Matematyczne aspekty analizy danych
semestr zimowy 2025/2026

Dr Anna Muranova
UWM w Olsztynie

Wykład 5

Rozdział 4. Statystyka opisowa

Statystyka opisowa

Statystyka opisowa – dział statystyki zajmujący się metodami opisu danych statystycznych uzyskanych podczas badania statystycznego. Celem stosowania metod statystyki opisowej jest podsumowanie zbioru danych i wyciągnięcie pewnych podstawowych wniosków i uogólnień na temat zbioru.

Statystykę opisową stosuje się zazwyczaj jako pierwszy i podstawowy krok w analizie zebranych danych.

Zbiór danych oraz tabela

Zbiór danych – kolekcja danych statystycznych zwykle ujętych w formie tablicy. Najczęściej kolumny odpowiadają obserwowanym cechom statystycznym, a każdy wiersz opisuje jedną obserwację z próby. Wartości komórek macierzy natomiast opisują realizacje danych zmiennych w kolejnych obserwacjach.

Imię	Wiek	Wzrost	Kolor oczu
Adam	26	167	Brązowe
Sylwia	34	164	Piwne
Tomasz	42	183	Niebieskie

W Pythonie zaczniemy obliczenia statystyczne od Biblioteki `Numpy` do obliczeń.

```
import numpy as np

imie = np.array(['Adam', 'Sylwia', 'Tomasz'])
wiek = np.array([26, 34, 42])
wzrost = np.array([167, 164, 183])
kolor_oczy = np.array(['Brazowe', 'Piwne', 'Niebeskie'])
```

Wyznaczanie miar rozkładu

Do opisu służą **miary rozkładu** – różnego rodzaju wielkości obliczane na podstawie uzyskanych danych. Interpretacja wartości tych miar dostarcza informacji na temat charakteru rozkładu cechy.

Miary można podzielić na kilka podstawowych kategorii:

- miary położenia, np. miary tendencji centralnej (np. średnia arytmetyczna, średnia geometryczna, średnia harmoniczna, średnia kwadratowa, mediana, moda), kwantyl
- miary zróżnicowania, np. odchylenie standardowe, wariancja, rozstęp, rozstęp ćwiartkowy, średnie odchylenie bezwzględne, odchylenie ćwiartkowe, współczynnik zmienności
- miary asymetrii, np. współczynnik skośności, współczynnik asymetrii, trzeci moment centralny
- miary koncentracji, np. współczynnik Giniego, kurtoza

Miary położenia klasyczne

Niech $X_1, X_2, \ldots, X_n \in \mathbb{R}$ są wartościami danej cechy ze zbioru danych.

- *Średnia arytmetyczna* jest zdefiniowana jako

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i.$$

- *Średnia ważona* jest zdefiniowana jako

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i w_i,$$

gdzie w_i są znane z góry i sumują się do 1 (inaczej podzielić przez $\sum w_i$)

- Jeżeli $X_i \geq 0$ dla każdego i, to *średnia geometryczna* jest zdefiniowana jako

$$G(X_1, \ldots, X_n) = \sqrt[n]{\prod_{i=1}^{n} X_i}.$$

- Jeżeli $X_i > 0$ dla każdego i, to *średnia harmoniczna* jest zdefiniowana jako

$$H(X_1, X_2, \ldots, X_n) = \frac{n}{\sum_{i=1}^{n} \frac{1}{X_i}}.$$

Miary położenia klasyczne: numpy, scipy

```
import numpy as np
#Biblioteka scipy
from scipy.stats import gmean
from scipy.stats import hmean
from scipy.stats import pmean

dane = np.array([3,9,27])

print(sum(dane)/len(dane))#13.0
print(np.mean(dane))#13.0
w = np.array([0.2,0.4,0.4])
print(dane*w)#[ 0.6  3.6 10.8]
print(np.sum(dane*w))#15.0
print(pmean(dane, p=1, weights=w))#15.0
print(pow(dane.prod(), 1 / len(dane)))#8.999999999999998
print(gmean(dane))#9.000000000000002
print(pmean(dane, p=0))#9.000000000000002
print(len(dane) / sum(1 / dane))#6.230769230769231
print(hmean(dane))#6.230769230769231
```

pmean: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.pmean.html

Miary położenia pozycyjne: moda

Dominanta (*wartość modalna*, *moda*) – wartość najczęściej występująca w probie. Należy pamiętać, że dominantą może być więcej niż jedna wartość.

```
import numpy as np
dane1 = np.array([1, 2, 3, 4, 4, 4, 4, 4, 5, 6, 7, 7, 7, 8])
dane2 = np.array([1, 2, 3, 4, 4, 4, 5, 6, 7, 7, 7, 8])
dane3 = np.array(["red", "blue", "black", "blue", "black", "black",
"brown"])
def my_mode(data):
    x = np.array([])
    res = np.array([])
    for i in np.unique(data):
        x = np.append(x, np.count_nonzero(data == i))
    m = max(x)
    for i in np.unique(data):
        if np.count_nonzero(data == i)==m:
            res = np.append(res, i)
    return res

print(my_mode(dane1))#[4.]
print(my_mode(dane2))#[4. 7.]
print(my_mode(dane3))#['black']
```

Miary położenia pozycyjne: moda

statistics oraz scipy

```
import numpy as np
from scipy.stats import mode
import statistics as st

dane1 = np.array([1, 2, 3, 4, 4, 4, 4, 4, 5, 6, 7, 7, 7, 8])
dane2 = np.array([1, 2, 3, 4, 4, 4, 5, 6, 7, 7, 7, 8])
dane3 = np.array(["red", "blue", "black", "blue", "black", "black",
                  "brown"])


print(mode(dane1))#ModeResult(mode=np.int64(4), count=np.int64(5))
print(mode(dane2))#tylko jedna ModeResult(mode=np.int64(4),
                                          count=np.int64(3))
#print(mode(dane3))#blad

print(st.mode(dane1))#4
print(st.mode(dane2))#tylko jedna 4
print(st.mode(dane3))#black
```

Miary położenia pozycyjne: kwantyle

Niech $X_1, X_2, \ldots, X_n$ są wartościami danej cechy ze zbioru danych, które da się uporządkować, i.e. możemy założyć, ze

$$X_1 \leq X_2 \leq \cdots \leq X_n$$

Kwantylem rzędu p, gdzie $0 \leq p \leq 1$, w rozkładzie empirycznym P_X próby X nazywamy każda wartość X_j, dla którego spełnione są nierówności

$$\#\{X \mid X \leq X_j\} \geq pn \text{ oraz } \#\{X \mid X \geq X_j\} \geq (1-p)n.$$

- Kwantyl rzędu $1/2$ to inaczej *mediana*.
- Kwantyle rzędu $1/4$, $2/4$, $3/4$ są inaczej nazywane *kwartylami (pierwszy, drugi, trzeci)* i oznaczany Q_1, Q_2, Q_3.
- Kwantyle rzędu $1/5$, $2/5$, $3/5$, $4/5$ to inaczej *kwintyle*.
- Kwantyle rzędu $1/10, 2/10, \ldots, 9/10$ to inaczej *decyle*.
- Kwantyle rzędu $1/100, 2/100, \ldots, 99/100$ to inaczej *percentyle*.

Rozstęp ćwiartkowy (*rozstęp międzykwartylowy*, *przedział międzykwartylowy*, *rozstęp kwartylny*, IQR (od ang. interquartile range)) – różnica między trzecim a pierwszym kwartylem.

Definicja mediany ściślej od zagadnień, przy jej obliczaniu z próbki o parzystej liczbie elementów często stosuje się średnią arytmetyczną dwóch środkowych elementów.

Miary położenia pozycyjne: kwantyle

```
import numpy as np

# create an array
dane = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])

# calculate the 0.25th, 0.50th and 0.75th quantile of the array
q25 = np.quantile(dane, 0.25)
q50 = np.quantile(dane, 0.50)
q75 = np.quantile(dane, 0.75)

print(q25, q50, q75)
#3.0 6.0 9.0

dane = np.array([0, 1, 2, 3, 4, 5, 6, 7])

q25 = np.quantile(dane, 0.25)
q50 = np.quantile(dane, 0.50)
q75 = np.quantile(dane, 0.75)

print(q25, q50, q75)
#1.75 3.5 5.25
```

Tendencja centralna

Tendencja centralna – liczba (albo wartość), używana do opisania zbioru danych za pomocą jednej liczby (albo jednej albo wartości).

Miarami tendencje centralnej mogą występować np.: mediana, moda, rożne średnie.

