Matematyczne aspekty analizy danych
semestr zimowy 2024/2025

Dr Anna Muranova
UWM w Olsztynie

Wykład 5

Rozkład normalny

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

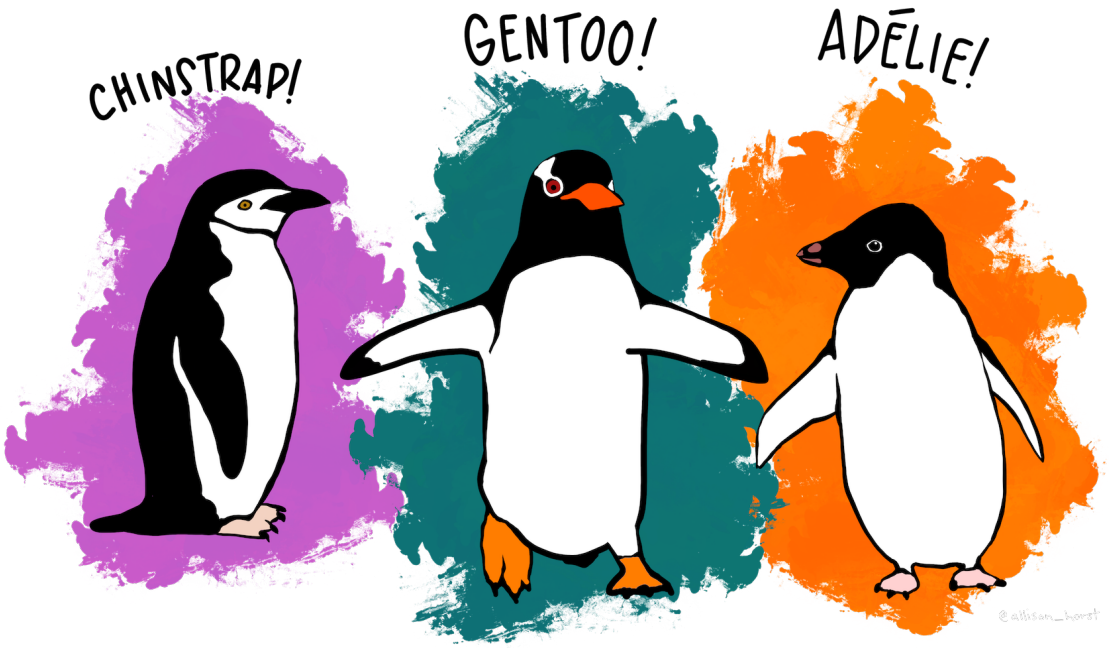
Jednak zdecydowanie najsłynniejszym rozkładem jest rozkład normalny.

