KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt *Fizyka wobec wyzwań XXI wieku* współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

Spis treści

Pojęcia wstępne

Zmienna losowa

Intuicyjnie
: Zmienna losowa to taka zmienna, która przyjmuje pewną konkretną wartość w wyniku

przeprowadzenia pomiaru lub eksperymentu. Na przykład zmienną losową jest liczba oczek, która wypada na kostce do gry. Dopóki trzymamy kostkę w ręce to nie wiemy jaką wartość ta zmienna przyjmie, konkretną wartość zmienna ta przyjmuje po rzucie. Jednak w kolejnych rzutach nie wiemy jakie będą jej wartości.

Bardziej formalnie

Zmienna losowa to funkcja przypisująca zdarzeniom elementarnym liczby.

Zupełnie formalnie

Do formalnej definicji potrzebne jest nam pojęcie przestrzeni probabilistycznej: Przestrzeń probabilistyczna to układ trzech elementów (Ω, F, P), gdzie:

- Ω jest pewnym zbiorem, zwanym przestrzenią zdarzeń elementarnych,
- F jest σ-ciałem podzbiorów zbioru Ω. Elementy tego σ-ciała nazywane są zdarzeniami losowymi,
- $P : F \to [0, 1]$ jest miarą probabilistyczną, tzn.
  - $P(A) \geq 0$ dla każdego $A \in F$,
  - $P(\Omega) = 1$,
  - jeżeli zbiory $A_1, A_2, \ldots, A_n \in F$ są parami rozłączne, to $P\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} P(A_i)$.

Niech (Ω, F, P) będzie przestrzenią probabilistyczną. **Zmienną losową** nazywamy dowolna funkcję X, określoną na przestrzeni zdarzeń elementarnych Ω, o wartościach ze zbioru liczb rzeczywistych i mierzalną względem przestrzeni (Ω, F, P). Zmienna losowa jest ciągła jeśli jej zbiór wartości jest ciągły, lub dyskretna, jeśli jej zbiór wartości jest dyskretny.

Rozkład prawdopodobieństwa

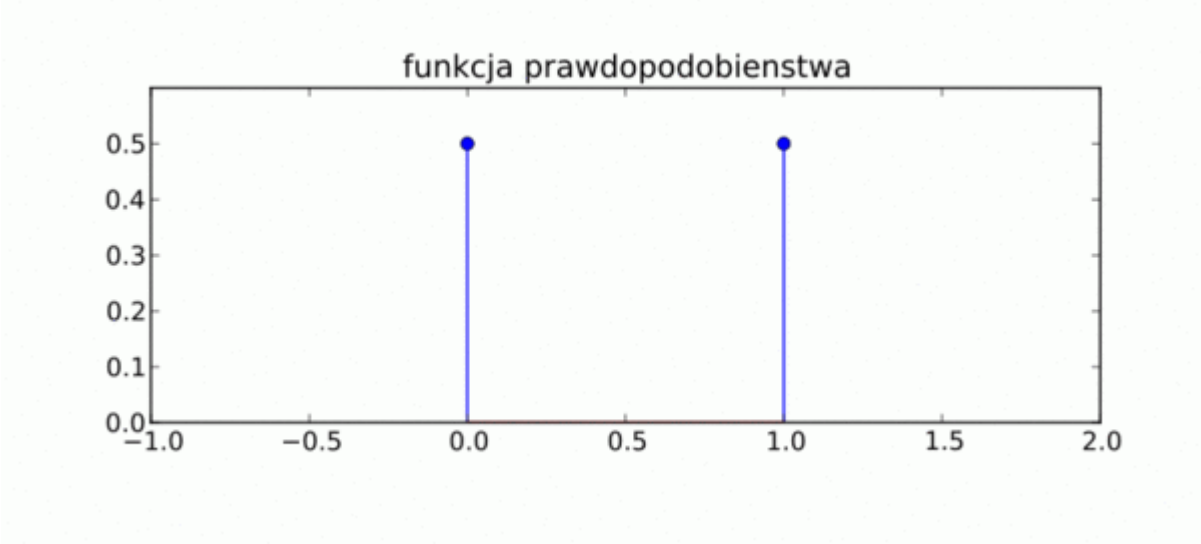
Rozkład prawdopodobieństwa — (rozkład zmiennej losowej) miara probabilistyczna określona na σ-ciele podzbiorów zbioru wartości zmiennej losowej, pozwalająca przypisywać prawdopodobieństwa zbiorom wartości tej zmiennej, odpowiadającym zdarzeniom losowym.

Dla zmiennej losowej dyskretnej możemy wprowadzić pojęcie **funkcji prawdopodobieństwa**: $P(X = x_i) = p_i$

Przykład: Dla rzutów monetą funkcję prawdopodobieństwa można zobrazować jako tabelkę:

