną dla `df.Weight.values`:

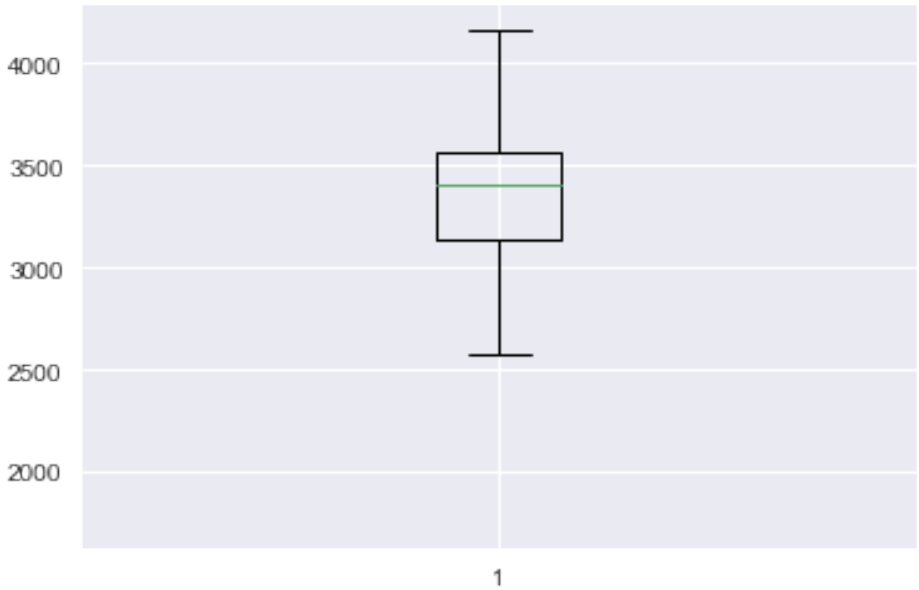
```
plt.plot(stats.cumfreq(x,numbins=25)[0])
plt.show()
```



Wizualizacja danych

- Narysujmy wykres pudełkowy (boxplot) dla `df.Weight.values`:

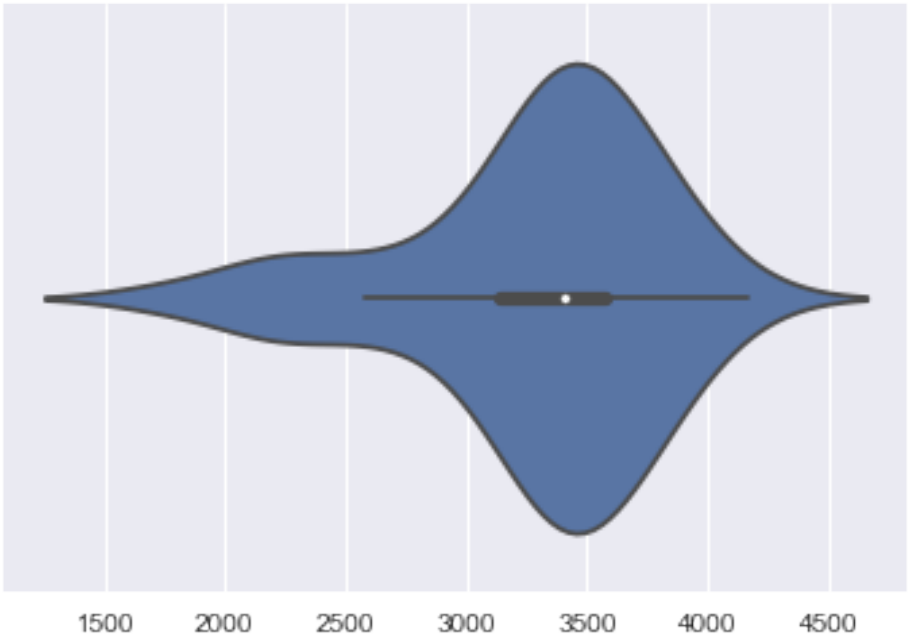
```
plt.boxplot(x, sym='*')
plt.show()
```



Wizualizacja danych

- Narysujmy wykres skrzypcowy (violin plot) dla `df.Weight.values`:

```
sns.violinplot(x)
plt.show()
```