Wzór Pearsona

Wzór Pearsona na relacje pomiędzy modą (Mo), medianą (Me) oraz średnią dla wartości liczbowych oraz dla rozkładów symetrycznych i umiarkowanie asymetrycznych:

$$\overline{X} - Mo = 3(\overline{X} - Me).$$

Miary zmienności klasyczne

Niech $X_1, X_2, \ldots, X_n \in \mathbb{R}$ są wartościami danej cechy z **populacji** danych. Wtedy dla (X_i) są zdefiniowane następne miary zmienności klasyczne:

- *Odchylenie przeciętne:*

$$d = \frac{1}{n} \sum_{i=1}^{n} |X_i - \overline{X}|,$$

- *Wariancja:*

$$\sigma^2 = S^2(X) := \frac{1}{n} \sum_{i=1}^{n} (X_i - \overline{X})^2,$$

- *Odchylenie standardowe:*

$$\sigma = S(X) := \sqrt{\frac{1}{n} \sum_{i=1}^{n} (X_i - \overline{X})^2}.$$

Miary zmienności klasyczne w Pythonie

```
import numpy as np
import math

# create an array
dane = np.array([0, 1, 5, 7, 9, 10, 14])
n = len(dane)
print(sum(abs(dane-np.mean(dane)))/n)#odchylenie precietne
print(sum((dane-np.mean(dane))**2)/n)#wariancja
print(np.var(dane))#wariancja
print(math.sqrt(sum((dane-np.mean(dane))**2)/n))#odchylenie
     #standartowe
print(np.std(dane))#odchylenie standartowe
```

```
3.918367346938776
21.387755102040813
21.387755102040813
4.624689730353898
4.624689730353898
```

Wartości z próby

Niech $X_1, X_2, \ldots, X_n \in \mathbb{R}$ są wartościami danej cechy z **próby** danych. Wtedy dla (X_i) są zdefiniowane następne miary zmienności klasyczne:

- *Wariancja:*

$$\sigma^2 = S^2(X) := \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \overline{X})^2,$$

- *Odchylenie standardowe:*

$$\sigma = S(X) := \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (X_i - \overline{X})^2}.$$

Wartości z próby w Pythonie

```
import numpy as np
import statistics as stat
import math

# create an array
dane = np.array([0, 1, 5, 7, 9, 10, 14])
n = len(dane)

print(sum((dane-np.mean(dane))**2)/(n-1))#wariancja
print(stat.variance(dane))#wariancja: na int zwraca int
print(math.sqrt(sum((dane-np.mean(dane))**2)/(n-1)))#odchylenie
     #standartowe
print(stat.stdev(dane.tolist()))#odchylenie standartowe
#dziala na listach
```

```
24.95238095238095
24
4.99523582550223
4.995235825502231
```

Dlaczego dzielimy przez $n - 1$

Czy zauważyłeś różnicę?

Kiedy uśredniamy różnice podniesione do kwadratu w probie, dzielimy przez $n - 1$ zamiast przez pełną liczbę elementów n. Dlaczego? Ma to na celu ograniczenie ewentualnego ciążenia próby i uniknięcie sytuacji, w której nie docenilibyśmy wariancji populacji. Zmniejsza liczbę elementów w mianowniku o 1, zwiększamy wariancję, co pozwala uchwycić większą pewność próby.

Miary zmienności pozycyjne

Niech $X_1, X_2, \ldots, X_n \in \mathbb{R}$ są wartościami danej cechy ze zbioru danych.

- *Rozstęp* jest różnicą między największą i najmniejszą wartością cechy. Oblicza się go ze wzoru:

$$R = X_{\max} - X_{\min},$$

gdzie $X_{\max} = \max_i X_i$, $X_{\min} = \min_i X_i$.

- *Rozstęp ćwiartkowy* (*rozstęp międzykwartylowy*, *przedział międzykwartylowy*, *rozstęp kwartylny*, IQR – od ang. interquartile range) jest różnicą między trzecim a pierwszym kwartylem

$$R_Q = Q_3 - Q_1.$$

- *Odchylenie ćwiartkowe* (*odchylenie kwartylne*) określone jest wzorem:

$$Q = \frac{R}{2} = \frac{Q_3 - Q_1}{2}.$$

- *Typowy obszar zmienności klasyczny* jest zdefiniowany jako $X_{typ} := [Me - Q, Me + Q]$.

Lemma

Pomiędzy odchyleniami: ćwiartkowym, przeciętnym i standardowym, obliczonymi z tego samego szeregu, zachodzi następująca relacja:

$$Q < d < S.$$

Podział cech statystycznych z dopuszczalnymi operacjami.

Skala	Własności miary	Operacje matematyczne	Operacje zaawansowane	Przykłady
nominalne (ang. nominal)	Klasyfikacja, członkostwo	$=, \neq$	Grupowanie	kolor oczy, płeć
porządkowe (ang. ordinal)	Porównanie, poziom	$=, \neq, >, <$	Sortowanie	wzrost, wiek
przedziałowe/interwałowe (ang. interval)	Różnica, bliskość	$=, \neq, >, <, +, -$	Porównanie ze standardem	data, temp. w ^{o}C
proporcjonalne (ang. ratio)	Wielkość, ilość	$=, \neq, >, <, +, -, *, /$	Iloraz	czas, masa

Podział cech statystycznych z miarami statystycznymi.

Skala	Własności miary	Tendencja centralna	Zmienność
nominalne (ang. nominal)	Klasyfikacja, członkostwo	Moda	Zróżnicowanie jakościowe*
porządkowe (ang. ordinal)	Porównanie, poziom	Mediana	Rozstęp, rozstęp ćwiartkowy**
interwałowe (ang. interval)	Różnica, bliskość	Średnia arytmetyczna	Odchylenie przeciętne
proporcjonalne (ang. ratio)	Wielkość, ilość	Średnia geometryczna oraz średnia harmoniczna	Wariancja

*Indeks zmienności jakościowej (IQV) jest miarą rozproszenia statystycznego w rozkładach nominalnych. Istnieje wiele takich indeksów, ale są one stosunkowo słabo zbadane w literaturze statystycznej. Najprostszy jest współczynnik zmienności, podczas gdy bardziej złożone wskaźniki obejmują entropię informacyjną.

Zmienne nominalne mogą być zapisane w postaci tablicy:

Kategoria	Ilość występowań	Częstotliwość

...

Wiele z IQV zostało podsumowanych i opracowanych przez Wilcoxa, który wymaga spełnienia następujących właściwości normalizacyjnych:

- Odchylenie waha się od 0 do 1.
- Zmienność wynosi 0 wtedy i tylko wtedy, gdy wszystkie wartości próbek należą do jednej kategorii.
- Odchylenie wynosi 1 wtedy i tylko wtedy, gdy wartości próbek są równomiernie podzielone na wszystkie kategorie.

W szczególności wartość tych standaryzowanych wskaźników nie zależy od liczby kategorii ani liczby próbek.

Niech $X_1, X_2, \ldots, X_n \in \mathbb{R}$ są wartościami danej cechy ze zbioru danych. *Indeks Wilcox'a* (*odchylenie przeciętne od mody*, DM) jest zdefiniowany jako

$$M = K\#\{X = Mo\} - n,$$

gzie K jest ilością kategorii (ilość różnych wartości X_i).
Współczynnik zmienności (*Indeks Freemana*):

$$\nu = 1 - \frac{\#\{X = Mo\}}{n},$$

tzn. $M = nK(1-\nu) - n$. *Wariancja od mody*:

$$\mathrm{ModVR} = \frac{K\nu}{K-1} = \frac{K(1 - \#\{X = Mo\}/n)}{K-1}.$$

Podział cech statystycznych z miarami statystycznymi.

Skala	Własności miary	Tendencja centralna	Zmienność
nominalne (ang. nominal)	Klasyfikacja, członkostwo	Moda	Zróżnicowanie jakościowe*
porządkowe (ang. ordinal)	Porównanie, poziom	Mediana	Rozstęp, rozstęp ćwiartkowy**
interwałowe (ang. interval)	Różnica, bliskość	Średnia arytmetyczna	Odchylenie przeciętne
proporcjonalne (ang. ratio)	Wielkość, ilość	Średnia geometryczna oraz średnia harmoniczna	Wariancja

**Każde dane porządkowe mogą być zapisany w postaci liczb z zachowaniem porządku. Rozstęp ćwiartkowy wskazuje na konsensus albo polaryzację wyników.