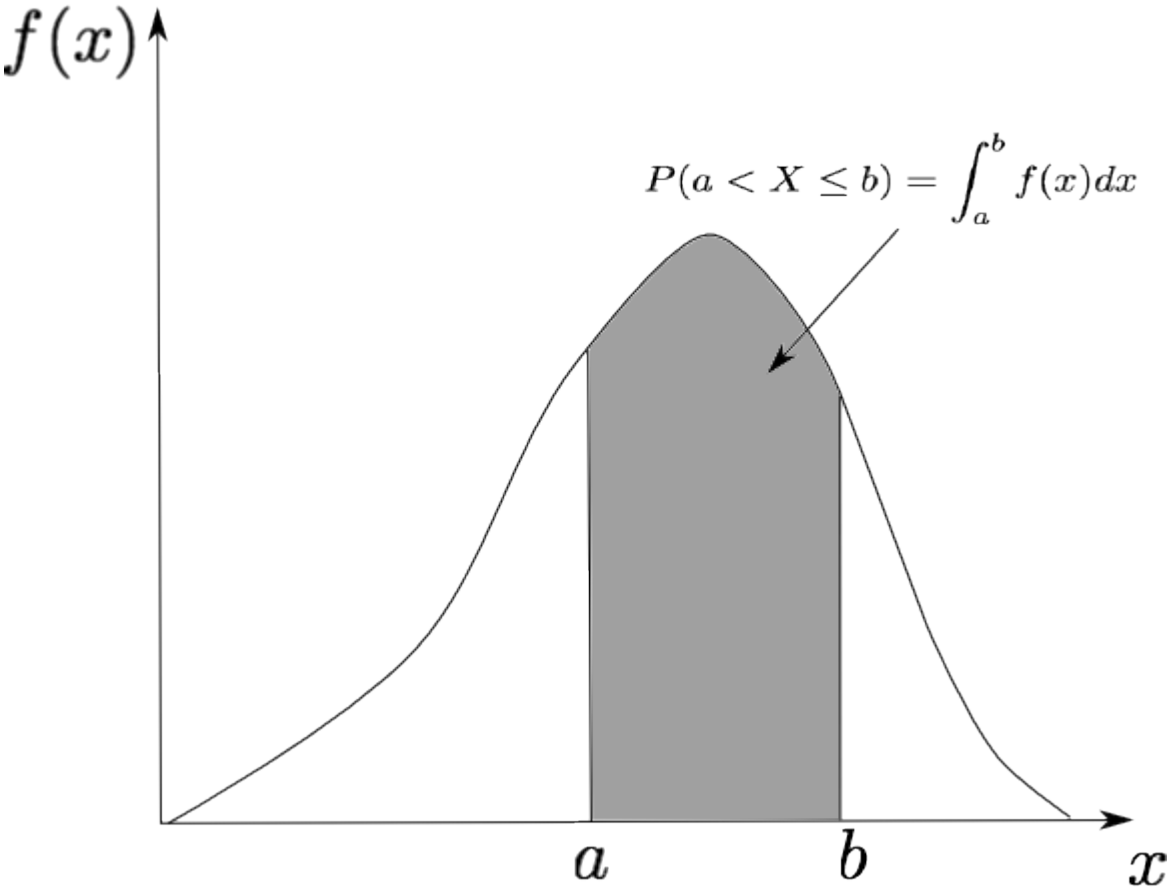
	Zdarzenie:	**orzeł**	**reszka**
zmienna losowa	x_i	0	1
prawdopodobieństwo	p_i	1/2	1/2

Można też przedstawić ją na wykresie:



Dla zmiennej losowej ciągłej zamiast funkcji prawdopodobieństwa wprowadzamy **funkcję gęstości prawdopodobieństwa**. Jest to funkcja określona na zbiorze liczb rzeczywistych posiadająca następujące własności:

- jest nieujemna: $f(x) \geq 0$
- prawdopodobieństwo tego, że zmienna losowa przyjmie wartość z przedziału $(a, b]$:
  $\int_a^b f(x)dx = P(a < X \leq b)$



- Prawdopodobieństwo tego, że zmienna losowa przyjmie dowolną wartość:
$$\int_{-\infty}^{+\infty} f(x)dx = P(-\infty < X \leq +\infty) = 1$$

Dystrybuanta

Dystrybuantą zmiennej losowej X nazywamy funkcję $F(x)$ określoną na zbiorze liczb rzeczywistych jako: $F(x) = P(X \leq x)$

czyli wartość dystrybuanty w punkcie x jest prawdopodobieństwem, że zmienna losowa przyjmie wartość nie większą niż x.

- Dla zmiennej dyskretnej:

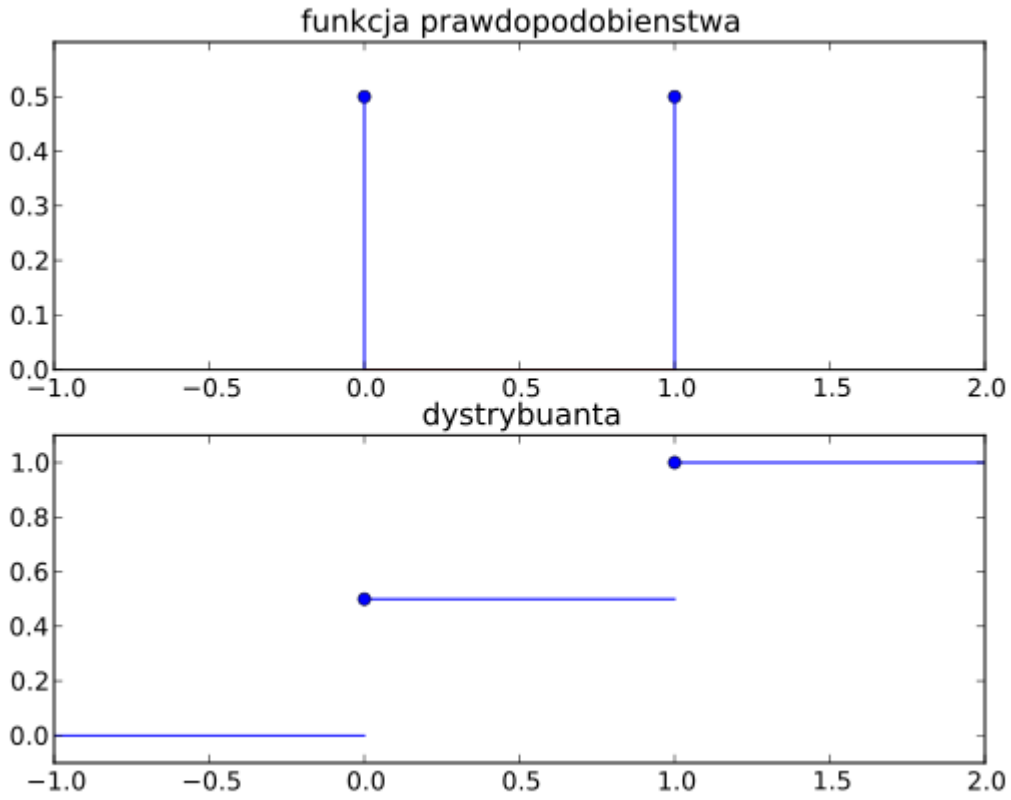
$$F(x) = P(X \leq x) = \sum_{x_i < x} P(X = x_i) = \sum_{x_i < x} p_i$$