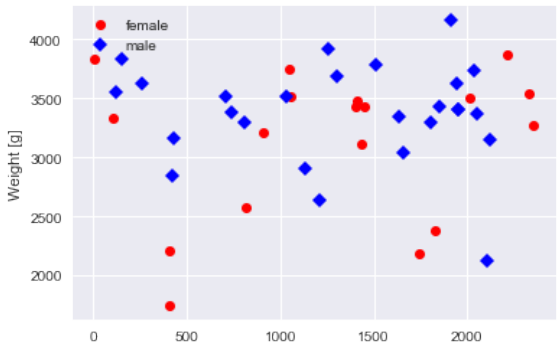
Wizualizacja danych

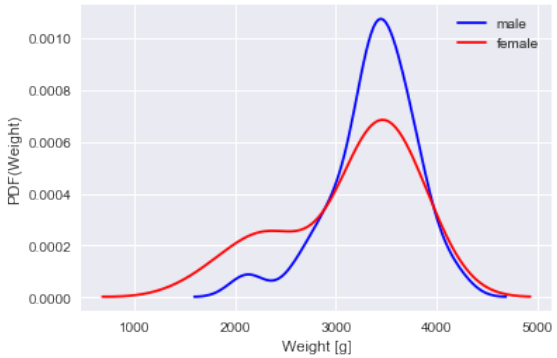
Zadanie

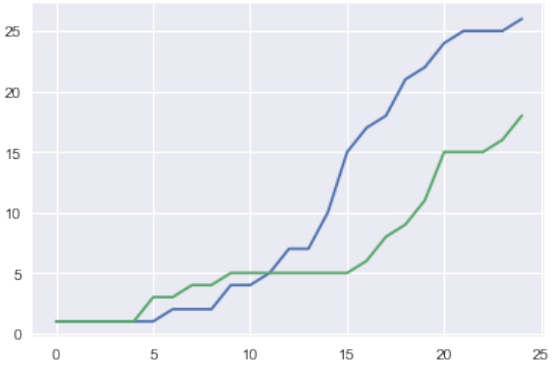
Proszę podzielić próbkę na dwie grupy ze względu na płeć (kolumna “sex”) i narysować dla nich:

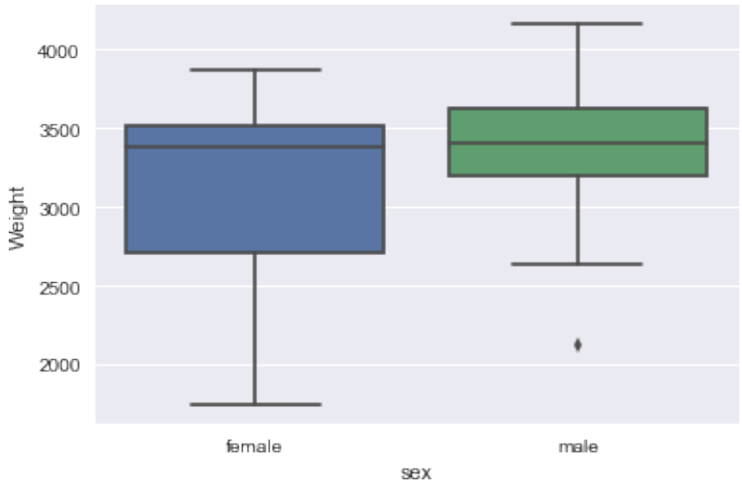
- wykres w kształcie szeregu czasowego,
- histogramy,
- estymacje gęstości,
- dystrybuanty empiryczne,
- wykresy pudełkowe,
- wykresy skrzypcowe.