Example (Skala Likerta)

Niech była uzupełniona ankieta:

zdecydowanie nie zgadzam się	raczej się nie zgadzam	nie mam zdania	raczej się zgadzam	zdecydowanie się zgadzam
1	2	3	4	5

i wyniki (uporządkowane) są:

$$[1,1,2,2,2,2,2,2,2,2,2,3,3,3,3][3,3,3,3,3,3,3,3,3,3,3,3,3,3,3]$$

$$[3,3,3,3,3,4,4,4,4,4,4,4,4,4,4][4,4,4,4,4,4,4,4,5,5,5,5,5,5,5],$$

tzn. $Q_1 = 3$, $Q_2 = 3$, $Q_3 = 4$, oraz $R_Q = Q_3 - Q_1 = 1$. To jest stosunkowo niewielki rozstęp ćwiartkowy, który wskazuje na konsensus. Z kolei większe rozstęp ćwiartkowy może sugerować, że opinie są spolaryzowane, tj. że respondenci mają tendencję do posiadania zdecydowanych opinii za lub przeciw temu tematowi.

Biblioteka Numpy

NumPy – otwarto-źródłowa biblioteka programistyczna dla języka Python dla obliczeń numerycznych, dodająca obsługę dużych, wielowymiarowych tabel i macierzy

Tablicy

Tablica (ang. array) w `NumPy` to struktura o jednym wymiarze lub większej liczbie wymiarów pozwalająca na działania ze zbiorami danych numerycznych, zaczynając od kilkuelementowych po ogromne zbiory przechowywane np. w chmurze. Pamiętajmy, że nie tylko wymiary charakteryzują tablicę – istnieje też kilka innych, równie ważnych cech.

- Tablice są stałej wielkości – nie możemy zmienić rozmiaru po jej utworzeniu.
- Przechowują elementy tego samego typu, np. liczby całkowite lub zmiennoprzecinkowe.
- Wykorzystywane „pod spodem" algorytmy pozwalają na bardzo szybkie operacje i efektywne wykorzystanie pamięci.

Niektórzy porównują tablice do list w Pythonie, ale listy różnią się od tablic, np. możliwością przechowywania elementów różnego typu (listy mogą być heterogeniczne), a także są jednowymiarowe (chociaż możliwe jest przechowywanie jednowymiarowej listy w drugiej liście).

Konwersja listy w tablicę

```
import numpy as np

# jednowymiarowa tablica na podstawie listy
arr_1d = np.array([1, 2, 3])
print(arr_1d)

# dwuwymiarowa tablica na podstawie listy
arr_2d = np.array([[1, 2, 3], [4, 5, 6]])
print(arr_2d)
```

```
[1 2 3]

[[1 2 3]
[4 5 6]]
```

Tworzenie tablic

```
import numpy as np

# jednowymiarowa tablica wypełniona zerami o długości 3
arr_zeros = np.zeros(5)
print(arr_zeros)#[0. 0. 0. 0. 0.]

# tablica z 10 równo rozłożonymi wartościami w zakresie od 0 do 2
a = np.linspace(0, 2, 10)
print(a)
#[0.         0.22222222 0.44444444 0.66666667 0.88888889 1.11111111
#  1.33333333 1.55555556 1.77777778 2.        ]

# dwuwymiarowa tablica wypełniona jedynkami o wymiarach 3x3
arr_2d_ones = np.ones((3, 3))
print(arr_2d_ones)
#[[1. 1. 1.]
# [1. 1. 1.]
# [1. 1. 1.]]
```

Tablica z wartości losowych.

```
import numpy as np

# jednowymiarowa tablica z losowymi
#wartościami z przedziału [0,1] o długości 3
arr = np.random.rand(3)
print(arr)
#[0.84218351 0.24822803 0.46510147]

# dwuwymiarowa tablica z losowymi
#wartościami z rozkładu normalnego o wymiarach 3x3
arr_2d = np.random.randn(3, 3)
print(arr_2d)
#[[-2.4500044  -0.37286908 -0.49917813]
# [-2.89081799  0.03262008 -0.73594147]
# [ 0.08182802 -0.64018268 -1.2193713 ]]
```

Tworzenie tablic: resize vs. reshape

```
import numpy as np

# tworzenie tablicy z 10 elementami
a = np.array([3, 7, 3, 3, 2, 9, 7, 1, 5, 4])
print(np.shape(a))#(10,)
b = a.reshape(2,5)
print(a) # [3 7 3 3 2 9 7 1 5 4]
print(b)
# [[3 7 3 3 2]
# [9 7 1 5 4]]
b = a.resize(2,5)
print(a)
#[[3 7 3 3 2]
# [9 7 1 5 4]]
print(b) # None
```

Podstawowe operacje

```
import numpy as np

# tworzenie jednowymiarych tablic a i b
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])

# dodawanie tablicy b do tablicy a
c = a + b
print(c)#[5 7 9]

# dodawanie 5 do tablicy a
print(5+a)#[6 7 8]

# mnożenie tablicy a przez stałą 2
d = 2 * a
print(d)#[2 4 6]
```

Operacje logiczne

Wynikiem operacji logicznych takich jak AND (koniunkcja), OR (alternatywa) czy NOT (negacja) jest tablica zawierająca wartości logiczne True i False.

```
import numpy as np

# tworzenie jednowymiarych tablic a i b
a = np.array([1, 2, 3])
b = np.array([3, 2, 1])
# sprawdzamy, które elementy w tablicy a są mniejsze od
# odpowiadających im elementów w tablicy b
c = a < b
print(c)#[True False False]

# sprawdzamy które elementy w tablicy a są równe 2 lub 3
d = (a == 2) | (a == 3)
print(d)# [False  True  True]
```

Operacje redukcyjne

Wynikiem operacji redukcyjnych takich jak suma, minimum, maksimum czy średnia jest skalar.

```
import numpy as np

# tworzenie jednowymiarowej tablicy a
a = np.array([1, 2, 3])

# suma elementów tablicy a
b = np.sum(a)
print(b) #6

# średnia arytmetyczna dla elementów tablicy a
c = np.mean(a)
print(c) #2.0

# największy element w tablicy a
d = np.max(a)
print(d) #3
```

Operacje algebraiczne

```
import numpy as np

# tworzenie macierzy 3x3
a = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
# transpozycja macierzy a do macierzy b
b = np.transpose(a)
print(a)
#[[1 2 3]
# [4 5 6]
# [7 8 9]]

print(b)
# [[1 4 7]
# [2 5 8]
# [3 6 9]]
```

Operacje algebraiczne

Dobrym przykładem jest także mnożenie macierzy i obliczenie wyznacznika.

```
import numpy as np

# tworzenie macierzy a i b
a = np.array([[1, 2], [3, 4]])
b = np.array([[5, 6], [7, 8]])

# operacja mnożenia macierzy a i b
c = np.dot(a, b)
print(c)
#[[19 22]
# [43 50]]

# porownaj
print(a*b)
#[[ 5 12]
# [21 32]]

# obliczenie wyznacznika macierzy a
d = np.linalg.det(a)
print(d)#-2.0000000000000004
```

Operacje algebraiczne

A także rozwiązanie układu równań liniowych.

```
import numpy as np

# wyznaczenie rozwiązania układu równań liniowych Ax = b
A = np.array([[1, 2], [3, 4]])
b = np.array([5, 6])
x = np.linalg.solve(A, b)
print(x)#[-4.   4.5]
print(np.dot(np.linalg.inv(A),b)) #[-4.   4.5]
```

Inne funkcje

Sortowanie tablicy

```
import numpy as np

# tworzenie tablicy z 10 elementami
a = np.array([3, 7, 3, 3, 2, 9, 7, 1, 5, 4])

# sortowanie tablicy
b = np.sort(a)
print(b)
#[1 2 3 3 3 4 5 7 7 9]
```

Indeksacja i krojenie

Jak w listach:

```
import numpy as np

data = np.array([1, 2, 3])

print(data[1]) #2
print(data[0:2]) #[1 2]
print(data[1:]) #[2 3]
print(data[-2:]) #[2 3]


b = np.array([[1., 2., 3], [3., 4., 5]])
print(b[1,2]) #5.0
print(b[:,2])  #[3. 5.]
```