- Dla zmiennej losowej ciągłej z funkcją gęstości prawdopodobieństwa f:

$$F(x) = P(X \leq x) = \int_{-\infty}^{x} f(s)ds$$

Zauważmy ścisły związek dystrybuanty z funkcją prawdopodobieństwa w przypadku dyskretnym i funkcją gęstości prawdopodobieństwa w przypadku ciągłym.

Przykład z monetą:



Momenty

Dla zmiennej losowej można obliczyć momenty zwykłe lub centralne.

Moment zwykły
: rzędu k zmiennej losowej X to wartość oczekiwana k-tej potęgi tej zmiennej. Oblicza się go tak:

- dla dyskretnej zmiennej losowej

$$m_k = E(X^k) = \sum_i x_i^k p_i$$

tu p_i oznacza prawdopodobieństwo, że $X = x_i$

- dla ciągłej zmiennej losowej

$$m_k = E(X^k) = \int_{-\infty}^{\infty} x^k f(x) dx$$

tu $f(x)$ jest funkcją gęstości prawdopodobieństwa.

Zauważmy, że pierwszy moment zwykły to średnia.

Moment centralny
rzędu k zmiennej losowej X to wartość oczekiwana k-tej potęgi funkcji $[x_i - E(x_i)]$. Zatem możemy go obliczyć tak:

- dla dyskretnej zmiennej losowej

$$\mu_k = E\left([X - E(X)]^k\right) = \sum_i [x_i - E(X)]^k p_i$$

- dla ciągłej zmiennej losowej

$$\mu_k = E\left([X - E(X)]^k\right) = \int_{-\infty}^{\infty} [X - E(X)]^k f(x)dx$$

Uwaga:

- Drugi moment centralny ma swoją nazwę. Jest to **wariancja**, często oznaczana przez σ^2.
- Trzeci moment centralny przydaje się do badania symetrii rozkładu. Przyjmuje on wartość 0 dla zmiennych o rozkładzie symetrycznym, wartości ujemne dla zmiennych o rozkładzie wydłużonym w lewą stronę i wartości dodatnie dla zmiennych o rozkładzie wydłużonym w prawą stronę.
- Czwarty moment centralny jest przydatny do konstrukcji miary spłaszczenia rozkładu zmiennej losowej **kurtozy**. Definiuje się ją następująco:

$$\mathrm{Kurt} = \frac{\mu_4}{\mu_2^4} - 3$$

Rozkłady prawdopodobieństwa można podzielić ze względu na wartość kurtozy na rozkłady:

- mezokurtyczne — wartość kurtozy wynosi 0, spłaszczenie rozkładu jest podobne do spłaszczenia rozkładu normalnego (dla którego kurtoza wynosi dokładnie 0)
- leptokurtyczne — kurtoza jest dodatnia, wartości cechy bardziej skoncentrowane niż przy rozkładzie normalnym
- platokurtyczne — kurtoza jest ujemna, wartości cechy mniej skoncentrowane niż przy rozkładzie normalnym

Kwantyle

Kwantylem rzędu *p*, gdzie $0 \leqslant p \leqslant 1$, w rozkładzie empirycznym P_X zmiennej losowej X nazywamy

każdą liczbę x_p, dla której spełnione są nierówności

$$P_X((-\infty, x_p]) \geqslant p$$

oraz

$$P_X([x_p, \infty)) \geqslant 1 - p.$$

W szczególności, kwantylem rzędu *p* jest taka wartość x_p zmiennej losowej, że wartości mniejsze lub równe od x_p są przyjmowane z prawdopodobieństwem co najmniej *p*, zaś wartości większe lub równe od x_p są przyjmowane z prawdopodobieństwem co najmniej $1-p$.

Możemy zatem myśleć o obliczaniu kwantyla tak:

- zaobserwowaliśmy *n* wartości zmiennej losowej,
- zapiszemy te wartości na liście,
- następnie listę tę sortujemy w porządku rosnącym,
- kwantylem rzędu *p* jest albo element znajdujący się na liście w pozycji *np* jeśli *np* jest całkowite albo średnią z elementów położonych najbliżej *np* w przeciwnym wypadku.