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/`

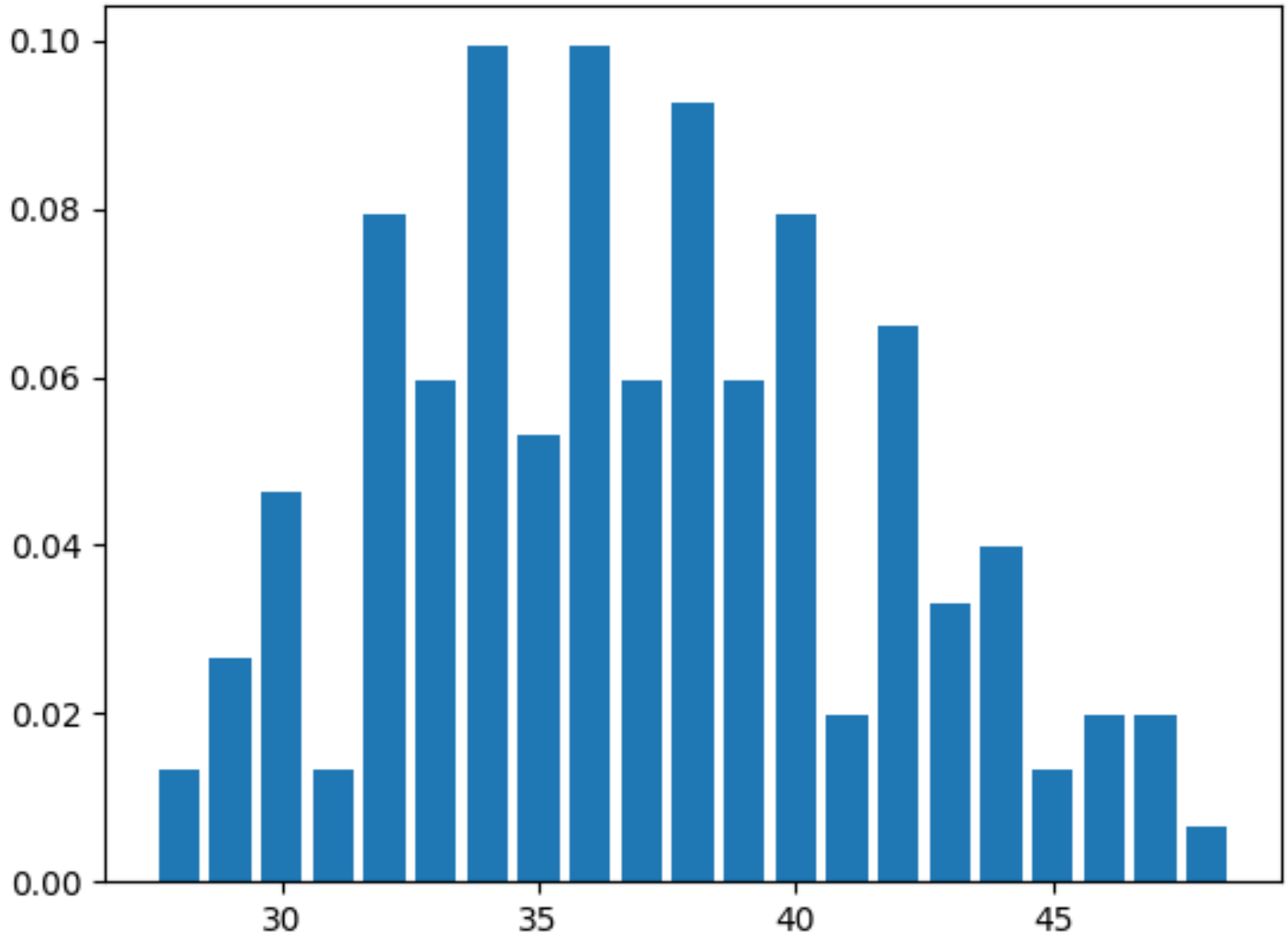


The Palmer Archipelago penguins. Artwork by @allison_horst

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

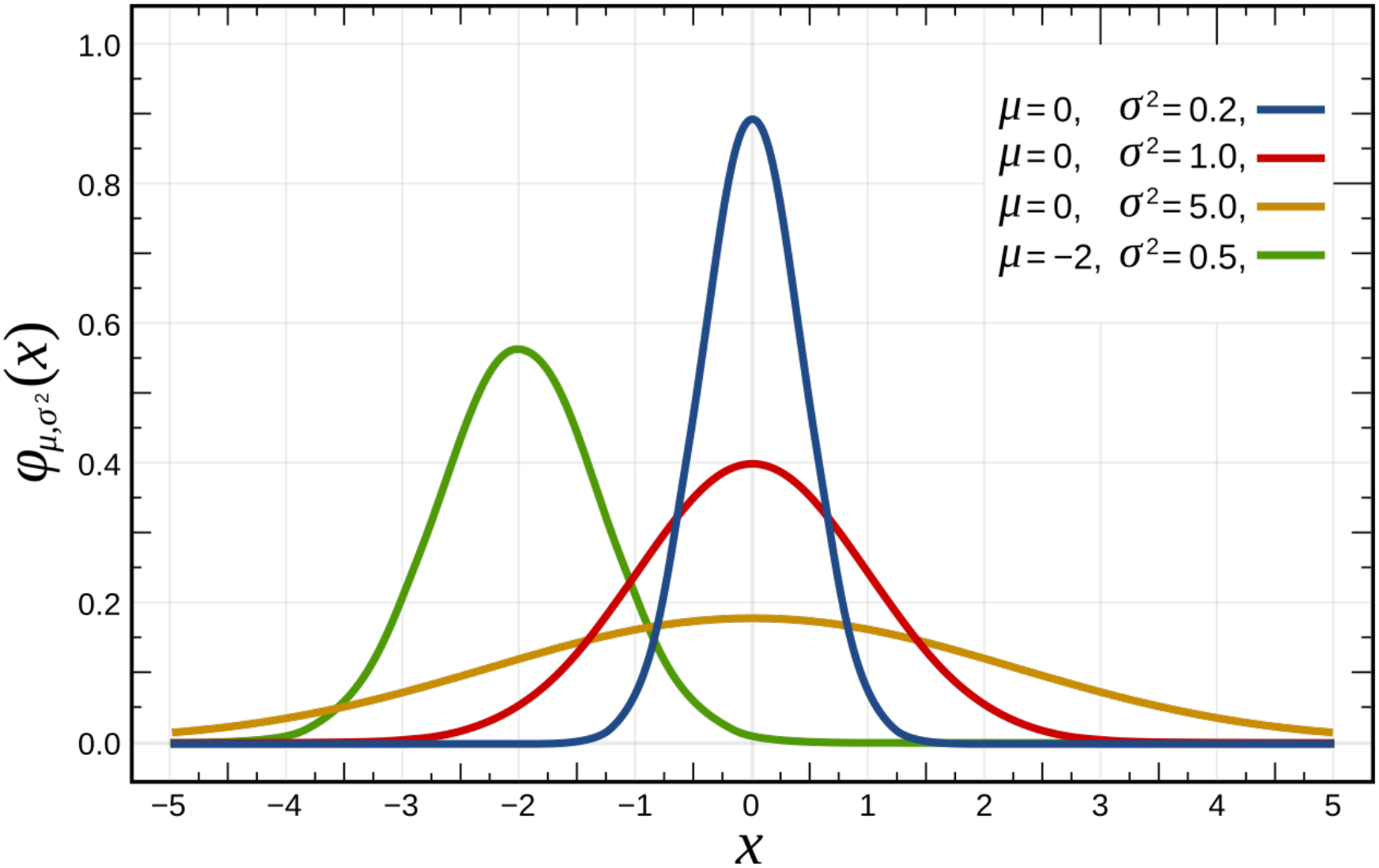


Waga pingwinów gatunku Adelie: kod

```
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.

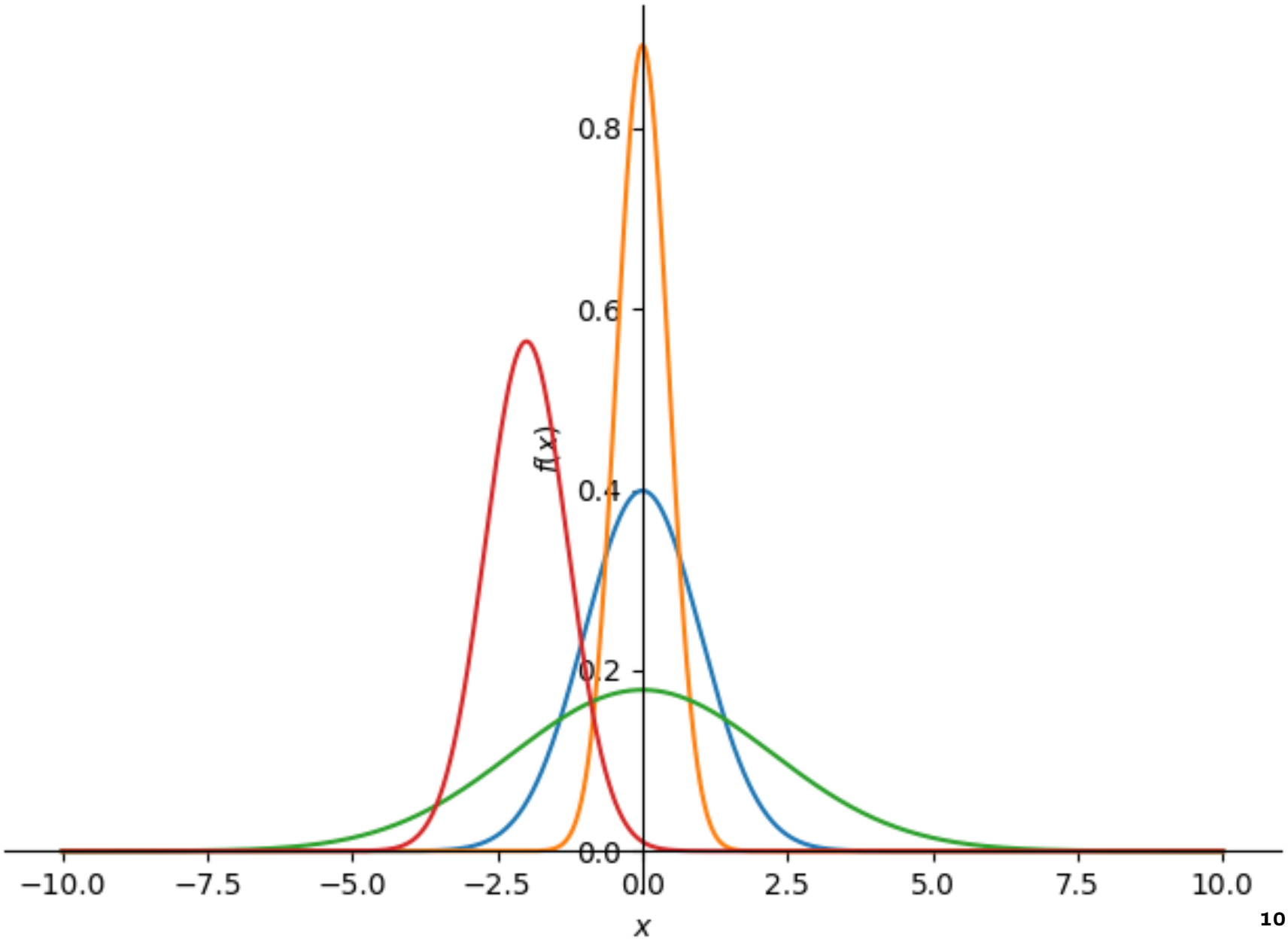
Funkcja gęstości

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 $X \sim \mathcal{N}(\mu, \sigma^2)$.

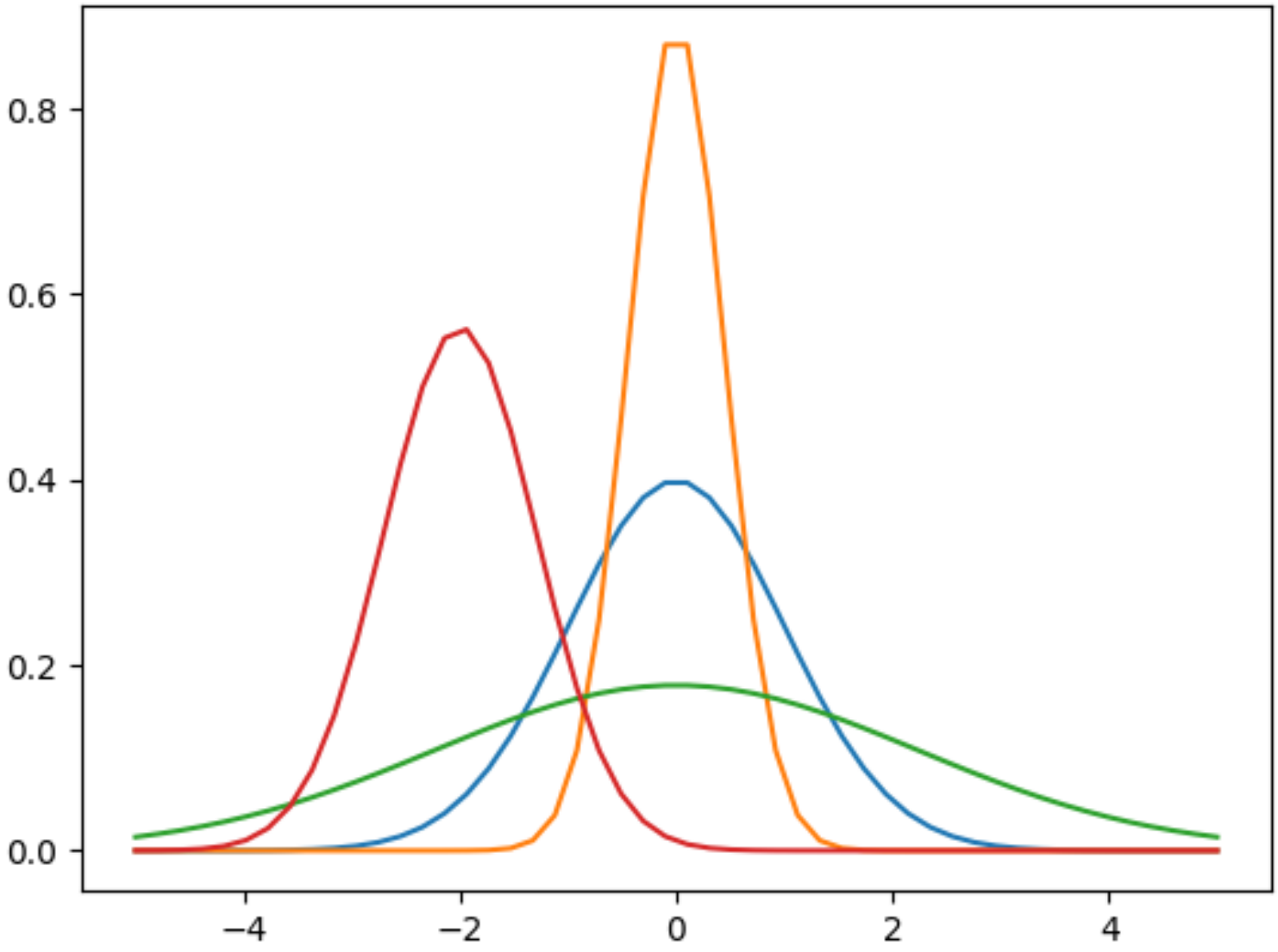
Rozkład normalny Python sympy



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()
```