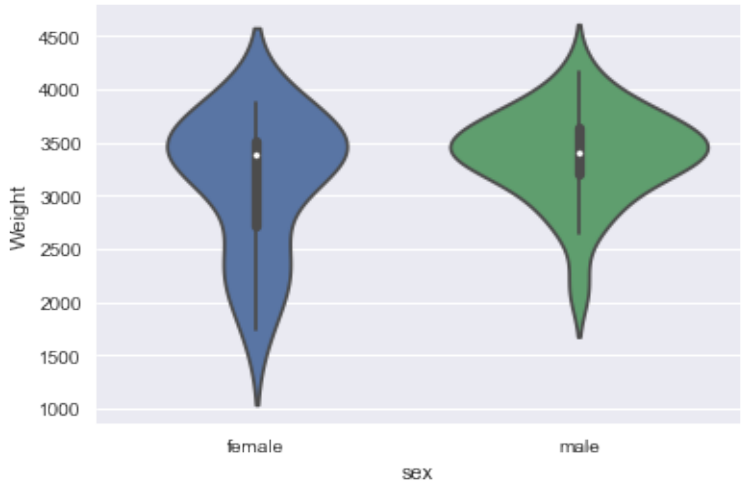
Wizualizacja danych



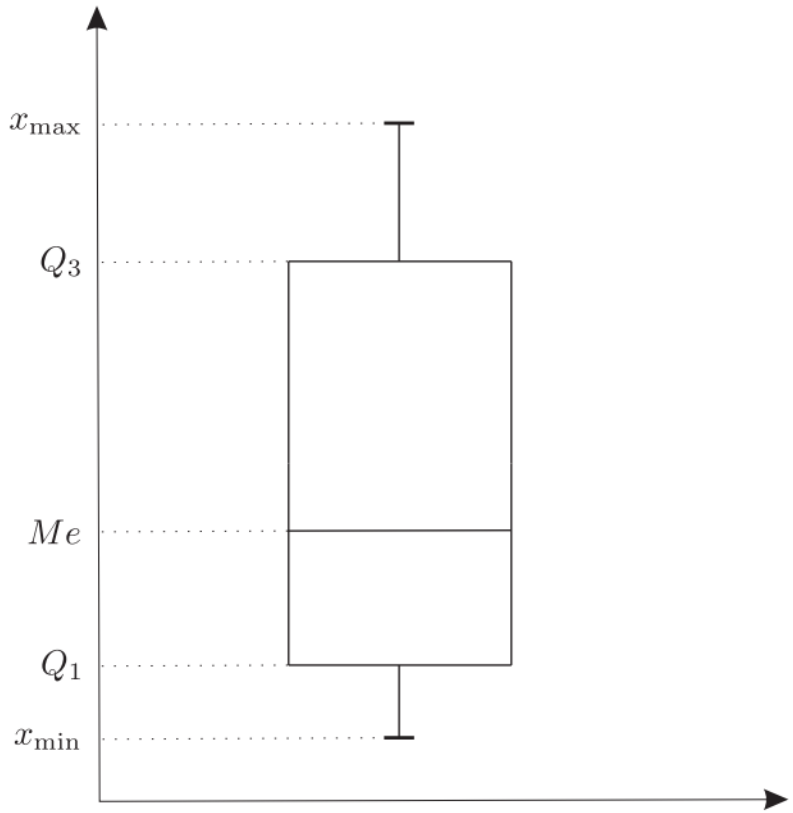




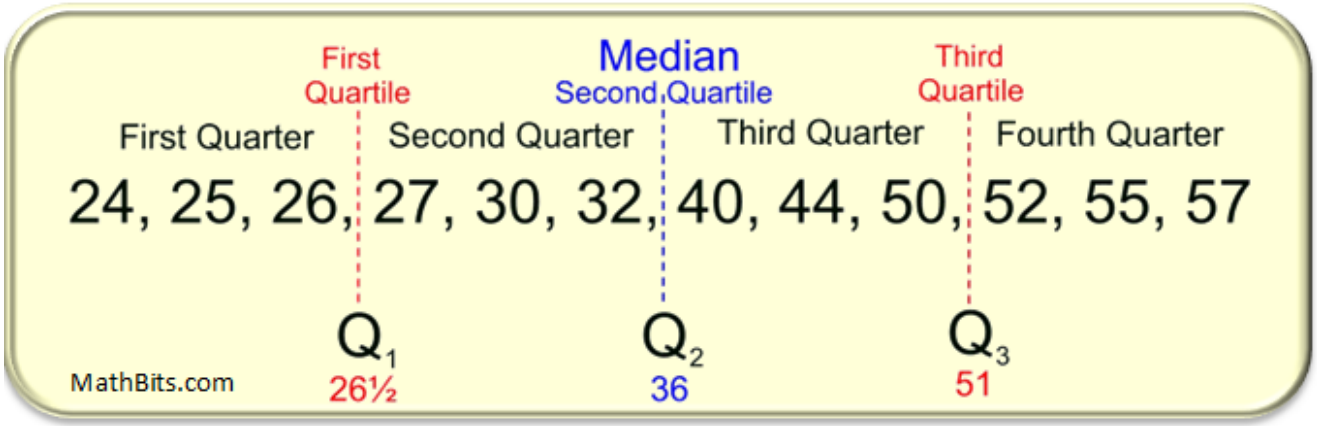


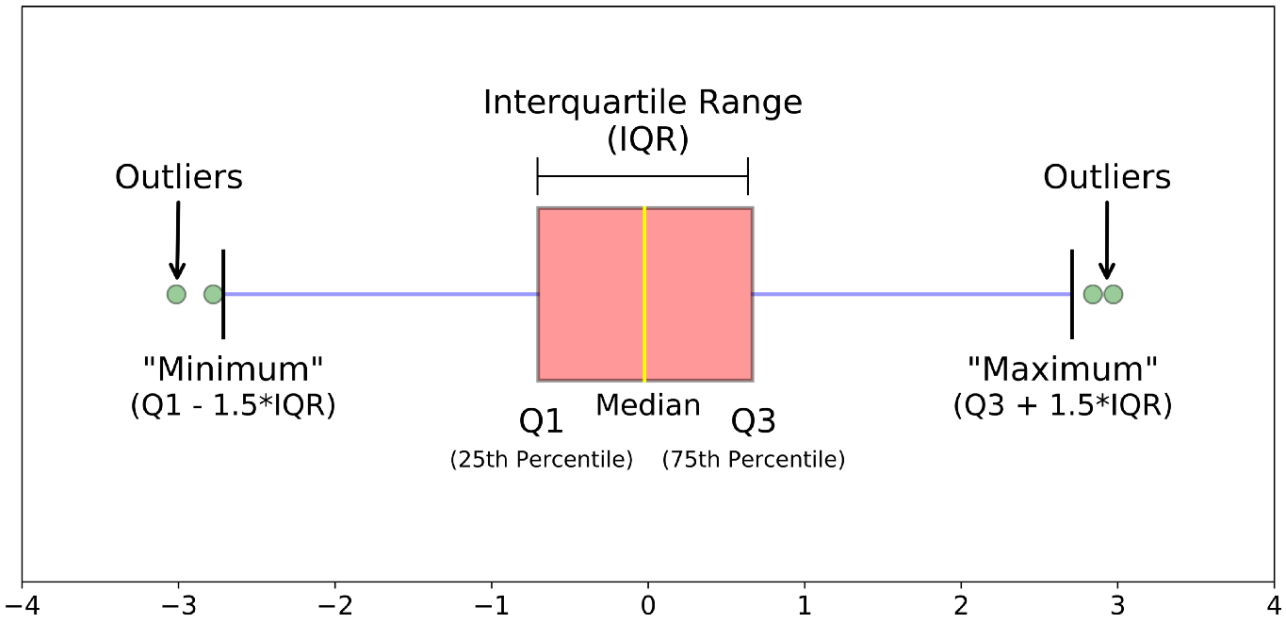


Wykres pudełkowy



Wykres pudełkowy





Wykres pudełkowy

Wykres pudełkowy inaczej ramkowy (ang. boxplot, box-and-whisker plot). Tworzymy go odkładając na pionowej osi wartości niektórych parametrów rozkładu:

- Nad osią umieszczony jest prostokąt (pudełko), którego
  - dolny bok jest wyznaczony przez pierwszy kwartyl,
  - górny bok zaś przez trzeci kwartyl.
- Wysokość pudełka odpowiada wartości rozstępu ćwiartkowego.
- Wewnątrz prostokąta znajduje się pozioma linia, określająca wartość mediany.