Nazwy poszczególnych kwantyli

Kwantyl rzędu 1/2 to inaczej mediana (ściślej zależy to od definicji mediany, przy jej obliczaniu z próbki o parzystej liczbie elementów często stosuje się średnią arytmetyczną dwóch środkowych elementów, szczegóły są w artykule).
Kwantyle rzędu 1/4, 2/4, 3/4 są inaczej nazywane kwartylami.
Kwantyle rzędu 1/5, 2/5, 3/5, 4/5 to inaczej kwintyle.
Kwantyle rzędu 1/10, 2/10, ..., 9/10 to inaczej decyle.
Kwantyle rzędu 1/100, 2/100, ..., 99/100 to inaczej percentyle.

Pseudolosowość

W obliczeniach numerycznych często korzystamy z liczb „losowych”. Tak naprawdę sekwencje liczb, których używamy są jedynie pseudolosowe, tzn. są wytwarzane w deterministyczny (algorytmiczny) sposób, ale sekwencja generowanych wartości ma pewne cechy losowości. W idealnym przypadku cechą tą jest nieprzewidywalność: na podstawie obserwacji dotychczasowych wartości sekwencji niemożliwe jest podanie kolejnych wartości. W praktyce oznacza to nie możność ustalenie ziarna lub stanu wewnętrznego generatora na podstawie obserwacji dowolnie długiego ciągu wygenerowanych bitów. Stan generatora przechowywany jest na zmiennych o skończonej precyzji. Jeśli stan jest przechowywany na n bitach to górną granicą na długość unikalnego ciągu liczb jest 2^n. Zazwyczaj unikalna sekwencja jest jednak krótsza. Wynika stąd, że generatory liczb pseudolosowych mają okres, po którym sekwencja liczb powtarza się. Podstawowe generatory liczb pseudolosowych wytwarzają liczby podlegające rozkładowi jednostajnemu (płaskiemu).

Istnieją generatory „prawdziwych” liczb losowych, które są osobnymi urządzeniami. Liczy przez nie

dostarczane są pewnymi parametrami jakichś procesów fizycznych, przebiegających w tych urządzeniach (zależnie od producenta). Do naszych zastosowań w zupełności wystarczą liczby generowane przez algorytmy komputerowe.

Proste generatory liczb pseudolosowych można otrzymać jako funkcje używające operacji dzielenia modulo (generatory kongruencyjne):

$$x_{n+1} = f(x_n, x_{n-1}, ..., x_{n-k}) \mod M.$$

Zakładamy, że argumentami funkcji f są liczby całkowite ze zbioru $0, 1, ..., M-1$. Dla ustalenia uwagi mogą to być generatory liniowe typu:

$$x_{n+1} = (ax_n + c) \mod M$$

Przykład generatora liniowego

Dla przykładu zaimplementujmy jeden taki generator:

```
# -*- coding: utf-8 -*-

import numpy as np
import pylab as py
def gen(x,N=1):
    '''Przykład generatora liniowego.
       funkcja zwraca liczby pseudolosowe z przedziału [0,1)
       x - ziarno,
       N - ile liczb zwrócić'''
    m=8191
    a=101
    c=1731
    y=np.zeros(shape=(N,))
    for i in range(N):
        x=(a*x+c) % m
        y[i]= x/m
    return y
x=gen(23,100000)
py.hist(x)
py.show()
```

Liczby `m, a, c` są parametrami generatora. Liczba `x`, od której startuje generator nazywana jest ziarnem generatora (ang. *seed*). Funkcja `py.hist()` zlicza i wyrysowuje histogram (czy wiemy co to jest histogram?).

- Wywołaj funkcję `gen` dla tego samego ziarna kilka razy zaobserwuj powtarzalność wartości.

W module `numpy` mamy do dyspozycji generator liczb pseudolosowych oparty na algorytmie [Mersenne Twister]

```
# -*- coding: utf-8 -*-
"""
kod demonstrujacy pseudolosowosc
"""
import numpy as np
seed=3
np.random.seed(seed)
x=np.random.random(5)

np.random.seed(seed)
y=np.random.random(5)
print('x:', x)
print('y:', y)
```

- Zaobserwuj powtarzalność wartości.
- Wygeneruj i wykreśl 200 kolejnych liczb pseudolosowych.

```
# -*- coding: utf-8 -*-
import numpy as np
import pylab as py
seed=3
np.random.seed(seed)
x=np.random.random(200)

py.plot(x,'.')
py.show()
```

Testy jakości

Dla wyżej wymienionych generatorów, proszę wykonać następujące testy:

- zobrazować rozkład prawdopodobieństwa (narysować histogram);
- test korelacji na rysunku: sporządzić wykres gdzie na jednej osi są wartości x_n na drugiej zaś wartości x_{n+1}.

```
# -*- coding: utf-8 -*-

import numpy as np
import pylab as py
def gen(x,N):
```

```
    m=8765132137 #8191
    a=42347#101
    c=1731
    y=np.zeros(shape=(N,))
    for i in range(N):
        x=(a*x+c) % m
        y[i] = x/m
    return y


N=50000
x=gen(23,N)

py.subplot(2,1,1)
py.hist(x,30,normed = True)
py.subplot(2,1,2)
py.plot(x[:-1],x[1:],'.')
py.xlabel('x_n')
py.ylabel('x_{n+1}')
py.show()
```

Bardziej wyczerpujące baterie testów jakości generatorów można znaleźć na przykład w:

- http://csrc.nist.gov/groups/ST/toolkit/rng/index.html
- http://www.phy.duke.edu/~rgb/General/rand_rate.php

Zmienne losowe w module `scipy.stats`

Wstęp

Zmienne losowe w module `scipy.stats` są zaimplementowane jako dwie klasy: `stats.rv_continuous` (bazuje na niej ponad 80 typów rozkładów ciągłych zmiennych losowych) i `stats.rv_discrete` (bazuje na niej 10 typów rozkładów dyskretnych zmiennych losowych). Aktualną listę dostępnych rozkładów można uzyskać np. przy pomocy następującego skryptu:

```
from scipy import stats
dc= [d for d in dir(stats) if isinstance(getattr(stats,d),
stats.rv_continuous)]
print(dc)
```

Przykładowo: zmienne losowe o rozkładzie normalnym są reprezentowane przez klasę `norm`.