Indeksacja i krojenie

Logiczne:

```
import numpy as np

a = np.array([1,2,3])
print(a[[True, True, False]])
#[1 2]

b = np.array([[1 , 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
print(b[b<5])
#[1 2 3 4]

divisible_by_2 = b[b%2==0]
print(divisible_by_2)#[ 2  4  6  8 10 12]

c = b[(b > 2) & (b < 11)]
print(b) #[[ 1  2  3  4]
# [ 5  6  7  8]
# [ 9 10 11 12]]
```

https://numpy.org/doc/stable/user/absolute_beginners.html#indexing-and-slicing

Zamiana wartości 1

```
import numpy as np

a = np.array([[1 , 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
a[2,1] = 0
print(a)
#[[ 1  2  3  4]
#[ 5  6  7  8]
#[ 9  0 11 12]]
c = a
a[2,:] = 0
print(a)
#[[1 2 3 4]
# [5 6 7 8]
# [0 0 0 0]]
print(c)
#[[1 2 3 4]
# [5 6 7 8]
# [0 0 0 0]]
```

https://numpy.org/doc/stable/user/absolute_beginners.html#indexing-and-slicing

Zamiana wartości 2

```
import numpy as np

b = np.array([[1 , 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
d = b.copy()
b[b>5] = 0
print(b)
#[[1 2 3 4]
# [5 0 0 0]
# [0 0 0 0]]

print(d)
#[[ 1  2  3  4]
# [ 5  6  7  8]
# [ 9 10 11 12]]


https://numpy.org/doc/stable/user/absolute_beginners.html#
indexing-and-slicing
```

Operacje matematyczne

https://numpy.org/doc/stable/reference/routines.math.html

Rozkład normalny

W poprzednim rozdziale mówiliśmy już o rozkładach prawdopodobieństwa, wspomnieliśmy zwłaszcza o rozkładzie dwumianowym i rozkładzie beta.

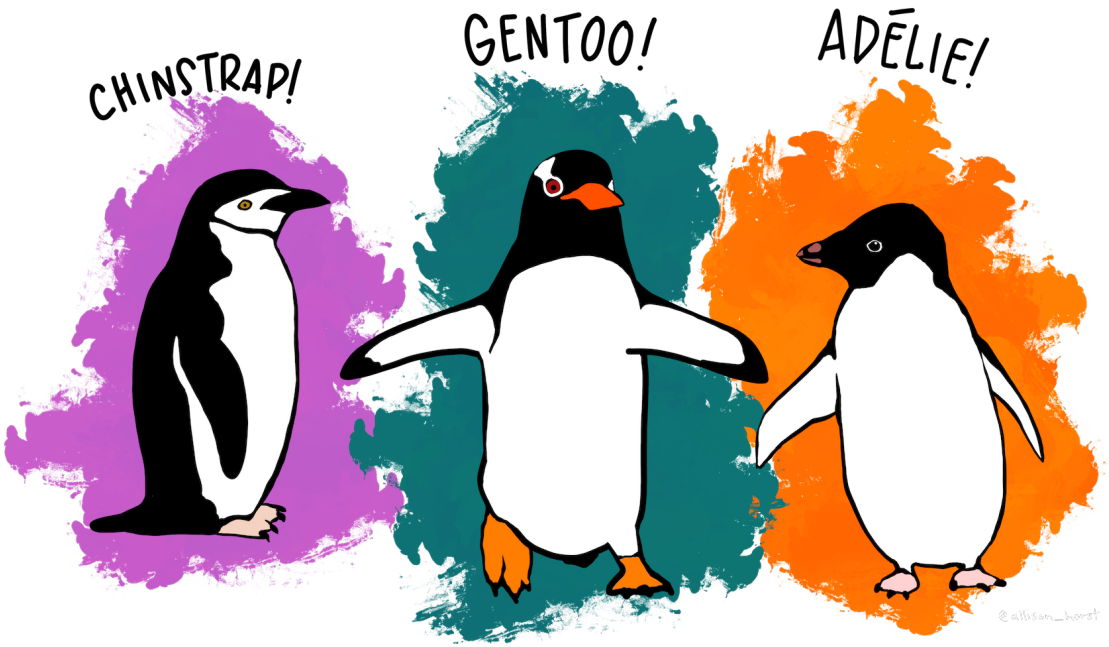
Jednak zdecydowanie najsłynniejszym rozkładem jest rozkład normalny.

Rozkład normalny, zwany też *rozkładem Gaussa*, to symetryczny rozkład w kształcie dzwonu, w którym większość masy skupia się wokół średniej, a rozrzut jest zdefiniowany jako odchylenie standardowe. „Ogony" po obu stronach stają się coraz cieńsze w miarę oddalania się od średniej.

Przyczyną jego znaczenia jest częstość występowania w naturze. Jeśli jakaś wielkość jest sumą lub średnią bardzo wielu drobnych losowych czynników, to niezależnie od rozkładu każdego z tych czynników jej rozkład będzie zbliżony do normalnego (centralne twierdzenie graniczne) – dlatego można go bardzo często zaobserwować w danych. Ponadto rozkład normalny ma interesujące właściwości matematyczne, dzięki którym oparte na nim metody statystyczne są proste obliczeniowo.

Palmer pinguins

Dane:
`https://gist.github.com/slopp/ce3b90b9168f2f921784de84fa445651`
Opis:
`https://allisonhorst.github.io/palmerpenguins/articles/intro.html`
`https://allisonhorst.github.io/palmerpenguins/`
`https://pypi.org/project/palmerpenguins/`