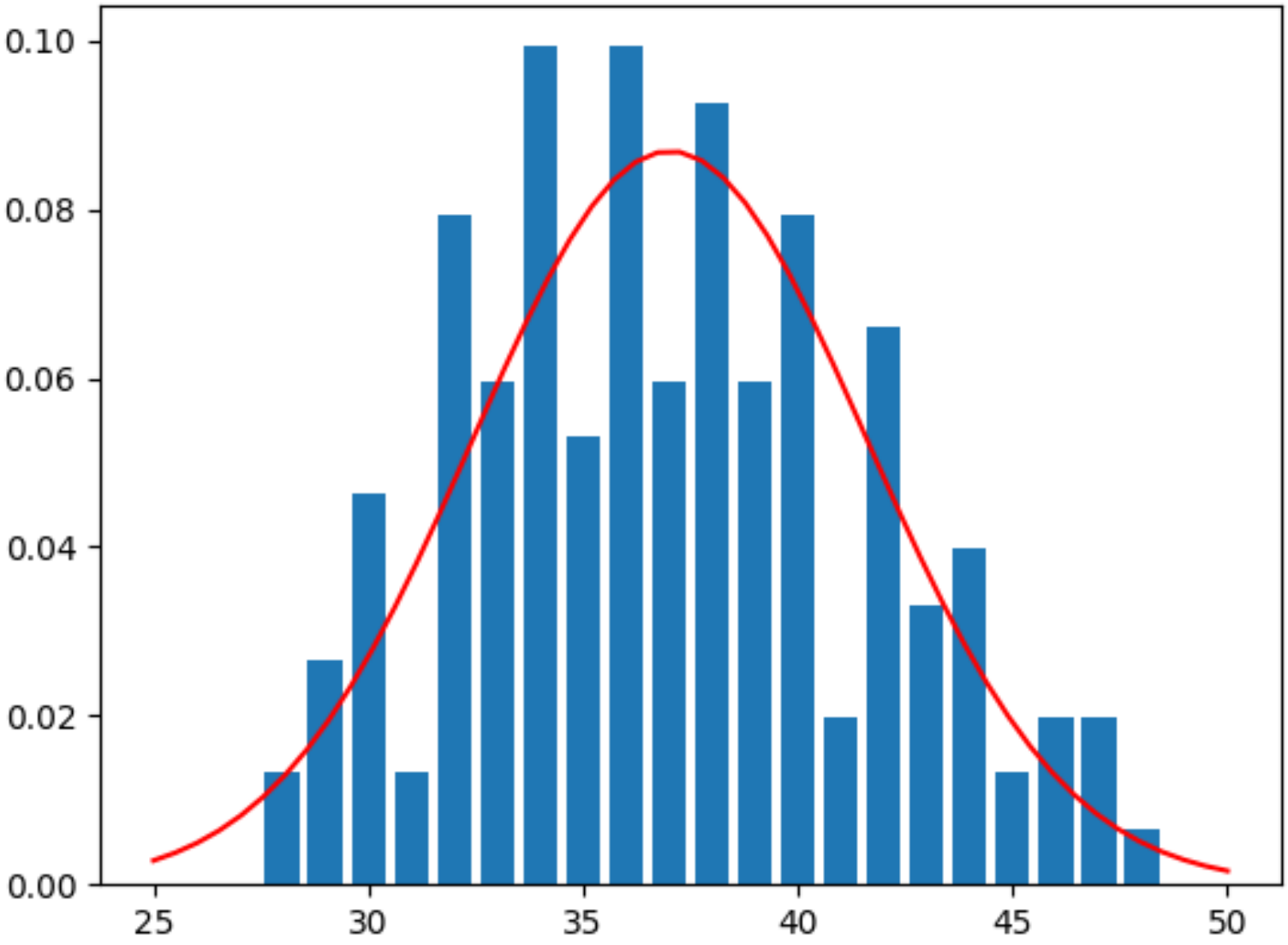
Pingwiny gatunku Adelie: kod

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()
```

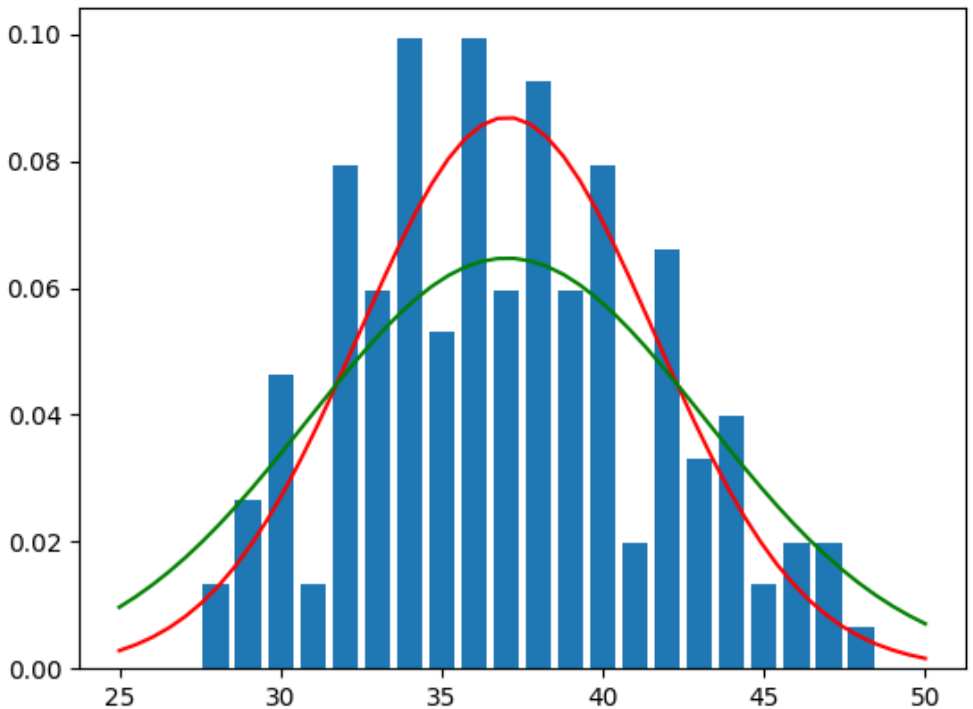
(pierwszeplt.show usunąć)

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

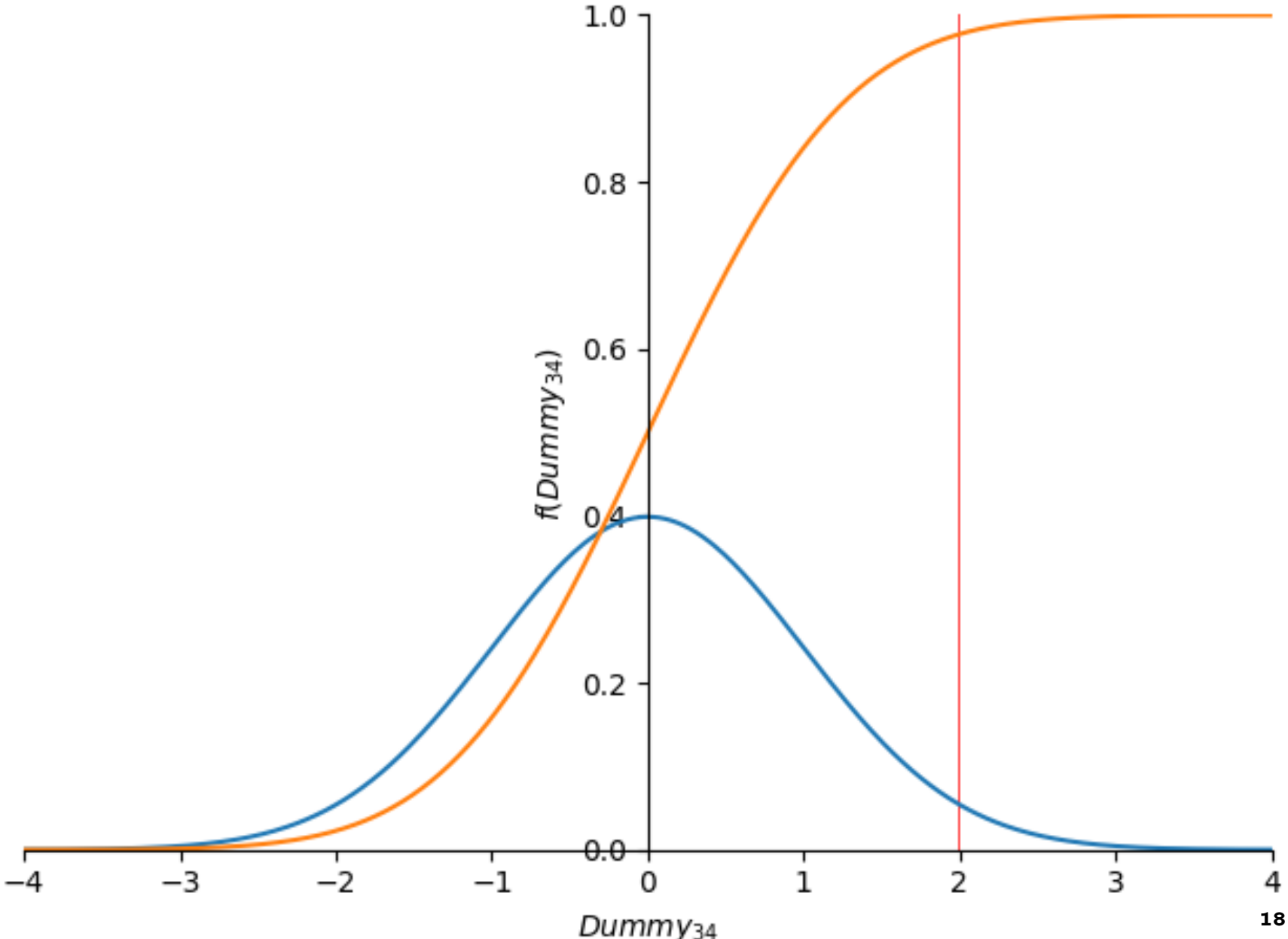
– funkcja, która zwraca pole obszaru do określonej wartości x dla danego rozkładu. Jeżeli $f(x)$ jest gęstość, to

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

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

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

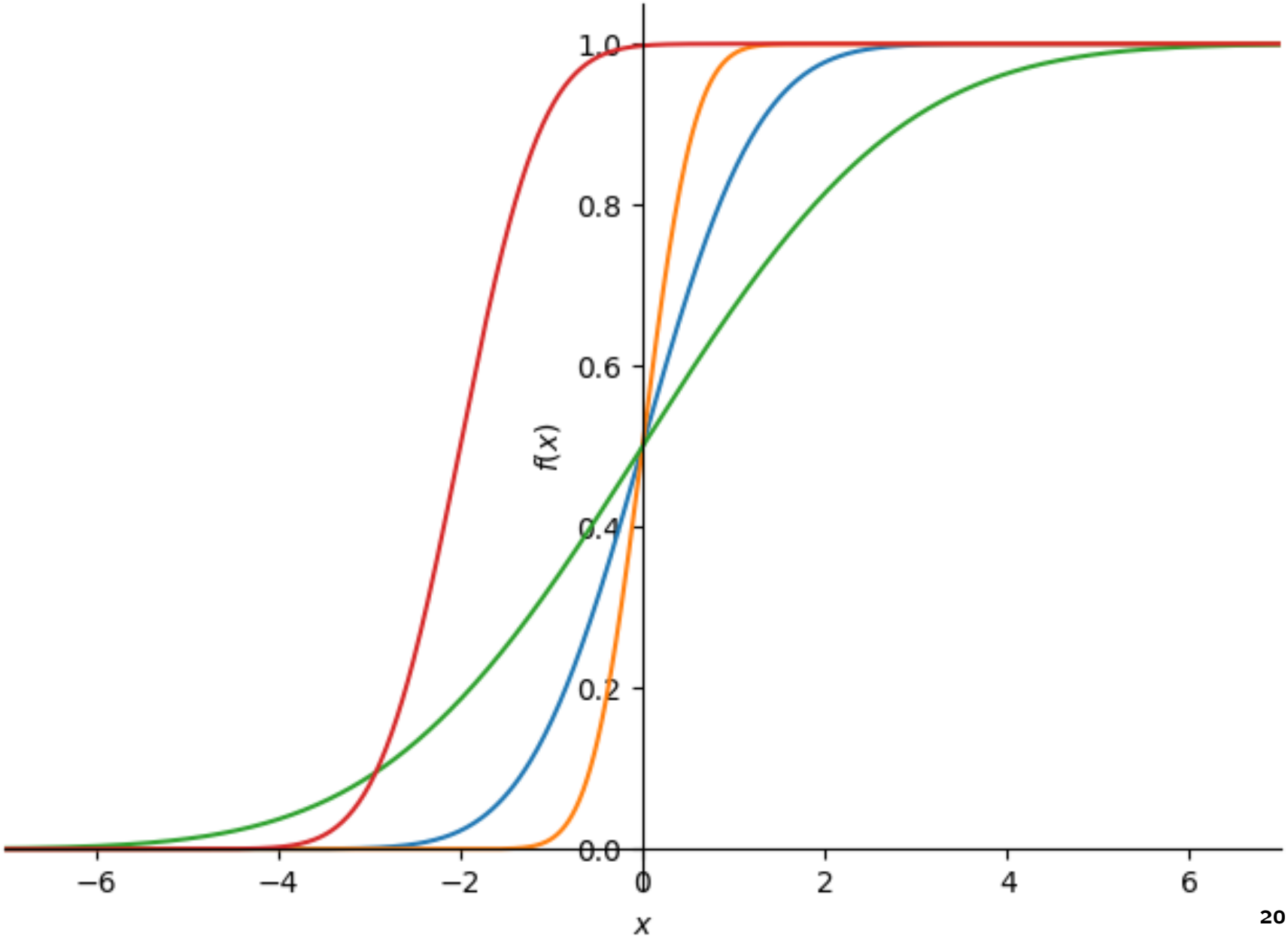
Dystrybuanta i gęstość rozkłady normalnego Python sympy



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

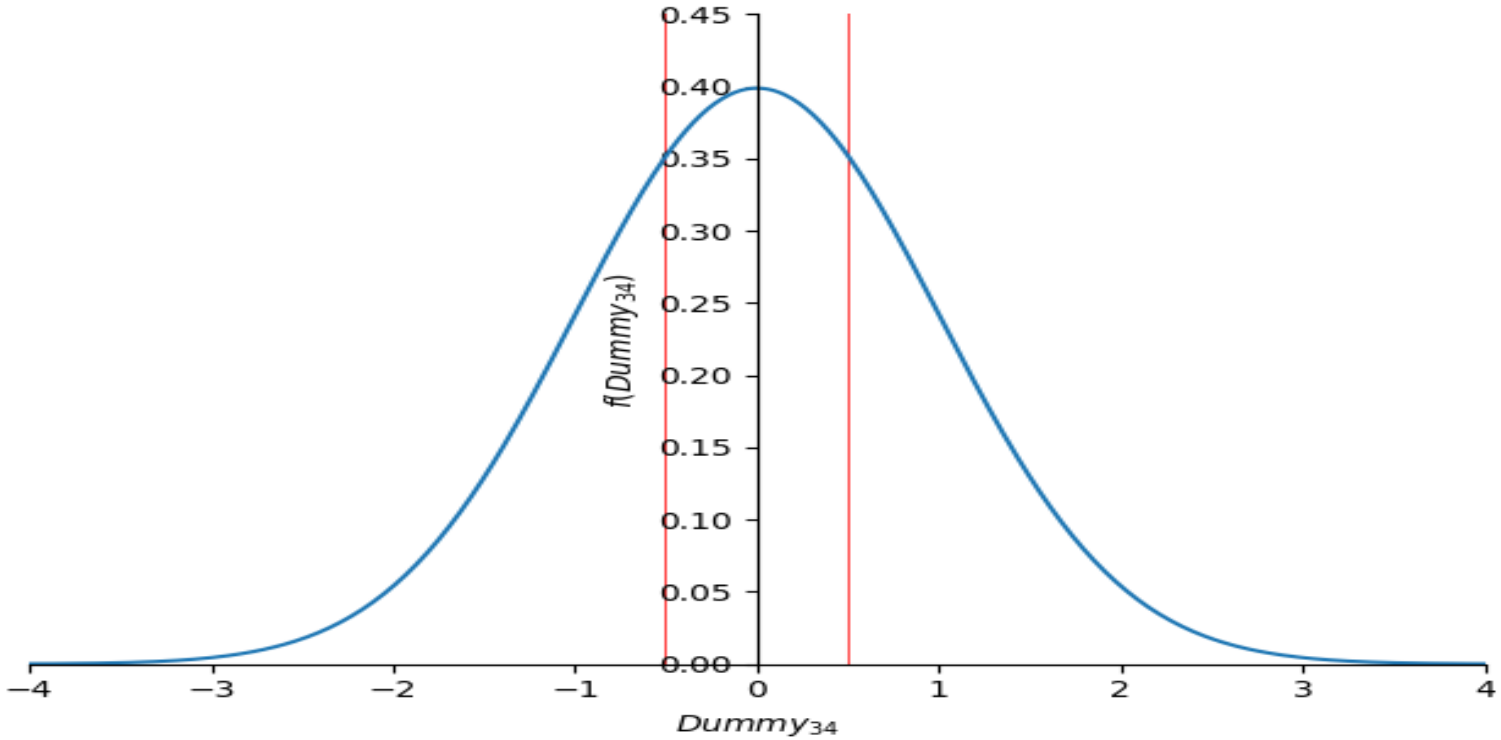
```
import matplotlib as plt
from sympy import *