Funkcje przydatne do pracy ze zmiennymi losowymi dostępne są jako metody

Obiekty reprezentujące zmienne losowe podlegające konkretnym rozkładom posiadają następujące

metody:

- `rvs`: **generator zmiennych losowych** danego typu;
- `cdf`: **dystrybuanta** (ang. Cumulative Distribution Function);
- `ppf`: **funkcja odwrotna do dystrybuanty** (ang. Percent Point Function (Inverse of CDF));
- `stats`: zwraca **momenty centralne rozkładu** (średnią, wariancję, skośność i kurtozę);
- `moment`: zwraca niecentralne momenty rozkładu.
- `pdf`: **funkcja gęstości prawdopodobieństwa** (ang. Probability Density Function);
- `pmf` dla rozkładów dyskretnych `pdf` jest zamienione na **funkcję prawdopodobieństwa** (ang. probability mass function; *masa* przez analogię z gęstością masy i masą punktową);
- `sf`: funkcja przeżycia (ang. Survival Function) (1−CDF);
- `isf`: funkcja odwrotna do funkcji przetrwania (ang. Inverse Survival Function (Inverse of SF));

Dla przykładu zobaczmy działanie tych metod dla zmiennych losowych z rozkładu normalnego:

```
import scipy.stats as st
import pylab as py
import numpy as np

# generowanie 10000 liczb z rozkładu normalnego
x = st.norm.rvs(size=10000)
py.subplot(2,2,4)
py.hist(x)
py.title('Histogram')
py.xlabel(u'wartość zmiennej')
py.ylabel(u'ilość zliczeń')

# wykreślenie gęstości prawdopodobieństwa
os_x = np.arange(-3,3,0.01)
y=st.norm.pdf(os_x)
py.subplot(2,2,3)
py.plot(os_x,y)
py.title(u'rozkład gęstości prawdopodobieństwa: pdf')
py.xlabel(u'wartość zmiennej')
py.ylabel(u'gęstość prawdopodobieństwa')

# wykreślenie dystrybuanty
z= st.norm.cdf(os_x)
py.subplot(2,2,1)
py.plot(os_x,z)
py.title(u'Dystrybuanta: cdf')

py.xlabel(u'wartość zmiennej')
py.ylabel(u'prawdopodobieństwo')

# wykreślenie funkcji odwrotnej do dystrybuanty
```

```
os_p = np.arange(,1,0.01)
f=st.norm.ppf(os_p)
py.subplot(2,2,2)
py.plot(os_p,f)
py.title(u'funkcja odwrotna do dystrybuanty: ppf')
py.ylabel(u'wartość zmiennej')
py.xlabel(u'prawdopodobieństwo')
py.show()
```

Zmienne losowe mogą być używane na jeden z dwóch sposobów: można podawać wszystkie parametry opisujące rozkład w każdym wywołaniu metody, albo wytworzyć obiekt reprezentujący rozkład o konkretnych parametrach (w dokumentacji `scipy` jest to nazywane zamrażaniem parametrów rozkładu, ang. *freezing*) Jako przykład zobaczmy funkcję odwrotną do dystrybuanty rozkładu normalnego N(2, 9):

```
>>> import scipy.stats as st
>>> print(st.norm.ppf(0.05, loc=2, scale=3))
-2.93456088085
>>> my_norm = st.norm(loc=2,scale=3)
>>> print(my_norm.ppf(0.05))
-2.93456088085
```

Pomoc do poszczególnych dystrybucji

Pełną dokumnetację każdej dystrybucji mamy dostępną w postaci docstringów. Dla przykładu:

```
help(st.nct)
```

wypisuje pełną informację z docstring o niecentralnym rozkładzie *t*. Podstawowe informacje można uzyskać wypisując treść pola `extradoc`

```
>>> print(st.nct.extradoc)
Non-central Student T distribution
df**(df/2) * gamma(df+1)
nct.pdf(x,df,nc) = --------------------------------------------------
2**df*exp(nc**2/2)*(df+x**2)**(df/2) * gamma(df/2)
for df > 0, nc > 0.
```

Generowanie zmiennych losowych podlegającym różnym rozkładom prawdopodobieństwa

Podstawowe typy generatorów liczb losowych wytwarzają liczby losowe podlegające rozkładowi jednostajnemu. Zastanowimy sie teraz jak mając zmienne losowe z rozkładu jednostajnego wytworzyć zmienne losowe o innych rozkładach.

Rozkład dwumianowy

Zmienna losowa, która zlicza liczbę sukcesów k w n próbach, gdzie p jest prawdopodobieństwem sukcesu w pojedynczej próbie, podlega rozkładowi dwumianowemu [więcej o rozkładzie dwumianowym]:

$$P_n(k) = \left(\begin{array}{c} n \\ k \end{array} \right) p^k q^{n-k} = \frac{n!}{k!(n-k)!} p^k q^{n-k}$$

Jak z rozkładu jednostajnego wytworzyć zmienne losowe o rozkładzie dwumianowym?