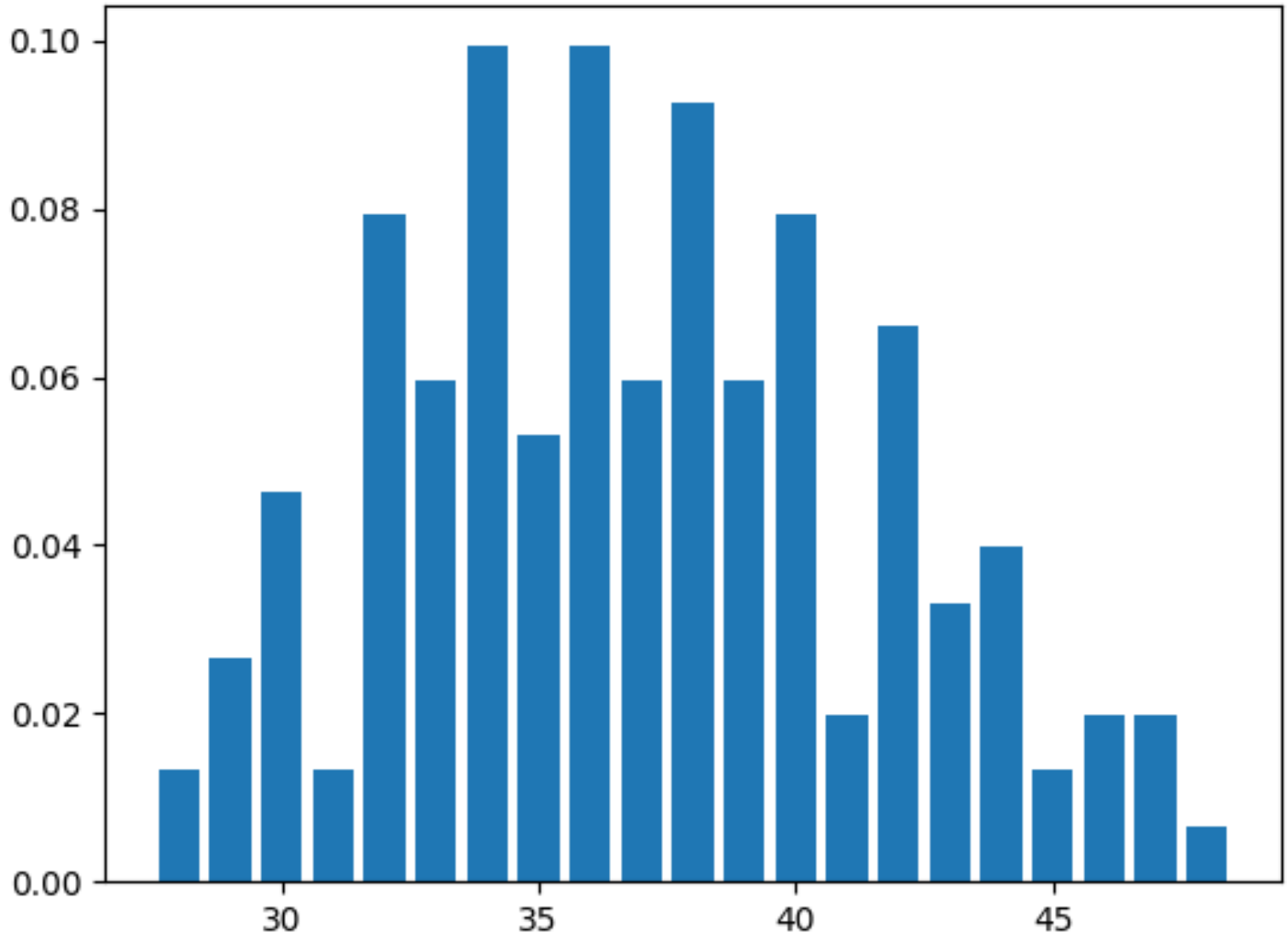


The Palmer Archipelago penguins. Artwork by @allison_horst

Palmer penguins: waga Adelie

```
weight0 = np.array([3750, 3800, 3250, 3450, 3650, 3625, 4675, 3475,
4250, 3300, 3700, 3200, 3800, 4400, 3700, 3450, 4500, 3325, 4200,
3400, 3600, 3800, 3950, 3800, 3800, 3550, 3200, 3150, 3950, 3250,
3900, 3300, 3900, 3325, 4150, 3950, 3550, 3300, 4650, 3150, 3900,
3100, 4400, 3000, 4600, 3425, 2975, 3450, 4150, 3500, 4300, 3450,
4050, 2900, 3700, 3550, 3800, 2850, 3750, 3150, 4400, 3600, 4050,
2850, 3950, 3350, 4100, 3050, 4450, 3600, 3900, 3550, 4150, 3700,
4250, 3700, 3900, 3550, 4000, 3200, 4700, 3800, 4200, 3350, 3550,
3800, 3500, 3950, 3600, 3550, 4300, 3400, 4450, 3300, 4300, 3700,
4350, 2900, 4100, 3725, 4725, 3075, 4250, 2925, 3550, 3750, 3900,
3175, 4775, 3825, 4600, 3200, 4275, 3900, 4075, 2900, 3775, 3350,
3325, 3150, 3500, 3450, 3875, 3050, 4000, 3275, 4300, 3050, 4000,
3325, 3500, 3500, 4475, 3425, 3900, 3175, 3975, 3400, 4250, 3400,
3475, 3050, 3725, 3000, 3650, 4250, 3475, 3450, 3750, 3700, 4000])

print(len(weight0)) #151
```

Waga pingwinów gatunku Adelie

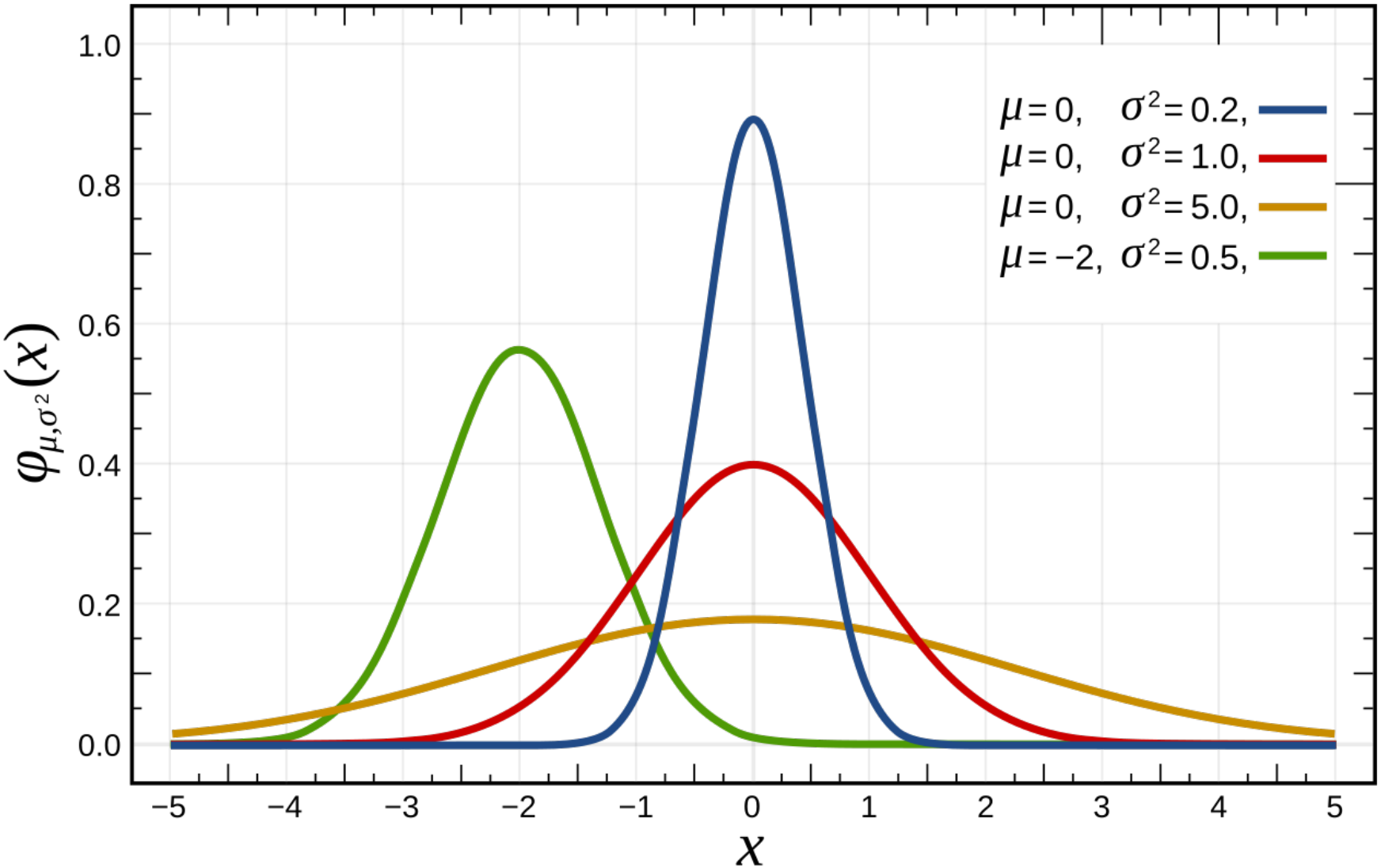


Waga pingwinów gatunku Adelie: kod

```
import numpy as np
import matplotlib.pyplot as plt

weight0 = np.array([3750, 3800, 3250, ...])
weight = np.array([round(i/100) for i in weight0])
data = {}
for i in  np.unique(weight):
    data[i] = 0
for i in weight:
    data[i] += 1
s = sum(data.values())
for i in data.keys():
    data[i] /= s

print(data)

plt.bar(*zip(*data.items()))
plt.show()
```

Dane:
https://gist.github.com/slopp/ce3b90b9168f2f921784de84fa445651

Rozkład normalny: obrazek z Wikipedii



Czerwona linia odpowiada standardowemu rozkładowi normalnemu , μ - mediana i średnia, σ^2 - wariancja.

Własności rozkładu normalnego

Rozkład normalny ma kilka własności, które sprawiają, że jest użyteczny:

- Jest symetryczny, obie strony są lustrzanym odbiciem wokół średniej, która jest w środku krzywej.
- Większość jego masy znajduje się w środku wokół średniej.
- Ma rozciągnięcie (jest wąski lub szeroki) określone przez odchylenie standardowe.
- „Ogony" są najmniej prawdopodobnymi wynikami i dążą do zera, ale nigdy go nie osiągają.
- Reprezentuje wiele zjawisk w naturze i codziennym życiu, a nawet generalizuje nietypowe problemy ze względu na centralne twierdzenie graniczne, o którym wkrótce porozmawiamy.