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()
```

Obliczenie dystrybuanty w scipy.stats: kod

```
from scipy.stats import norm

mu = 0
sigma = 1

x = norm.cdf(0,mu,sigma)
print(x)  #0.5
print(norm.cdf(0.5,mu,sigma)) #0.6914624612740131
print(norm.cdf(1,mu,sigma))#0.8413447460685429
print(norm.cdf(1.5,mu,sigma))#0.9331927987311419
print(norm.cdf(1.7,mu,sigma))#0.955434537241457
print(norm.cdf(2,mu,sigma))#0.9772498680518208
```

CDF – cumulative distribution function

Obliczenie prawdopodobieństwa środkowego zakresu w `scipy.stats`




```
from scipy.stats import norm

mu = 0
sigma = 1

x = norm.cdf(0.5,mu,sigma) - norm.cdf(-0.5,mu,sigma)
print(x)  # 0.3829492254802624
```

Obliczenie prawdopodobieństwa środkowego zakresu w `scipy.stats`: kod obrazku

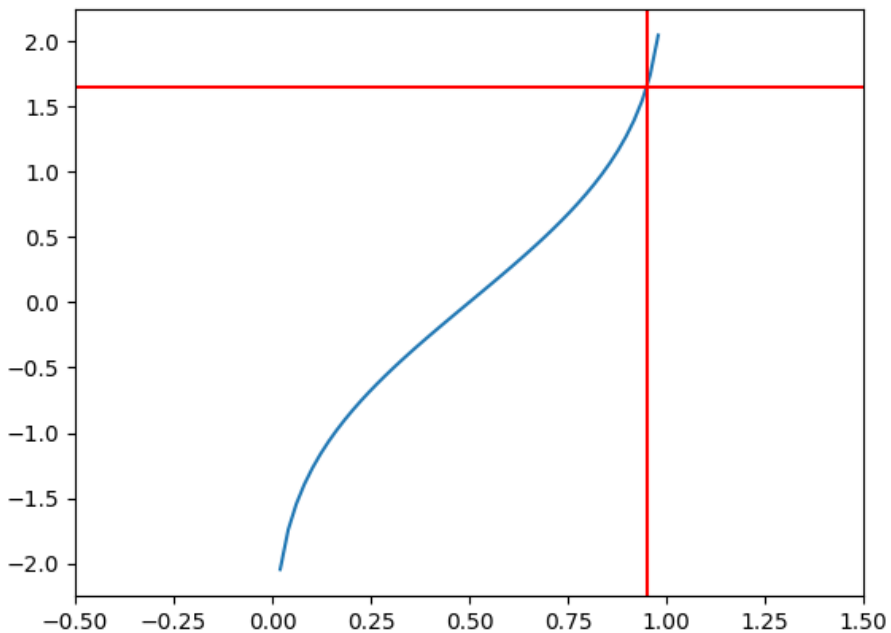
```
import matplotlib as plt
from sympy import *

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=[0,0.45], show=False)
x0 = Symbol('x0')
line = plot_implicit(Eq(x0, 0.5),line_color='r', show=False)
p1.extend(line)
line = plot_implicit(Eq(x0, -0.5),line_color='r', show=False)
p1.extend(line)
p1.show()
```

Obliczenie dystrybuanty dla pingwinów

```
import numpy as np
from scipy.stats import norm
import statistics as stat

weight0 = np.array([3750, 3800, 3250, .....])
weight = np.array([round(i/100) for i in weight0])
mu = np.mean(weight)
print(mu) # 37.01986754966887
sigma = (np.std(weight))
print(sigma) #4.589752765240528

sigmaProba = (stat.stdev(weight.tolist()))
print(sigmaProba) #4.605026527141107
print(norm.cdf(40,mu,sigma)-norm.cdf(35,mu,sigma)) #0.4119893168549857
print(norm.cdf(40,mu,sigmaProba)-norm.cdf(35,mu,sigmaProba)) #0.4107642
print(len(weight[(35 < weight) & (weight < 40)])/len(weight)) #0.311258

print(norm.cdf(45,mu,sigma)-norm.cdf(30,mu,sigma)) #0.8958803032421255
print(norm.cdf(45,mu,sigmaProba)-norm.cdf(30,mu,sigmaProba)) #0.8947395
print(len(weight[(30 < weight) & (weight < 45)])/len(weight)) #0.854304
```

Dystrybuanta odwrotna

Kiedy zajmiemy się testowaniem hipotez, będziemy chcieli odwrotnie znaleźć x na podstawie wartości pola pod krzywej. Wiec będziemy musieli użyć funkcje, odwrotnej dystrybuancie (PPf – percent-point function)



Dystrybuanta odwrotna: kod

```
from scipy.stats import norm
import matplotlib.pyplot as plt
import numpy as np

mu = 0
sigma = 1
x = np.linspace(0,1)
y = [norm.ppf(i, loc=mu,scale=sigma) for i in x]

print(norm.ppf(0.95, loc=mu,scale=sigma)) #1.644853626951472
plt.plot(x,y)
plt.xlim([-0.5,1.5])
plt.axvline(x = 0.95, color = 'r')
plt.axhline(y = norm.ppf(0.95, loc=mu,scale=sigma), color = 'r')

plt.show()
```

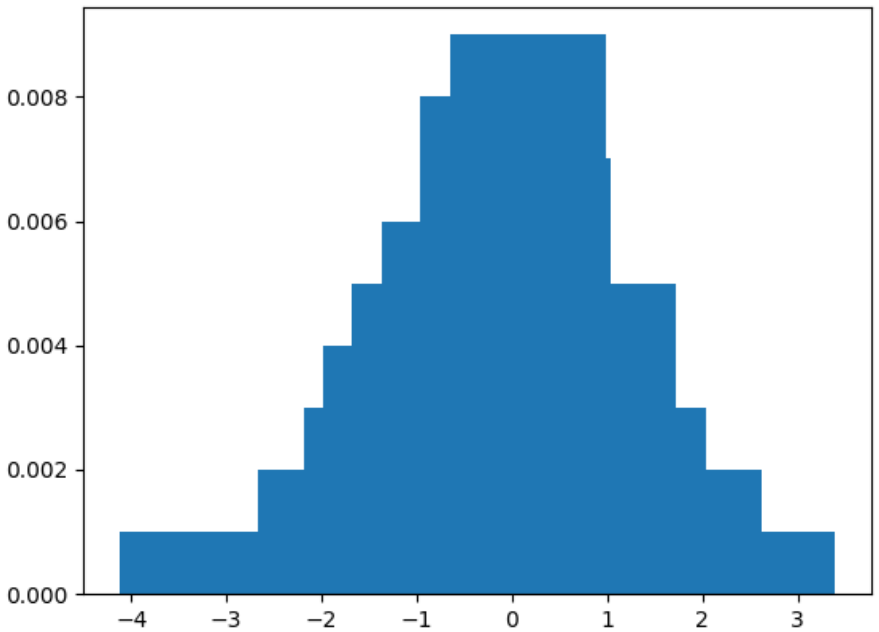
Dystrybuanta odwrotna

Mniej niż ile waży 95% pingwinów?