- Wykonać symulację!

```
# -*- coding: utf-8 -*-
import numpy as np
import pylab as py


def gen_binom(n,p,N):
        '''funkcja zwracająca zmienną losową z rozkładu dwumianowego:
        ilość sukcesów w n próbach przy prawdopodobieństwie sukcesu p
        parametry:
                n: ilość prób
                p: prawdopodobieństwo sukcesu
                N: ilość liczb do generowania
        funkcja zwraca:
                N liczb będącymi ilościami sukcesów
        '''
        k = np.zeros(N)
        # w pętli wytwarzam pojedyncze zmienne losowe
        # losuję n liczb z rozkładu jednostajnego [0,1) i jako sukcesy
        #traktuję wylosowanie liczby nie większej niż p
        for i in range(N):
                r = np.random.random(size=n)
                k[i]=np.sum(r<=p)
        return k

# kod testujący
n=20
N=100000
p=0.8
x = gen_binom(n,p,N)
bins =range(22) #granice binów
py.hist(x,bins)
py.show()
```

Zadanie

Z oszacowań agencji wynika, że średnio 2 z 3 reklam spotyka się z pozytywnym odzewem. Akcja marketingowa obejmuje 12 reklam. Niech X oznacza liczbę reklam skutecznych. Czy X podlega rozkładowi dwumianowemu? Jakie jest prawdopodobieństwo, że 10 reklam będzie skutecznych? Prawdopodobieństwo obliczyć ze wzoru oraz korzystając z symulacji.

Wskazówka informacje o zmiennych z rozkładu dwumianowego (`binom`) można uzyskać tak:

```
# >>> print(st.binom.extradoc)
#   Binomial distribution
#
#   Counts the number of successes in *n* independent
#   trials when the probability of success each time is *pr*.
#
#   binom.pmf(k,n,p) = choose(n,k)*p**k*(1-p)**(n-k)
#   for k in {0,1,...,n}
```

Odp: $p = 0{,}127$

Rozwiązanie:

```
# -*- coding: utf-8 -*-
import numpy as np
import pylab as py
import scipy.stats as st
import numpy.random as rnd

p=2./3 #prawdopodobieństwo sukcesu w jednej próbie
n=12   # ilość prób w jednej akcji reklamowej

# ze wzoru

# >>> print(st.binom.extradoc)
#   Binomial distribution
#
#   Counts the number of successes in *n* independent
#   trials when the probability of success each time is *pr*.
#
#   binom.pmf(k,n,p) = choose(n,k)*p**k*(1-p)**(n-k)
#   for k in {0,1,...,n}
p_wz = st.binom.pmf(10,n,p)
print(p_wz)
# z symulacji
N_powt =100000 # ilość powtórzeń symulacji
k = np.zeros(N_powt)
```

```
for i in range(N_powt):
        r = rnd.random(size=n) # symulowana akcja reklamowa
        k[i]=np.sum(r<=p)        # liczba sukcesów
p_sym = sum(k==10)/N_powt
print(p_sym)
```

Zadanie

Na egzaminie testowym jest 30 pytań. Na każde pytanie są podane cztery możliwe odpowiedzi, z czego tylko jedna jest prawdziwa. Za prawidłową odpowiedź student otrzymuje 1 punkt a za fałszywą −0,5 punktu. Próg zaliczenia wynosi 15 punktów. Oblicz prawdopodobieństwo, że udzielając czysto losowych odpowiedzi student zaliczy egzamin.

Rozwiązanie:

Liczba prawidłowych odpowiedzi w teście może być potraktowana jako liczba sukcesów k w n = 30 próbach. Prawdopodobieństwo sukcesu pojedynczego zdarzenia (prawidłowa odpowiedź na jedno pytanie) wynosi p = 1/4. Ze względu na obecność ujemnych punktów za fałszywe odpowiedzi zmienia się efektywna ilość punktów wymaganych do zaliczenia zgodnie z równaniem: n-0.5*(30-n) = 15 co daje n = 20 Zatem problem redukuje się do wyznaczenia prawdopodobieństwa co najmniej 20 sukcesów w 30 próbach, przy czym prawdopodobieństwo sukcesu w 1 próbie wynosi p=1/4:

```
# -*- coding: utf-8 -*-
import numpy as np
import scipy.stats as st

p = 1./4 #prawdopodobieństwo sukcesu w jednej próbie
n = 30   # ilość prób w jednej akcji reklamowej
k_kryt = 20
# ze wzoru
p_wz = np.sum(st.binom.pmf(range(k_kryt,n+1),n,p))
print(p_wz)

# z symulacji
N_powt =10000000 # liczba powtórzeń symulacji
r = np.random.random(size=(n,N_powt)) # symulowany test
k = np.sum(r<=p,)    # ilość sukcesów
p_sym = sum(k>=20)/N_powt
print(p_sym)
```

Rozkład Poissona

$P(k) = \frac{\mu^k e^{-\mu}}{k!}$ Rozkładowi Poissona podlegają zmienne losowe zliczające w jednostce czasu ilość zdarzeń o niskim prawdopodobieństwie zajścia. Np. ilość rozpadów promieniotwórczych na

jednostkę czasu [więcej o rozkładzie Poissona]

Przykład

Lekarz pełniący dyżur w szpitalu jest wzywany do pacjentów średnio 3 razy w ciągu nocy. Załóżmy, że liczba wezwań na noc podlega rozkładowi Poissona. Jakie jest prawdopodobieństwo, że noc upłynie lekarzowi spokojnie? U nas $\mu = 3$, $x = 0$ więc

$$P(0) = \frac{3^0 e^{-3}}{0!} = e^{-3} = 0,0498$$

Przykład

Jak ze zmiennych podlegających rozkładowi jednostajnemu uzyskać zmienne podlegające rozkładowi Poissona?