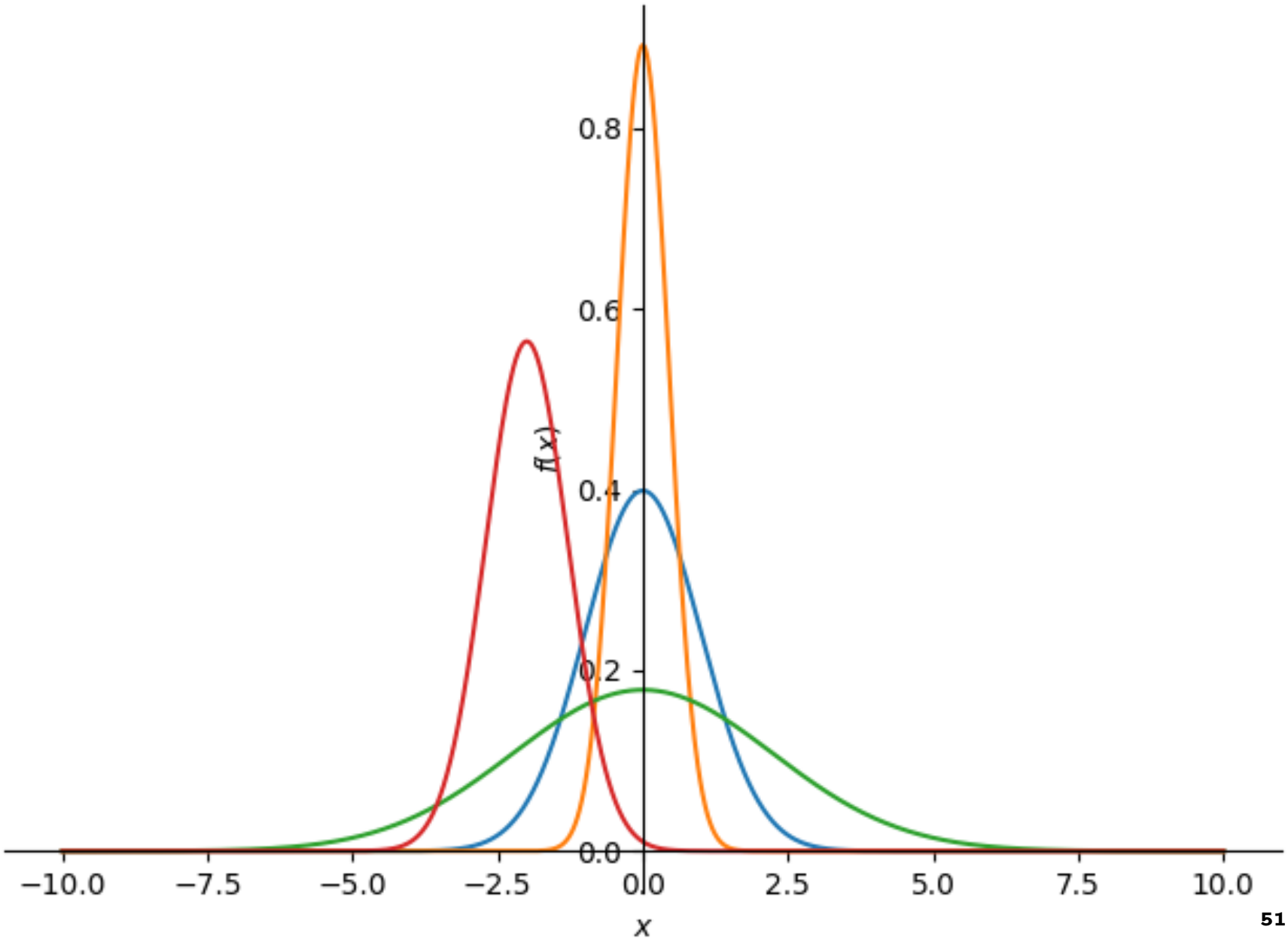
Funkcja gęstości

Funkcja gęstości prawdopodobieństwa rozkładu normalnego ze średnią μ i odchyleniem standardowym σ (równoważnie: wariancją σ^2) jest przykładem funkcji Gaussa. Dana jest ona wzorem:

$$f_{\mu,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right).$$

Fakt, iż zmienna losowa X ma rozkład normalny z wartością oczekiwaną μ i wariancją σ^2 zapisuje się często $X \sim \mathcal{N}(\mu, \sigma^2)$.

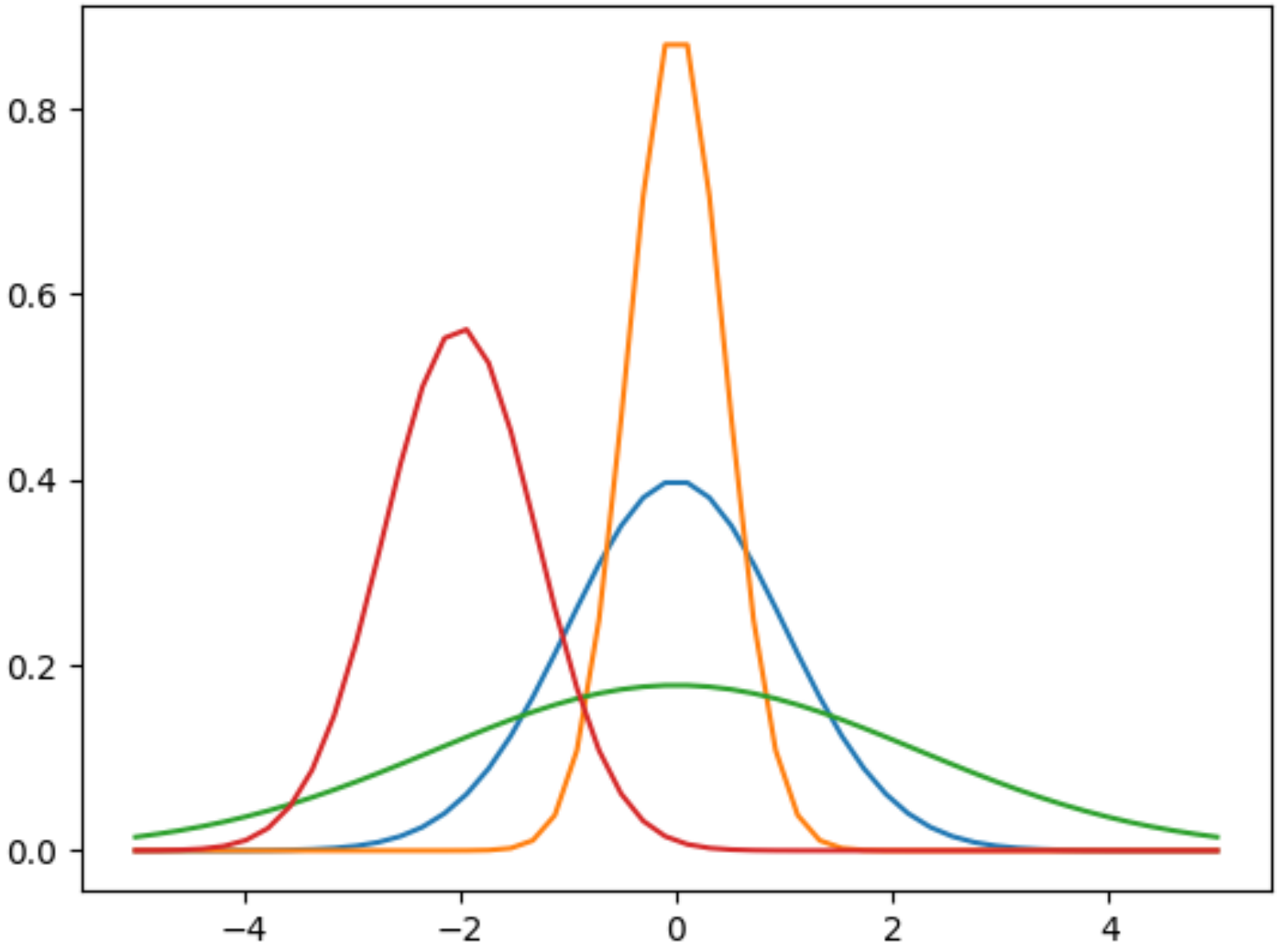
Rozkład normalny Python sympy



Rozkład normalny Python sympy: kod

```
import matplotlib as plt
from sympy import *

x = symbols('x')
def f(x, mu = 0, war = 1):
    sigma = sqrt(war)
    return 1/(sigma*sqrt(2*pi))*exp(-(x-mu)**2/(2*sigma**2))

p1 = plot(f(x), show=False)
p2 = plot(f(x, 0, 0.2), show=False)
p3 = plot(f(x, 0, 5), show=False)
p4 = plot(f(x, -2, 0.5), show=False)

p1.extend(p2)
p1.extend(p3)
p1.extend(p4)

p1.show()
```

Rozklad normalny Python sympy



Rozklad normalny Python numeryczne: kod

```
from math import *
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(-5,5)
def f(x, mu = 0, sigma2 = 1):
    sigma = sqrt(sigma2)
    return 1/(sigma*sqrt(2*pi))*exp(-(x-mu)**2/(2*sigma**2))

y1 = np.array([f(i) for i in x])
y2 = np.array([f(i,0,0.2) for i in x])
y3 = np.array([f(i,0,5) for i in x])
y4 = np.array([f(i,-2,0.5) for i in x])
plt.plot(x,y1)
plt.plot(x, y2)
plt.plot(x, y3)
plt.plot(x,y4)
plt.show()
```