```
from scipy.stats import norm
import matplotlib.pyplot as plt
import numpy as np

mu = 37
sigma = 4.6
print(norm.ppf(0.95, loc=mu,scale=sigma)) #44.566326

weight0 = np.array([3750, 3800, 3250...])
weight = np.array([round(i/100) for i in weight0])

print(len(weight[weight < 44 ])/len(weight))#0.9006622516556292
```

Generowanie liczb losowych zgodnie z rozkładem normalnym

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

x = np.random.normal(loc=0.0, scale=1.0, size=1000)
print(x)
```



Kod do poprzedniego obrazku

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

x = np.random.normal(loc=0.0, scale=1.0, size=1000)
print(x)

x = [round(i,2) for i in x]

data = {}
for i in np.unique(x):
    data[i] = 0
for i in x:
    data[i] += 1
s = sum(data.values())
for i in data.keys():
    data[i] /= s

print(data)

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

Standaryzacja

Rozkład normalny często przeskalowuje się tak, aby średnia wynosiła 0, a odchylenie standardowe 1, co określa się mianem standardowego rozkładu normalnego.
Wzór:

$$Z = \frac{X - \mu}{\sigma}$$

Właściwości standardowego rozkładu normalnego:

- Ułatwia on porównanie rozproszenia dwóch rozkładów normalnych, nawet jeśli mają one różne średnie i wariancje.
- Wyraża on wszystkie wartości x w kategoriach odchyleń standardowych znanych jako *wyniki* Z (ang. Z-score)

Standaryzacja: przykład

Mamy dwa domy w dwóch różnych dzielnicach. W dzielnicy A średnia cena domu wynosi 140 000 dol. z odchyleniem standardowym 3 000. W dzielnicy B średnia w domu wynosi 800000 dol. z odchyleniem standardowym 10 000. Niech mamy po jednym domu w każdej dzielnice. Dom z dzielnicy A kosztuje 150 000 dol, a tamten z dzielnice B – 815 000 dol. Który dom jest droższy względem średniej ceny domu w swojej dzielnice?

$$\mu_A = 140\,000, \sigma_A = 3\,000, x_A = 150\,000$$

$$\mu_B = 800\,000, \sigma_A = 10\,000, x_B = 815\,000$$

Po standaryzacje:

$$Z_A = \frac{150\,000 - 140\,000}{3\,000} = 3,333333...$$

$$Z_B = \frac{815\,000 - 800\,000}{10\,000} = 1,5$$

Zatem dom w dzielnicy A jest względnie droższy niż w dzielnice B.

Kod w Pythonie

Przekształcanie Z-scores w wartości x i odwrotnie:

```
def wynik_z (x, srednia, std):
    return (x - srednia) / std

def z_na_x (z, srednia, std):
    return (z std) + srednia

srednia = 140000
odch_std = 3000
x = 150000

# Przekształcamy na Z-score a następnie z powrotem na X
z = wynik_z (x, srednia, odch_std)
z powrotem_na_x = z_na_x (z, srednia, odch_std)

print("Wynik Z: {}".format(z)) # Wynik Z: 3.333
print("Z powrotem na X: {}".format(z_powrotem_na_x))
#Z powrotem na X: 150000.0
```

Współczynnik zmienności

Przydatnym narzędziem do pomiaru rozproszenia jest *współczynnik zmienności*. Pozwala porównać dwa rozkłady i ocenić, jak bardzo rozproszony jest każdy z nich. Łatwo go obliczyć przez średnią. Oto wzór:

$$cv = \frac{\sigma}{\mu}$$

oraz przykład porównujący dwie dzielnicy::

$$cv_A = \frac{3\ 000}{140\ 000} = 0.0214$$

$$cv_B = \frac{10\ 000}{800\ 000} = 0.0125.$$

Jak widać, dzielnica A, choć tańsza od dzielnicy B, ma większe zróżnicowanie.

Współczynnik zmienności: jeszcze przykład

Kontrola w piekarni wykazała, iż średnia waga bochenka chleba to 500 gramów, zaś odchylenie standardowe to $2,5$ grama. Średnia waga ciastka z kremem to 115 gramów, zaś odchylenie $2,4$ grama.

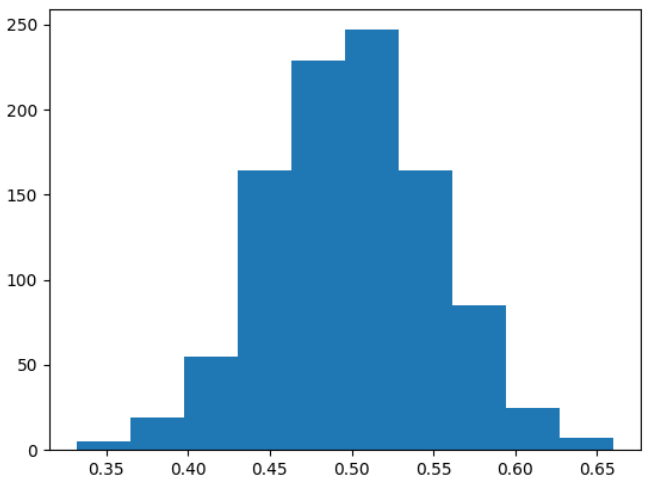
$$V_{chleba} = \frac{2{,}5}{500} \cdot 100\% = 0{,}5\%,$$

$$V_{ciastka} = \frac{2{,}4}{115} \cdot 100\% = 2{,}087\%.$$

Pomimo iż obie badane populacje charakteryzowały się podobnym odchyleniem, to współczynnik zmienności ciastek z kremem jest ponad 4-krotnie wyższym niż chleba.

Centralne twierdzenie graniczne (CTG)

Rozkład normalny jest użyteczny między innymi dlatego, że często występuje w naturze (nprz. waga pingwinów). Pojawia się jednak również w bardziej interesującym kontekście poza populacjami naturalnymi. Kiedy zaczynamy mierzyć wystarczająco duże próby populacji, nawet takiej, która nie podlega rozkładowi normalnemu, i tak dochodzimy rozkładu normalnego.
Przypuśćmy, że mierzę populację, która jest naprawdę i jednolicie losowa. Każda wartość od $0,0$ do $1,0$ jest jednakowo prawdopodobna i żadna nie jest preferowana. Kiedy jednak zaczynamy brać coraz większe próby z tej populacji, uśredniać je i nanosić na histogram, dzieje się coś fascynującego.



Kod w Pythonie

```
import random
import matplotlib.pyplot as plt

rozmiar_probki = 31
liczba_probek = 1000

#Centralne twierdzenie graniczne, 1000 prób,
# każda z 50 liczbami losowymi od 0,0 do 1,0

wartosci_x = [sum([random.uniform (0.0, 1.0) for i in
  range(rozmiar_probki)])/rozmiar_probki for _ in range(liczba_probek)]

# mozna rozpisac przy pomocy petli:
#wartosci_x =[]
#for _ in range(liczba_probek):
#    wartosci_x.append(sum([random.uniform (0.0, 1.0) for i
#         in range(rozmiar_probki)])/ rozmiar_probki)

print(wartosci_x)
plt.hist(x=wartosci_x)
plt.show()
```

Centralne twierdzenie graniczne: sformułowanie

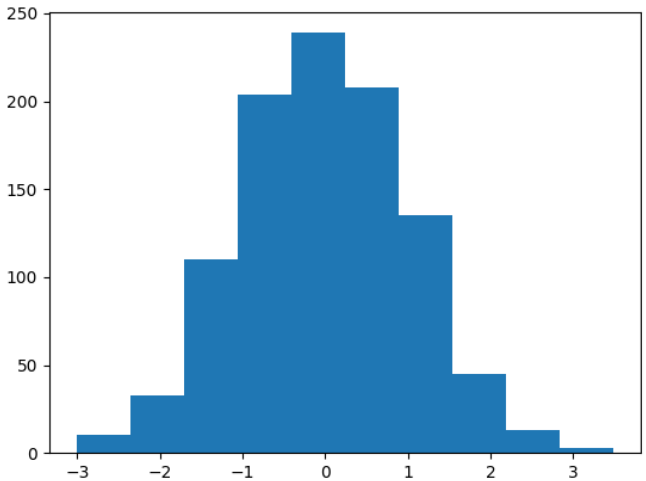
Jeśli X_i są niezależnymi zmiennymi losowymi pochodzącymi z tej samej populacji o wartości oczekiwanej μ oraz dodatniej i skończonej wariancji σ^2, to ciąg zmiennych losowych, w postaci znormalizowanych wartości oczekiwanych U_n

$$U_n = \frac{\frac{1}{n}\sum_{i=1}^{n} X_i - \mu}{\sigma/\sqrt{n}}$$

zbieżny jest według rozkładu do standardowego rozkładu normalnego, gdy $n \to +\infty$.

Tzn.

$$\lim_{n\to\infty} P(U_n < u) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{u} e^{-x^2/2}\, dx$$



Kod w Pythonie dla U_n

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

def u (x, n):
    return (x - 0.5) / (np.sqrt(1/12)/np.sqrt(n))

rozmiar_probki = 31
liczba_probek = 1000

wartosci_x = np.array([sum([random.uniform (0.0, 1.0) for i in
   range(rozmiar_probki)])/rozmiar_probki for _ in range(liczba_probek)]