Wykonać symulację wynikającą z następujących spostrzeżeń:

- Weźmy jednostkę czasu. Ma w niej zachodzić średnio μ zdarzeń.
- Podzielmy jednostkę czasu na *NT* odcinków o jednakowej długości.
- Prawdopodobieństwo, że zdarzenie zajdzie w konkretnym odcinku jest $p = \frac{\mu}{NT}$ Takie prawdopodobieństwo ma też wylosowanie przy pomocy generatora liczb z rozkładu jednostajnego [0,1) liczby nie większej niż *p*.
- Zatem, aby wytworzyć jedną liczbę z rozkładu Poissona można policzyć ile spośród *NT* liczb z rozkładu jednostajnego [0,1) jest nie większych niż *p*.

```
# -*- coding: utf-8 -*-
import numpy as np
import pylab as py

def gen_poiss(mi,NT,N):
        '''funkcja zwracająca zmienne losowe z rozkładu poissona:
        średnia ilość zdarzeń o niskim prawdopodobieństwie w jednostce czasu
        parametry:
                mi: średnia ilość zdarzeń
                NT: ilość odcinków czasu
                N: ilość liczb do generowania
        funkcja zwraca:
                N liczb będących zliczeniami zdarzeń
        '''
        p = mi/NT
        k = np.zeros(N)
        # w pętli wytwarzam pojedyncze zmienne losowe
        # losuję NT liczb z rozkładu jednostajnego [0,1) i jako zajście
zdarzenia
        # traktuję wylosowanie liczby nie większej niż p. Ilość zdarzeń to
```

```
liczba z rozkładu Poissona
        for i in range(N):
                r = np.random.random(size=NT)
                k[i]=np.sum(r<=p)
        return k

# kod testujący
mi=4
N=10000
NT = 10000
x = gen_poiss(mi,NT,N)
bins =range(31)
py.hist(x,bins)
py.show()
```

Rozkład normalny i Centralne Twierdzenie Graniczne

Gęstość prawdopodobieństwa rozkładu normalnego dana jest wzorem:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$ gdzie: μ — średnia, σ — odchylenie standardowe (σ^2 — wariancja).

Jeden ze sposóbw wytwarzania zmiennych losowych o rozkładzie normalnym polega na zastosowaniu Centralnego Twierdzenia Granicznego.

$$X = \frac{\sqrt{12}}{\sqrt{N}} \left(\sum_{n=1}^{N} Y_n - \frac{N}{2} \right)$$

gdzie Y zmienna losowa z rozkładu jednostajnego $[0,1)$ W granicy dużych N rozkład zmiennej X dąży do rozkładu normalnego $N(0,1)$.

Zadanie

Proszę:

- napisać funkcję `gen_CTG_1(N)`, która zwraca liczbę losową X otrzymaną zgodnie powyższym wzorem przez sumowanie `N` liczb z rozkładu jednostajnego.
- napisać funkcję `gen_CTG(N, N_liczb)` korzystającą z `gen_CTG_1(N)` i zwracającą zadaną ilość liczb `N_liczb`
- narysować histogramy 10 000 liczb X uzyskanych kolejno dla $N = 1, 2, ..., 12$
- Jakie parametry charakteryzują rozkład do którego zbiegają sumy? Przy każdym z powyższych histogramów wypisz średnią i wariancję z próby (funkcje `np.mean` i `np.var`).

Rozwiązanie:

Projekt *Fizyka wobec wyzwań XXI wieku* współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

```
# -*- coding: utf-8 -*-
import numpy as np
import pylab as py
import numpy.random as rnd

def gen_CTG_1(N):
        '''Funkcja generująca pojedynczą zmienną losową będącą sumą N liczb z
rozkładu jednostajnego [0,1)
        pomnożoną przez czynnik normujący zapewniający wariancję równą 1.
        parametry:
                N: ile liczb sumować
        funkcja zwraca:
                jedną liczbę losową'''
        X = np.sqrt(12/N) * (np.sum(rnd.random(N)) - N/2)
        return X

def gen_CTG(N,N_liczb):
        '''Funkcja zwraca wektor (tablicę) zmiennych losowych będących sumami
N liczb z rozkładu jednostajnego [0,1).
        parametry:
                N: ile liczb sumować
                N_liczb: ilość liczb do generowania
        funkcja zwraca:
                N_liczb losowych'''

        X=np.zeros(N_liczb)

        for i in range(N_liczb):
                X[i] = gen_CTG_1(N)

        return X


# kod testujący
N_liczb=10000
Nmax=12

for N in range(1,Nmax+1):
        X=gen_CTG(N,N_liczb)
        py.subplot(3,4,N)
        bins = np.arange(-4,4,0.1)
        py.hist(X,bins=bins,normed=True)
        py.title("N=%i, μ=%.2g, σ²=%.2g"%(N, X.mean(), X.var(ddof=1)))

py.show()
```

Problem: transformacja rozkładu normalnego

Rozkład o średniej 0 i wariancji 1 (notacja $N(0,1)$) jest nazywany rozkładem standardowym i często jest oznaczany literą Z. Dokonując odpowiedniej transformacji można z rozkładu Z uzyskać dowolny inny rozkład normalny.