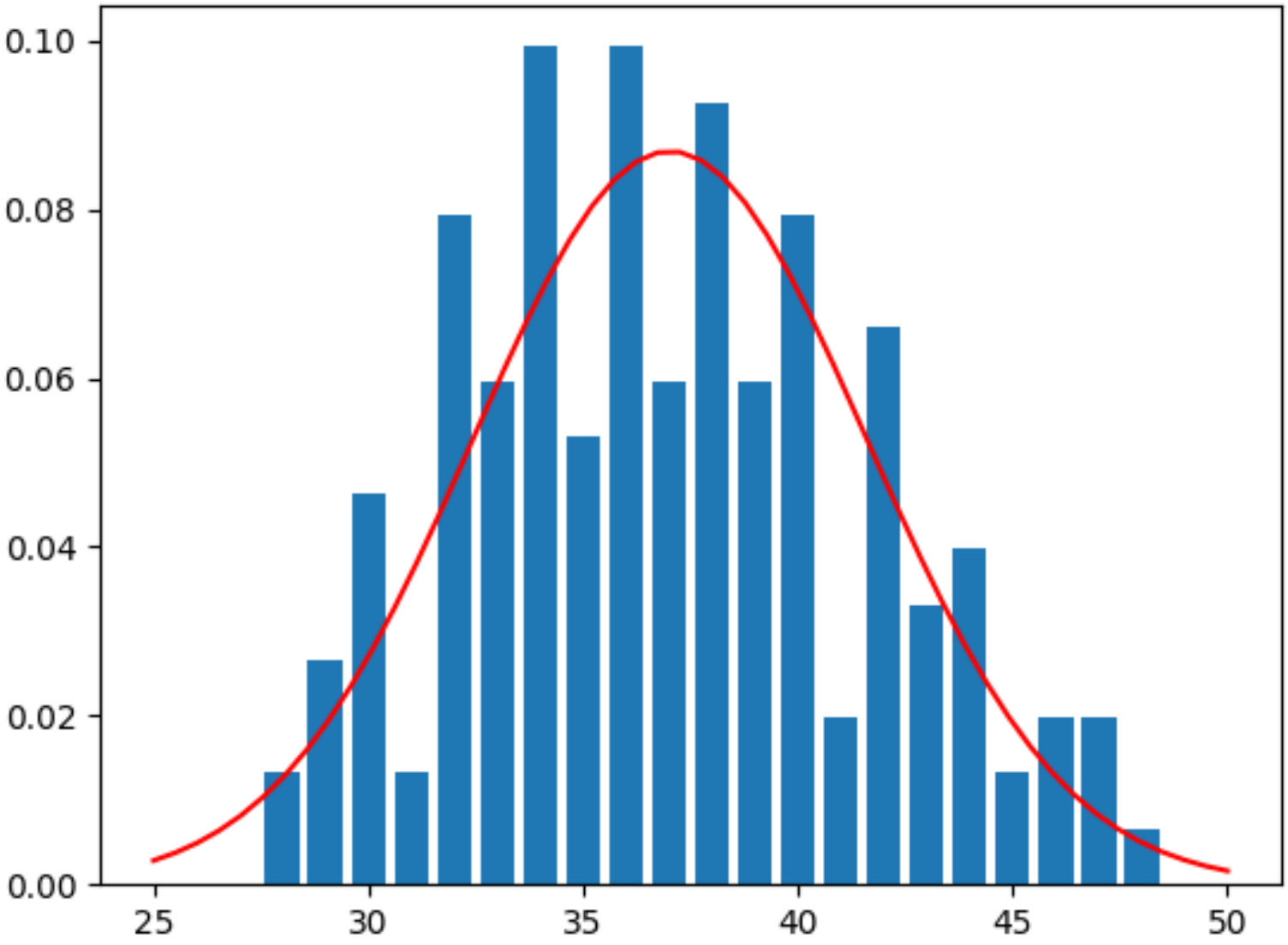
Pingwiny gatunku Adelie: kod

Dorysujemy do wykresy wagi pingwinów teoretyczne rozkład normalny:

```
from math import *

mu0 = np.mean(weight)
var0 = (np.var(weight))
print(mu0)
print(var0)

x = np.linspace(25,50)
def f(x, mu = 0, sigma2 = 1):
    sigma = sqrt(sigma2)
    return (1/(sigma*sqrt(2*pi)))*exp(-(x-mu)**2/(2*sigma**2))

y1 = np.array([f(i, mu0, var0) for i in x])

plt.plot(x,y1, 'r')
plt.show()
```

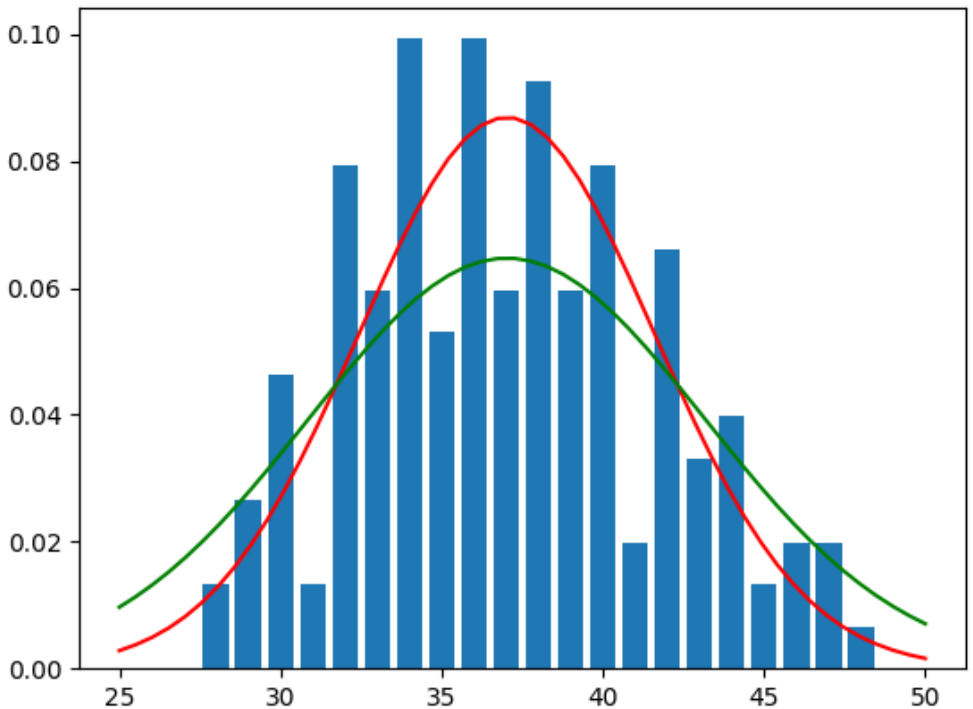
(pierwszeplt.show usunąć)

Pingwiny gatunku Adelie



Pingwiny gatunku Adelie

Oraz teoretyczne rozkład normalny z próby




```
import statistics as stat
#(przed plt.show())
y2 = np.array([f(i, mu0, stat.variance(data)) for i in x])
plt.plot(x,y2, 'g')
```