print(np.mean(u(wartosci_x, rozmiar_probki)))
print(np.var(u(wartosci_x, rozmiar_probki)))
plt.hist(x=u(wartosci_x, rozmiar_probki))
plt.show()
```

Ponieważ przy rozkładzie jednostajnym na $[0, 1]$ zachodzi $\mu = 0.5, \sigma^2 = 1/12$.

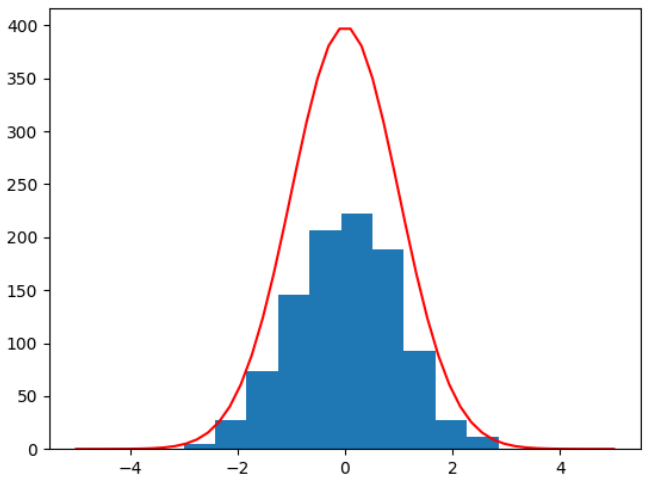
https://pl.wikipedia.org/wiki/Rozk%C5%82ad_jednostajny_ci%C4%85g%C5%82y

Porównanie z teoria

Dorysujemy do wykresy teoretyczne rozkład normalny:

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

y1 = np.array([f(i)*liczba_probek for i in x]) #uwaga na mnożenie

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

(pierwsze `plt.show` usunąć)



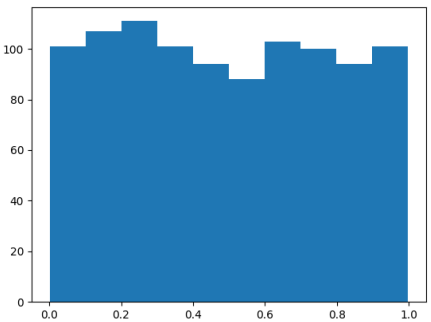
Właściwości dużych prób

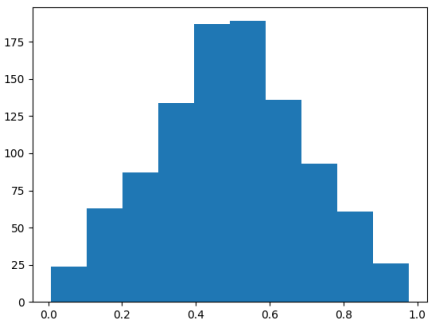
- Średnia średnich prób jest równa średniej populacji.
- Jeśli populacja ma rozkład normalny, średnie prób mają rozkład normalny.
- Jeśli populacja nie ma rozkładu normalnego, ale rozmiar prób jest większy niż 30, średnie prób tworzą rozkład zbliżony do normalnego.
-

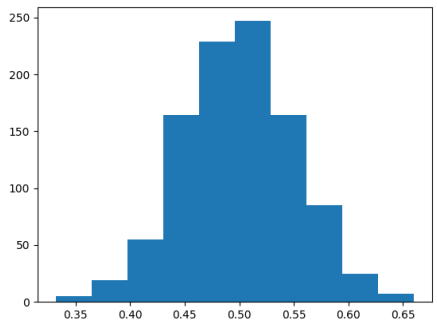
$$\textit{odchylenie standardowe prob} = \frac{\textit{odchylenie standardowe populacje}}{\sqrt{\textit{rozmiar proby}}}$$

Liczba 31

31 to podręcznikowa liczba w statystyce, ponieważ właśnie przy niej rozkład pr często zbiega się do rozkładu normalnego, zwłaszcza kiedy mierzymy średnią lub inne parametr prób. Kiedy próba liczy mniej niż 31 elementów, zamiast rozkładu normalnego trzeba użyć rozkładu t, który nie omawiamy w tam kursie (może na końcu...), im mniejszy jest rozmiar próby.







Przykład ze slajdów 36-37 z różnymi rozmiarami próby (1,2,31)

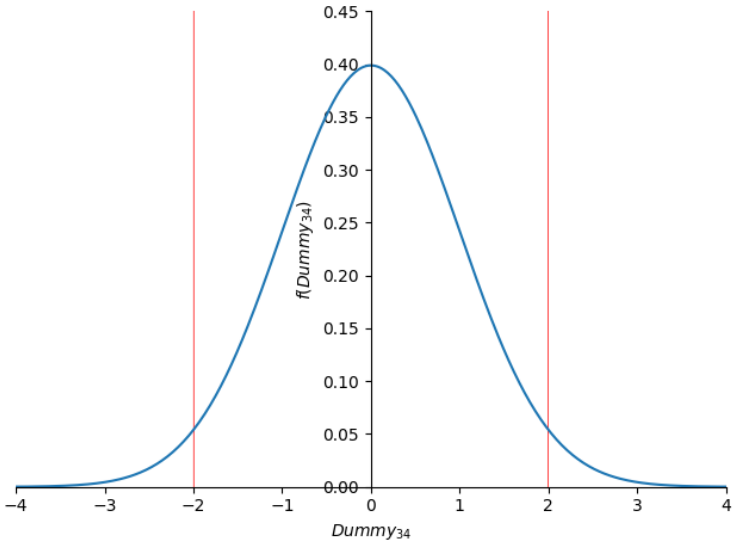
Jak duża próba jest wystarczająca?

Choć 31 to podręcznikowa liczba elementów, które musisz mieć w próbie, aby spełnić centralne twierdzenie graniczne i zaobserwować rozkład normalny, czasem to nie wystarcza. W niektórych przypadkach będziesz potrzebował jeszcze większej próby, na przykład kiedy bazowy rozkład jest asymetryczny lub multimodalny (co oznacza, że ma kilka szczytów, a nie tylko jeden przy średniej, nprz. waga, która zależna od płcie).

Dalej, mówiąc o przedziałach ufności i testowaniu hipotez, założymy, że mamy przynajmniej 31 elementów w próbie

Przedział ufności

Środkowy zakres w rozkładzie normalnym, dla którego prawdopodobieństwo wynosi konkretną liczbą procentów (najczęściej 95%)
(Porównaj ze slajdem 23)



Ten zakres można dobrać:

```
from scipy.stats import norm

mu = 0
sigma = 1

x = norm.cdf(2,mu,sigma) - norm.cdf(-2,mu,sigma)
print(x)  # 0.9544997361036416, [-2,2]
```

Przedział ufności

Lub obliczyć przy pomoc PPF (odwrotnej dystrybuanty)

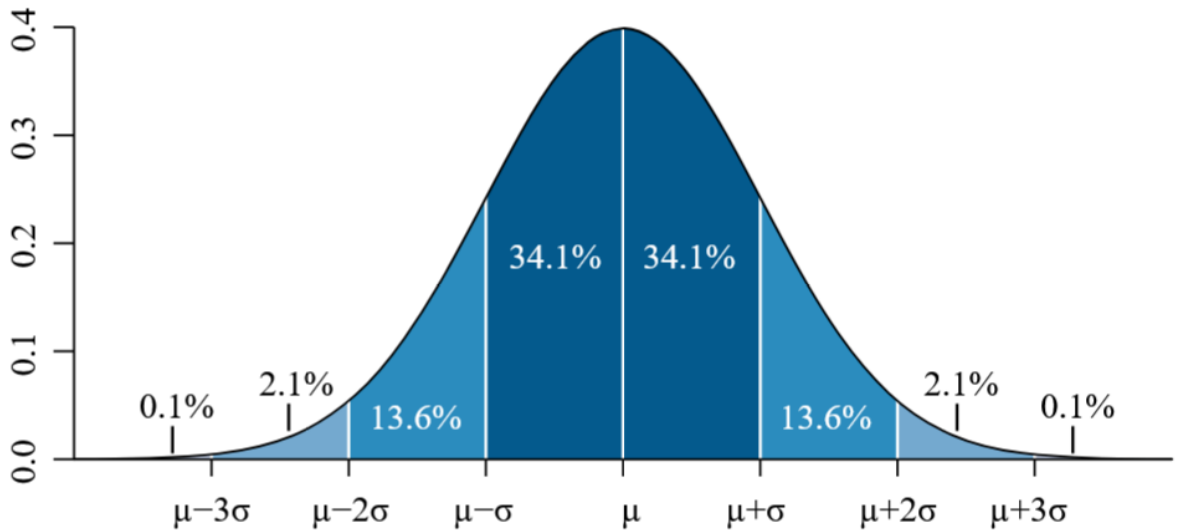
```
from scipy.stats import norm
import numpy as np

mu = 0
sigma = 1

print(norm.ppf(0.975, loc=mu,scale=sigma)) #1.959963984540054
print(norm.ppf(0.025, loc=mu,scale=sigma)) #-1.959963984540054
```

Przedział ufności

Lub obliczyć użyć tak zwanej *reguły trzech sigm*:

Dla danego rozkładu normalnego $\mathcal{N}(\mu, \sigma^2)$ oznacza że w przedziale $[\mu - 3\sigma, \mu + 3\sigma]$ znajduje się 99.7% wszystkich obserwacji. Oprócz tego, zachodzi w tym przypadku:

$$P(\mu - \sigma \leq X \leq \mu + \sigma) \approx 68.27\%$$
$$P(\mu - 2\sigma \leq X \leq \mu + 2\sigma) \approx 95.45\%$$
$$P(\mu - 3\sigma \leq X \leq \mu + 3\sigma) \approx 99.73\%$$



By Ainali - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=3141713

Przykład używania

Załóżmy taką sytuację: jesteśmy właścicielami firmy produkującej obuwie. Zastanawiamy się nad tym w jakich rozmiarach mamy produkować buty. Załóżmy, że po badaniach wnioskowaliśmy że średni rozmiar męskiego buta wynosi $\mu = 42$ a odchylenie standardowe $\sigma = 1.5$. Korzystając z Reguł Trzech Sigm produkując buty w rozmiarach od 37.5 do 46.5 w naszych butach będzie mogło chodzić 99.7% mężczyzn. To bardzo dobry wynik ale załóżmy, że nasza firma nie ma za dużo pieniędzy na wyprodukowanie partii butów, co wtedy? Być może lepszym wyjściem jest zrobienie butów w rozmiarach od 39 do 45. Wtedy w naszych butach będzie mogło chodzić 95.4% mężczyzn a zaoszczędzimy trochę na produkcji dużych numerów.
Dzięki rozkładowi normalnemu wiemy również których rozmiarów wyprodukować więcej – tych najbliżej średniej – od 40.5 do 43.5 prawie 70% butów.

Margines błędu

Dla rozkładu normalnego definiuje się jako

$$E = \pm z \frac{\sigma}{\sqrt{n}}.$$

gdzie

- z – to końcówki przedział ufności w standardowym rozkładzie normalnym (w zależności od wybranego poziomu ufności),
- σ – to odchylenie standardowe,
- n – to wielkość próbki

Wtedy, jeżeli $\overline{x}$ jest średnia z próby, to mówimy ze „średnie znaczenie badanej cechę w populacji jest $\overline{x} \pm E$ z prawdopodobieństwem [odpowiednim wziętemu przedziału ufności]%"

Uwaga! Tutaj chodzi o populacje, nie o probie!

z dla różnych przedziałów ufności

przedział ufności %	z	przedział ufności %	z
68	0.994457883210	99.9	3.290526731492
90	1.644853626951	99.99	3.890591886413
95	1.959963984540	99.999	4.417173413469
98	2.326347874041	99.9999	4.891638475699
99	2.575829303549	99.99999	5.326723886384
99.5	2.807033768344	99.999999	5.730728868236
99.7	2.967737925342	99.9999999	6.109410204869

Z Wikipedii, można sprawdzić w Pythonie:

```
print(norm.ppf((1-0.99999999)/2, loc=0,scale=1))#-6.10941020938345
```

Przykład: znowu pingwiny

Po obliczeniach dla pingwinów mamy

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

weight0 = np.array([3750, 3800, ...])
weight = np.array([round(i/100) for i in weight0])

mu = np.mean(weight)
sigma = np.sqrt(np.var(weight))

print(np.mean(weight)) #srednia - 37.019867549668874
print(np.sqrt(np.var(weight))) #odchylenie - 4.589752765240528
E =  sigma*norm.ppf(0.975, loc=0,scale=1)/np.sqrt(len(weight))
print(E)#0.732063762386783
```

Średnia waga pingwina w populacji jest 3.7 ± 0.073 kilo z prawdopodobieństwem 95% (waga w programie jest w deko).

Wartość istotności (Wartość p)

Wartość istotności lub *wartość p* – to prawdopodobieństwo, że wynik wystąpił przez przypadek. Wartość istotności jest porównywana z wcześniej ustalonym odcięciem (poziomem istotności) w celu ustalenia, czy test jest statystycznie istotny. Jeśli wartość istotności jest niższa niż poziom istotności (domyślnie 0.05), test jest oceniany jako statystycznie istotny.