Wykres pudełkowy

- Rysunek pudełka uzupełniamy od góry i od dołu odcinkami (wąsy):
  - dolny koniec dolnego odcinka wyznacza najmniejszą wartość,
  - górny koniec górnego odcinka, to wartość największa.
- Końcowe wartości wąsów muszą spełniać dodatkowy warunek:
  - dolny koniec nie może być mniejszy niż

$$Q1 - 1,5\Delta(Q3 - Q1)$$

  - górny koniec nie może być większy niż

$$Q3 + 1,5\Delta(Q3 - Q1)$$

- Jeśli występują obserwacje spoza tego przedziału, to nanoszone są na wykres indywidualnie (są to tzw. obserwacje odstające (ang. outlier)).

Wykres skrzypcowy (ang. violin plot)

- Można go traktować jako wygładzoną wersję wykresu pudełkowego.
- Przydatny jest zwłaszcza w przypadku danych wielomodalnych.
- Jest to w zasadzie wykres pudełkowy, gdzie szerokość skrzypiec w punkcie x odpowiada natężeniu obserwacji o wartości cechy zbliżonej do x (estymator jądrowy gęstości).

Estymator jądrowy gęstości

Definicja (Estymator jądrowy gęstości)

Estymator jądrowy gęstości lub jądrowy estymator gęstości to rodzaj estymatora nieparametrycznego, przeznaczony do wyznaczania gęstości **rozkładu zmiennej losowej** (my na razie myślimy o rozkładzie punktów), na podstawie uzyskanej próby, czyli wartości jakie badana zmienna przyjęła w trakcie dotychczasowych pomiarów.

Estymator jądrowy gęstości

Niech dany będzie d-wymiarowy zbiór danych X. Jej estymator jądrowy

$$\hat{f} : \mathbb{R}^d \to [0, \infty)$$

wyznacza się w oparciu o wartości n-elementowej próby losowej $x_1, x_2, ..., x_n$ w swej podstawowej formie jest on definiowany wzorem:

$$\hat{f}(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) ,$$

gdzie mierzalna, symetryczna względem zera oraz posiadająca w tym punkcie słabe maksimum globalne funkcja

$$K : \mathbb{R}^n \to [0, \infty)$$

spełnia warunek $\int_{\mathbb{R}^d} K(x)dx = 1$ i nazywana jest jądrem, natomiast dodatni współczynnik h określa się mianem parametru wygładzania.

Estymator jądrowy gęstości

$$\hat{f}(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) ,$$

Najefektywniejszym w sensie kryterium błędu średniokwadratowego jest tak zwane jądro Epanecznikowa