Dystrybuanta

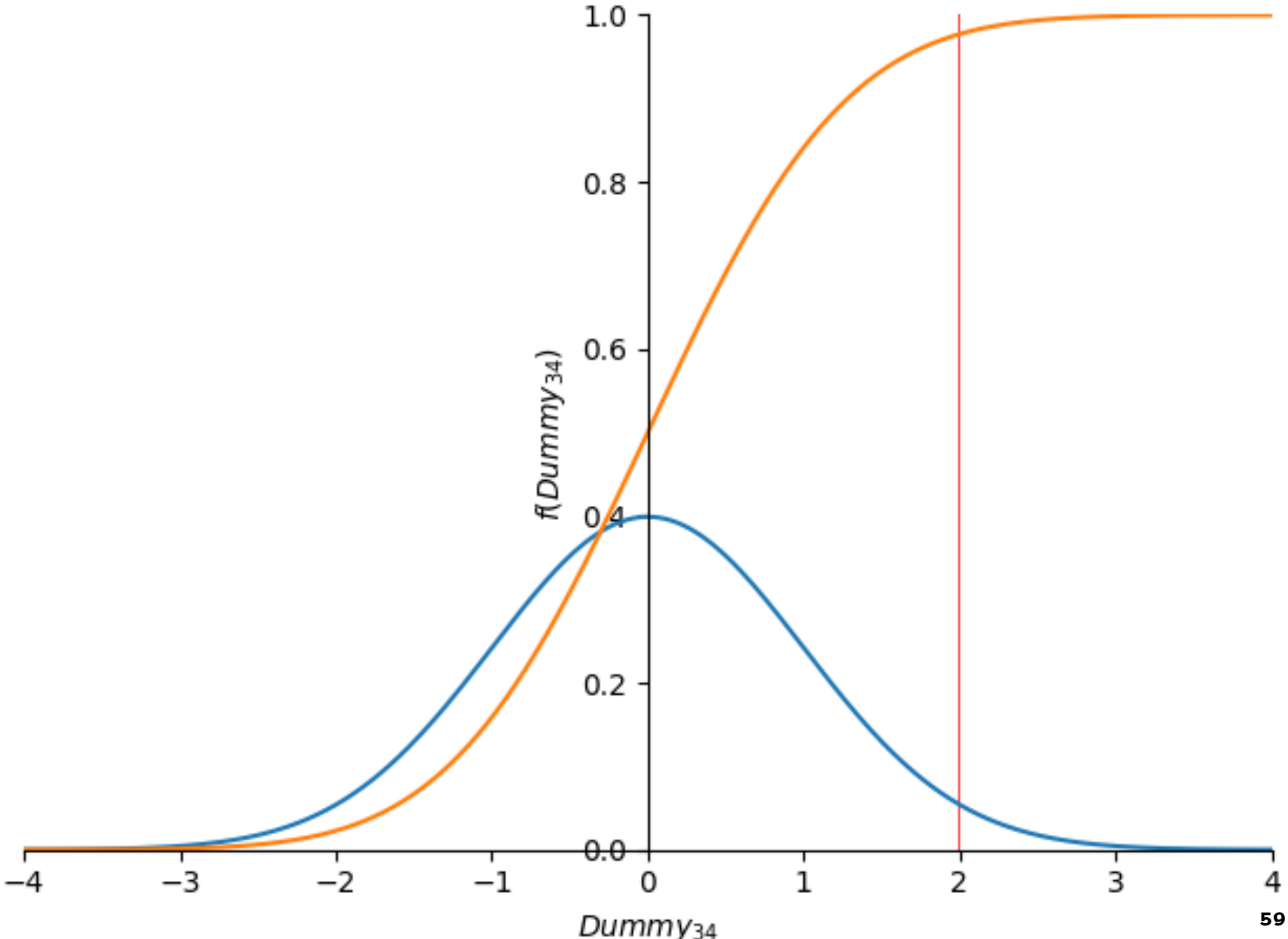
– funkcja, która zwraca pole obszaru do określonej wartości x dla danego rozkładu. Jeżeli $f(x)$ jest gęstość, to

$$F(x) = \int_{-\infty}^{x} f(t)dt$$

tzn. dla rozkładu normalnego ze średnią μ i wariancją σ:

$$F(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(\frac{-(t-\mu)^2}{2\sigma^2}\right).$$

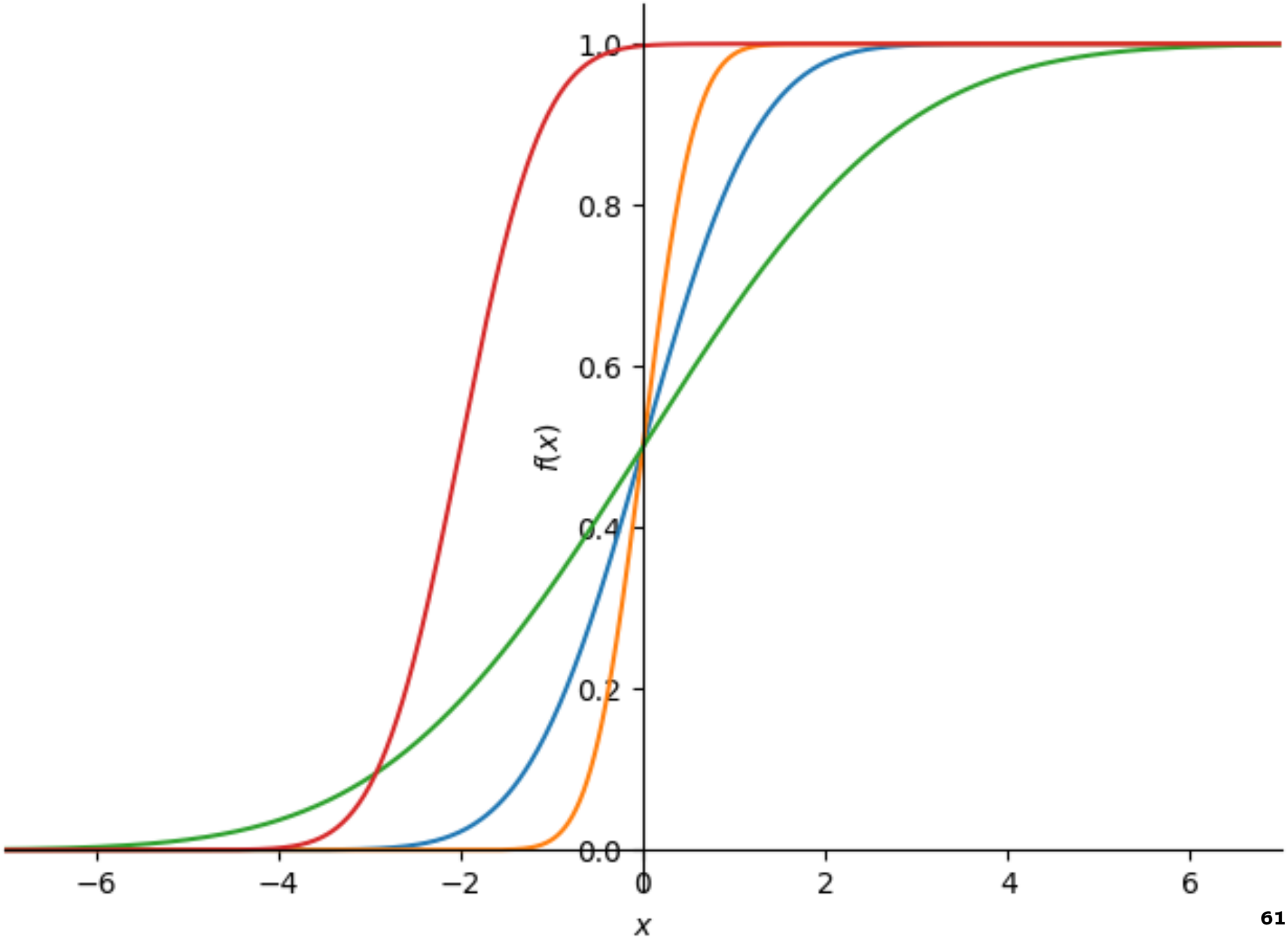
Dystrybuanta i gęstość rozkłady normalnego Python sympy



Dystrybuanta i gęstość rozkłady normalnego Python sympy: kod

```
import matplotlib as plt
from sympy import *

t, x = symbols('t, x')
def f(x, mu = 0, sigma2 = 1):
    sigma = sqrt(sigma2)
    return 1/(sigma*sqrt(2*pi))*exp(-(x-mu)**2/(2*sigma**2))

def F(x, mu = 0, sigma2 = 1):
    return integrate(f(t, mu, sigma2),(t,-oo,x))

p1 = plot(f, xlim=[-4,4], ylim=[0,1], show=False)
p2 = plot(F(x),  show=False)
p1.extend(p2)
x0 = Symbol('x0')
line = plot_implicit(Eq(x0, 2),line_color='r', show=False)
p1.extend(line)
p1.show()
```

Dystrybuanta rozkłady normalnego Python sympy



Dystrybuanta rozkłady normalnego Python sympy: kod

```
import matplotlib as plt
from sympy import *

t, x = symbols('t, x')
def f(x, mu = 0, sigma2 = 1):
    sigma = sqrt(sigma2)
    return 1/(sigma*sqrt(2*pi))*exp(-(x-mu)**2/(2*sigma**2))

def F(x, mu = 0, sigma2 = 1):
    return integrate(f(t, mu, sigma2),(t,-oo,x))

p1 = plot(F(x), xlim=[-7,7], show=False)
p2 = plot(F(x, 0,0.2),  show=False)
p3 = plot(F(x, 0, 5), show=False)
p4 = plot(F(x, -2, 0.5), show=False)
p1.extend(p2)
p1.extend(p3)
p1.extend(p4)
p1.show()
```