Wartość istotności (Wartość p): przykład

Jak wynaleziono wartość p?

W 1925 roku matematyk Ronald Fisher był na przyjęciu. Jedna z jego koleżanek, Muriel Bristol, twierdziła, że potrafi wykryć, czy herbata została nalana do filiżanki przed mlekiem, po prostu przez spróbowanie napoju.

Zaintrygowany Ronald natychmiast postanowił przeprowadzić eksperyment.

Przygotował osiem filiżanek herbaty. Do czterech najpierw nalał mleko, a do pozostałych czterech najpierw herbatę.

Następnie wręczył je koleżance-koneserce i poprosił, żeby określiła kolejność napełniania każdej z nich. Co godne uwagi, poprawnie zidentyfikowała każdą z nich, a prawdopodobieństwo, że stanie się tak przypadkiem, wynosi 1 do 70, czyli $0,01428571$.

To oznacza że prawdopodobieństwo, że Muriel wskazała właściwe filiżanki zupełnie przypadkowo, wynosi 1,4%. Ten $1,4$ procenta nazywamy wartością p.

Hipotez zerowa i hipoteza alternatywna

Kiedy prowadzimy eksperyment, zawsze musimy uwzględnić możliwość, ze na wynik wpłynął słępy los.

Możemy zatem postawić *hipotezę zerową* (H_0), która zakłada, że interesująca nas zmienna nie miała wpływu na eksperyment, a ewentualne pozytywne wyniki są dziełem przypadku. *Hipoteza alternatywna* (H_1) postuluje, że interesująca zmienna (zwana *zmienną kontrolną*) powoduje pozytywny wynik.

Tradycyjnym progiem istotności statystycznej jest wartość p równa 5% lub mniej.

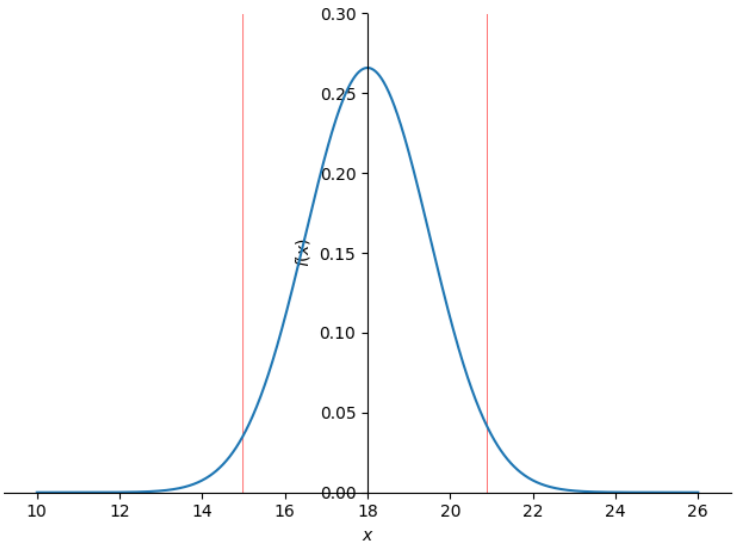
W przykładzie: Ponieważ $0,014 < 0,05$, uzasadnione jest odrzucenie hipotezy zerowej postulującej, że Muriel zgadywała na chybil-trafił. Możemy więc zacząć promować hipotez alternatywną przyjmującą, że Muriel miała szczególną zdolność wykrywania, czy najpierw nalano herbatę, czy mleko.

Testowanie hipotez: przykład

Jedną rzeczą, której zabrakło w naszym herbacianym przykładzie, jest to, że kiedy obliczamy wartość p, określamy prawdopodobieństwo zdarzenia o takiej częstości lub rzadszego. Omówimy to dokładniej w następnym przykładzie z rozkładem normalnym.

Z istniejących badań wynika, że średni czas powrotu do zdrowia po przeziębieniu wynosi 18 dni z odchyleniem standardowym równym $1,5$ dnia i że ma rozkład normalny.

Oznacza to, że istnieje około 95% szans, że zdrowienie potrwa $15 - 21$ dni.



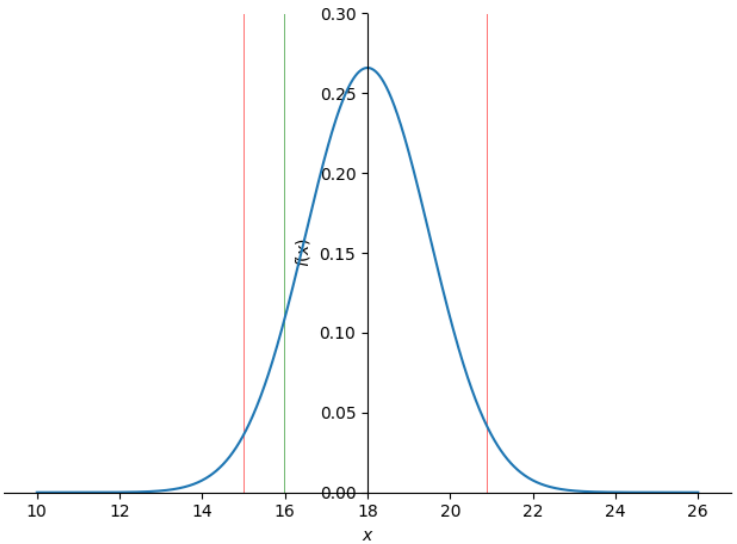
Kod

```
import matplotlib as plt
from sympy import *
from scipy.stats import norm
x = symbols('x')
def f(x, mu = 0, sigma = 1):
    return 1/(sigma*sqrt(2*pi))*exp(-(x-mu)**2/(2*sigma**2))

print(norm.ppf(0.975, loc=18,scale=1.5))#20.93994597681008
print(norm.ppf(0.025, loc=18,scale=1.5))#15.06005402318918

p1 = plot(f(x,18, 1.5),(x, 10,26),ylim=[0,0.3], show=False)
line1 = plot_implicit(Eq(x, 20.9),(x, 10,26),line_color='r',
                                                show=False)
line2 = plot_implicit(Eq(x, 15.0),(x, 10,26), line_color='r',
                                                show=False)
#line3 = plot_implicit(Eq(x, 16.0),(x, 10,26), line_color='g',
                                                show=False)
p1.extend(line1)
p1.extend(line2)
#p1.extend(line3)
p1.show()
```

Testowanie hipotez: przykład

Oznacza to, że istnieje około 95% szans, że zdrowienie potrwa $15 - 21$ dni. Z pozostałych 5% prawdopodobieństwa możemy zatem wywnioskować, że **istnieje** $2,5\%$ **szans, że ze zdrowienie zajmie więcej niż** 21 **dni oraz** $2,5\%$ **szans, że zajmie mniej niż** 15 **dni.**

Przypuśćmy teraz, że testowej grupie 40 osób podano nowe, eksperymentalne lekarstwo, a pacjenci ci wyzdrowieli średnio w ciągu 16 dni.



Czy lekarstwo miało na to jakiś wpływ? Jeśli się nad tym zastanowisz, dojdziesz do wniosku, że pytanie powinno brzmieć: czy lekarstwo ma statystycznie istotne wyniki? Czy też nie zadziałało, a 16-dniowy okres zdrowienia był przypadkiem w grupie testowej? Pierwsze pytanie określa naszą hipotezę alternatywną, a drugie zerową.

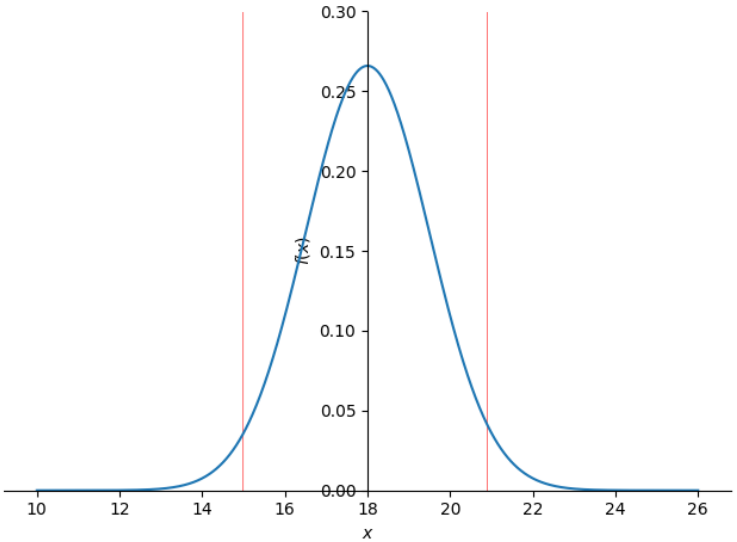
Istnieją dwa sposoby, żeby to policzyć: test jednostronny i test dwustronny.

Test dwustronny

Aby przeprowadzić test dwustronny, formułujemy hipotezę zerową i alternatywną w strukturze „równy” i „nierówny”.

W naszym teście lekarstwa powiemy, że w hipotezie zerowej średni czas zdrowienia wynosi 18 dni. Ale w hipotezie alternatywnej średni czas zdrowienia nie wynosi 18 dni za sprawą nowego medykamentu:
H_0 : średnia populacji $= 18$, $\quad H_1$: średnia populacji $\neq 18$

Uwaga: Nasza hipoteza alternatywna nie bada, czy lekarstwo skraca czas zdrowienia, ale czy **ma jakikolwiek wpływ**. Obejmuje to sprawdzenie, czy wydłużyło czas przeziębienia. Czy to zrozumiałe? Zapamiętaj to na przyszłość.

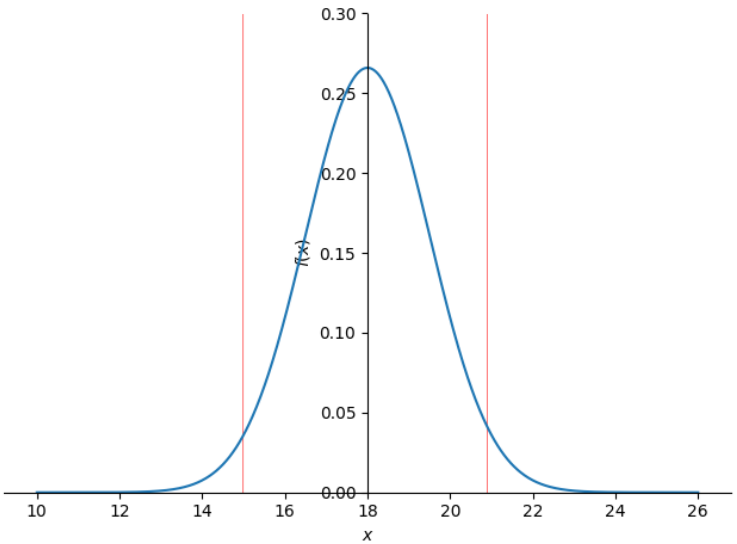


Rozciągamy próg istotności statystycznej naszej wartości p na dwa ogony"krzywej, nie tylko na jeden. Jeśli testujemy pod kątem istotności statystycznej 5%, dzielimy ją na pół i przypisujemy $2,5\%$ każdemu ogonowi. Jeśli średni czas zdrowienia po zażyciu lekarstwa przypada na którykolwiek z dwóch regionów, uznajemy test za udany i odrzucamy hipotezę zerową.