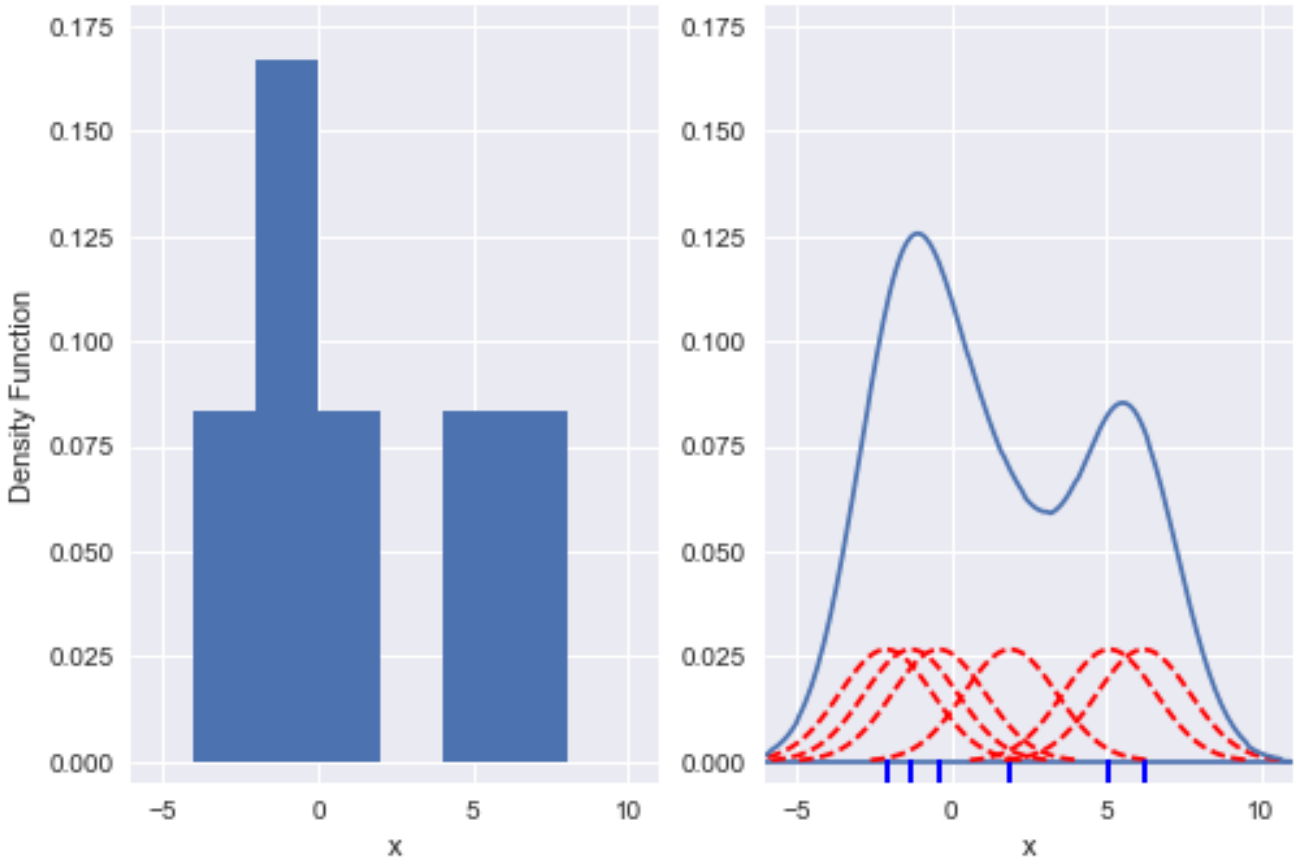
$$K(x) = \begin{cases} \frac{3}{4}(1 - x^2) & \text{dla } x \in [-1, 1] \\ 0 & \text{dla } x \in (-\infty, -1) \cup (1, \infty) \end{cases} .$$

Można też używać gęstości rozkładu normalnego.

https://pl.wikipedia.org/wiki/Estymator_j%C4%85drowy_g%C4%99sto%C5%9Bci

Estymator jądrowy gęstości

https://github.com/przem85/bootcamp/blob/master/statistics/D01_Z11.ipynb



Próbka

Definicja

Prostą próbą losową (lub krócej próbą losową) o liczności n nazywamy **ciąg niezależnych zmiennych losowych** $X_1, X_2, \ldots, X_n$ określonych na przestrzeni zdarzeń elementarnych Ω i takich, że każda ze zmiennych ma taki sam rozkład.

Próbka

Definicja

Prostą próbą losową (lub krócej próbą losową) o liczności n nazywamy **ciąg niezależnych zmiennych losowych** $X_1, X_2, \ldots, X_n$ określonych na przestrzeni zdarzeń elementarnych Ω i takich, że każda ze zmiennych ma taki sam rozkład.

Uwaga

Konkretny ciąg wartości $x_1, x_2, ..., x_n$ (prostej) próby losowej $X_1, X_2, \ldots, X_n$ nazywamy realizacją (prostej) próby losowej lub próbką.

Próbka

Definicja

Prostą próbą losową (lub krócej próbą losową) o liczności n nazywamy **ciąg niezależnych zmiennych losowych** $X_1, X_2, \ldots, X_n$ określonych na przestrzeni zdarzeń elementarnych Ω i takich, że każda ze zmiennych ma taki sam rozkład.

Uwaga

Konkretny ciąg wartości $x_1, x_2, ..., x_n$ (prostej) próby losowej $X_1, X_2, \ldots, X_n$ nazywamy realizacją (prostej) próby losowej lub próbką.

Uwaga

Statystyką nazywamy każdą zmienną losową będącą ustaloną funkcją próby losowej $X_1, X_2, \ldots, X_n$.

O co chodzi z tą zmienną losową?