Test dwustronny

Aby przeprowadzić test dwustronny, formułujemy hipotezę zerową i alternatywną w strukturze „równy” i „nierówny”:
H_0 : średnia populacji $= 18$, H_1 : średnia populacji $\neq 18$

Uwaga: Nasza hipoteza alternatywna nie bada, czy lekarstwo skraca czas zdrowienia, ale czy **ma jakikolwiek wpływ**. Obejmuje to sprawdzenie, czy wydłużyło czas przeziębienia. Czy to zrozumiałe? Zapamiętaj to na przyszłość.



Rozciągamy próg istotności statystycznej naszej wartości p na dwa ogony"krzywej. Jeśli testujemy pod kątem istotności statystycznej 5%, dzielimy ją na pół i przypisujemy $2,5\%$ każdemu ogonowi. Jeśli średni czas zdrowienia po zażyciu lekarstwa przypada na którykolwiek z dwóch regionów, uznajemy test za udany i odrzucamy hipotezę zerową.

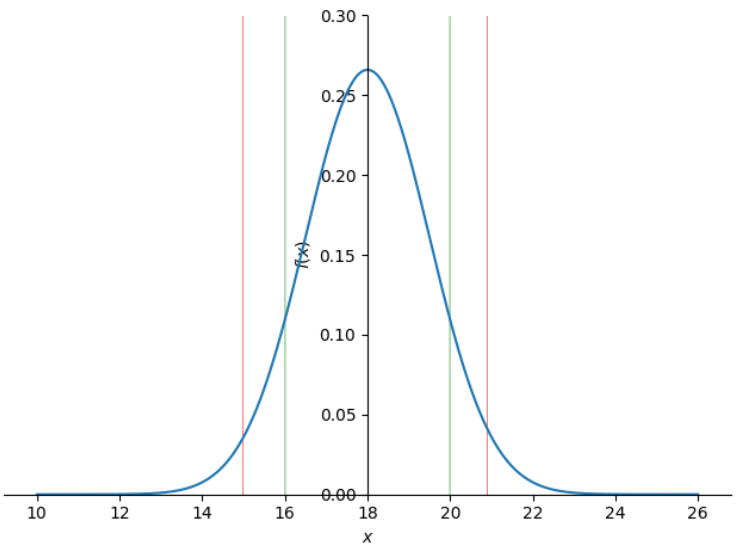
W naszym przypadku nie odrzucamy H_0.

Test dwustronny: inne sposób wnioskowania

H_0 : średnia populacji $= 18$, H_1 : średnia populacji $\neq 18$

Aby odrzucić hipotezę zerową, musimy wykazać, ze średnia próby pacjentów, które przyjmowali lekarstwo, zapewne nie jest przypadkowa. Ponieważ tradycyjnie za statystyczne istotą uważa się wartość p równą 0.05 lub mniej, użyjemy jej jako naszego progu.

```
2*print(norm.cdf(16, loc=18,scale=1.5)) + #0.18242243945173575 > 0.05
print(norm.cdf(16, loc=18,scale=1.5)+1.0-norm.cdf(20, loc=18,scale=1.5)
```



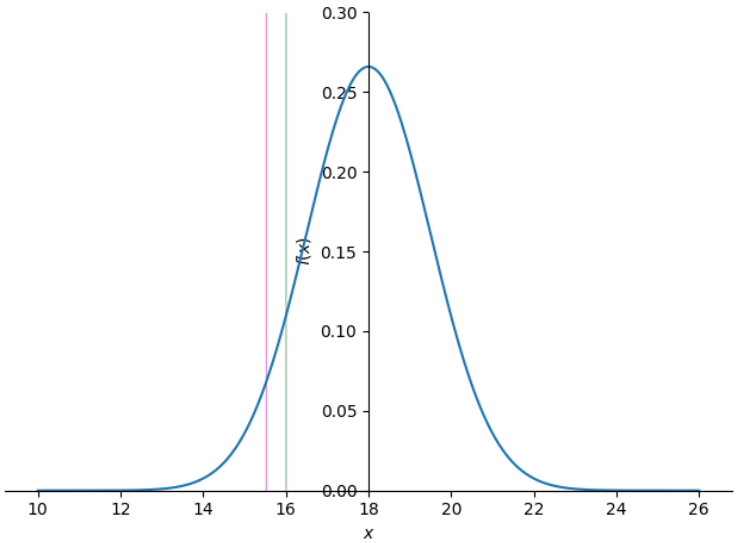
Nasza wartość p reprezentuje nie tylko obszar poniżej wartości 16, ale również równoważny symetryczny obszar pod prawym ogonem". Ponieważ 16 to 4 dni poniżej średniej, uwzględnimy również obszar ponad wartością 20, czyli 4 dni powyżej średniej.

Test jednostronny

Przystępując do testu jednostronnego, zwykle formułujemy hipotezę zerową i alternatywną przy użyciu nierówności. Stawiamy hipotezy wokół średniej populacji: H_0 : średnia populacji $= 18$, H_1 : średnia populacji < 18.

Aby odrzucić hipotezę zerową, musimy wykazać, ze średnia próby pacjentów, które przyjmowali lekarstwo, zapewne nie jest przypadkowa. Ponieważ tradycyjnie za statystyczne istotą uważa się wartość p równą 0.05 lub mniej, użyjemy jej jako naszego progu, **które badamy z jednej strony krzywej**.

```
print(norm.ppf(0.05, loc=18,scale=1.5))#15.53271955957279
```




```
line_color = 'm'
```

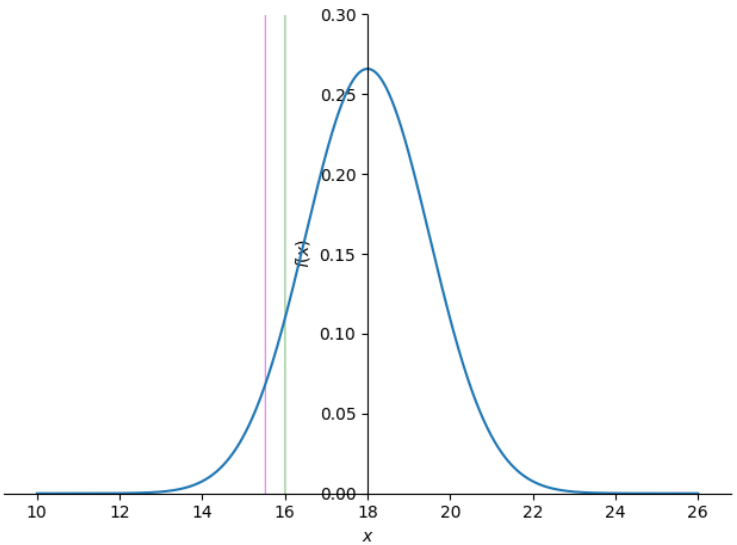
Jeśli zatem w próbnej grupie osiągniemy średni czas zdrowienia wynoszący 15,53 lub mniej. to skuteczność naszego leku będzie statystycznie istotna. Jednak średni czas zdrowienia próbie w rzeczywistości wynosi 16 dni i nie mieści się w strefie, która pozwalałaby odrzucić hipotezę zerową Zatem test istotności statystycznej nie powiódł się.

Test jednostronny: inne sposób wnioskowania

Przystępując do testu jednostronnego, zwykle formułujemy hipotezę zerową i alternatywną przy użyciu nierówności. Stawiamy hipotezy wokół średniej populacji: H_0 : średnia populacji $= 18$, H_1 : średnia populacji < 18

Aby odrzucić hipotezę zerową, musimy wykazać, ze średnia próby pacjentów, które przyjmowali lekarstwo, zapewne nie jest przypadkowa. Ponieważ tradycyjnie za statystyczne istotą uważa się wartość p równą 0.05 lub mniej, użyjemy jej jako naszego progu.

```
print(norm.cdf(16, loc=18,scale=1.5))#0.09121121972586788 > 0.05
```



Ponieważ wartość p (prawdopodobieństwo z lewej strony od 16) równa 0.0912 jest większa niż próg istotności statystycznej równy $0,05$, nie uważamy testów lekarstwa za udane i nie odrzucamy hipotezy zerowej.

Test jednostronny vs. test dwustronny

Test jednostronny sprawdza „wzrost" lub „spadek" parametru, natomiast test dwustronny sprawdza „zmianę" (może to być wzrost lub spadek) parametru.

Wartość p jest dwukrotnie wyższą dla testu dwustronnego. To oznacza, ze dla tego samego progu istotności istnieją takie wyniki testu, ze odrzucamy H_0 w teście dwustronnym i akceptujemy w jednostronnym:
W teście jednostronnym akceptujemy $H_1 < \mu$ ($H_1 > \mu$), w teście dwustronnym zostaje się $H_0 = \mu$. Oznacza to, że test dwustronny utrudnia odrzucenie hipotezy zerowej i wymaga mocniejszych dowodów.
Przykład: jeżeli w probie z lekiem wyzdrowienie odbywało się za 15.5 dni, to

```
print(norm.cdf(15.5, loc=18,scale=1.5))#0.0477903522728147<0.05
print(norm.cdf(15.5, loc=18,scale=1.5)*2)#0.09558070456294>0.05
```

W teście jednostronnym wyjaśniamy, ze lek zmniejsza czas na wyzdrowienia, w teście dwustronnym wnioskujemy ze lek nie ma wpływu na czas zdrowienia.

Czy lek zmniejsza czas zdrowienia? – raczej tak.
Czy ma istotny wpływ? – raczej nie.

To jest dlatego, ze test dwustronny tez uwzględnia przypadek ze lek wydłużył czas wyzdrowienia.

Rozkład Studenta (rozkład t Studenta, rozkład t)

Rozkład o gęstości:

$$, f(t) \;=\; \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\pi\ \nu}\ \Gamma\left(\frac{\nu}{2}\right)} \left(\ 1+\frac{t^2}{\nu}\ \right)^{-(\nu+1)/2} ,$$

gdzie $\nu = n - 1$ – ilość stopni swobody, n – ilość próby. Używa się dla prób $n \leq 30$ zamiast rozkładu normalnego.

Srednia: 0

Wariacja: $\frac{\nu}{\nu - 2}$

Im mniejszy rozmiar próby, tym grubsze „ogony"w rozkładzie 1. Interesujące jest to, że kiedy zbliżamy się do liczby 31 elementów, rozkład T staje się praktycznie nieodróżnialny od rozkładu normalnego, co dobrze odzwierciedla idee stojące za centralnym twierdzeniem granicznym.