Proszę:

- wygenerować 100000 zmiennych losowych z rozkładu $N(0,1)$ (skorzystać z funkcji `st.norm.rvs(loc=0, scale=1, size=100000)`)
- przetransformować je do zmiennych losowych $N(2,9)$.
- narysować histogramy zmiennych przed i po transformacji.
- narysować rozkład gęstości prawdopodobieństwa dla rozkładu $N(2,9)$ (skorzystać z funkcji `st.norm.pdf(...)`).

Rozwiązanie

```
# -*- coding: utf-8 -*-
import numpy as np
import pylab as py
import scipy.stats as st

Z=st.norm.rvs(loc=,scale=1,size=100000) #losowanie liczb z rozkładu N(0,1)
(czyli Z)

std=9
mu=2
X=Z*std+mu #transformacja do σ=9, μ=2

szerokosc_binu=0.5
bins=np.arange(-40,40.5,szerokosc_binu) #te same granice binów dla obu
histogramów

py.subplot(211)
py.hist(Z,bins=bins,normed=True,label="histogram przed transformacją")
py.title("μ = %.3g    σ = %.3g"%(Z.mean(),Z.std()))
py.legend()

py.subplot(212)
py.hist(X,bins=bins,normed=True,label="histogram po transformacji")

PDF=st.norm.pdf(bins,loc=mu,scale=std) #modelowa gęstość prawdopodobieństwa
dla σ=9, μ=2
py.plot(bins+szerokosc_binu/2,PDF,"ro",label="gęstość prawd.")
```

```
py.title("μ = %.3g    σ = %.3g"%(X.mean(),X.std()))
py.legend()

py.show()
```

Transformacja Boxa-Mullera

To inna popularna metoda metoda generowania liczb losowych o rozkładzie normalnym, na podstawie dwóch wartości zmiennej o rozkładzie jednostajnym na przedziale [0,1) http://en.wikipedia.org/wiki/Box-Muller_transform, ale nie będziemy się nią tu szerzej zajmować.

Przykład

Producent silników twierdzi, że jego silniki mają średnią moc 220 KM, a odchylenie standardowe wynosi 15 KM. Potencjalny klient testuje 100 silników. Jakie jest prawdopodobieństwo, że średnia z próby będzie mniejsza niż 217 KM?

Przypomnijmy, że z CTG dla dużych liczebności próby $n\ \bar{x} \sim N(\mu, \sigma^2/n)$ (dowód). Zatem szukamy

$$P(\bar{x} < 217) = P\left(Z < \frac{217-\mu}{\frac{\sigma}{\sqrt{n}}}\right) = P\left(Z < \frac{217-220}{\frac{15}{\sqrt{100}}}\right) = P(Z < -2) = 0,0228$$

```
import scipy.stats as st
import pylab as py
import numpy as np

# ze wzoru
p=st.norm.cdf(-2)
print('prawdopodobieństwo odczytane z dystrybuanty rozkładu
normalnego: %(0).4f' %{'0':p})

# symulacja

mu=220
sig=15
N_prob=100

m_kryt=217

N_rep=int(1e4)
srednia=np.zeros(N_rep)
for i in range(N_rep):
    seria=st.norm.rvs(loc=mu, scale=sig, size=N_prob)
    srednia[i]=np.mean(seria)
```

```
h=py.hist(srednia,30);

py.plot([m_kryt, m_kryt],[, max(h[])],'r')
p1=np.sum(srednia<=m_kryt)/N_rep
print('prawdopodobieństwo uzyskane z symulacji: %(0).4f' %{'0':p1})
py.show()
```

Zadania

Teraz kilka najprostszych zadań dotyczących rozkładu normalnego:

- Znajdźmy prawdopodobieństwo, że $Z < -2{,}47$. Proszę zrobić to na dwa sposoby: raz z użyciem wygenerowanych zmiennych z rozkładu normalnego, drugi raz z użyciem dystrybuanty `norm.cdf`

[odp: $p = 0{,}0068$]

Rozwiązanie:

```
import scipy.stats as st

N=100000
z_crit = -2.47
Z=st.norm.rvs(loc=, scale=1, size=N)
p= sum(Z < z_crit )/N
p_cdf = st.norm.cdf(z_crit, loc=, scale=1)
print('symulowane p: %.4f' % p )
print('p z dystrybuanty: %.4f' % p_cdf)
```

- Znaleźć prawdopodobieństwo $P(|Z| < 2)$ [Odp: $p = 0{,}9545$]

Rozwiązanie:

```
import scipy.stats as st
import numpy as np

N=100000
z_crit = 2
Z=st.norm.rvs(loc=, scale=1, size=N)
p= sum(np.abs(Z) < z_crit )/N
p_cdf = st.norm.cdf( z_crit, loc=, scale=1) - st.norm.cdf( -z_crit, loc=,
scale=1)
print('symulowane p: %.4f' % p )
print('p z dystrybuanty: %.4f' % p_cdf)
```

- Koncentracja zanieczyszczeń w półprzewodniku używanym do produkcji procesorów podlega rozkładowi normalnemu o średniej 127 cząsteczek na milion i odchyleniu standardowemu 22. Półprzewodnik może zostać użyty jedynie gdy koncentracja zanieczyszczeń spada poniżej 150 cząstek na milion. Jaka proporcja półprzewodników nadaje się do użycia?

Prawdopodobieństwo obliczyć korzystając z dystrybuanty rozkładu normalnego oraz z symulacji. [Opd: p = 0,852]

Rozwiązanie:

```
import scipy.stats as st
import pylab as py
import numpy as np

mu=127
sig=22
m_kryt=150
p=st.norm.cdf(m_kryt, loc = mu, scale = sig )
print('proporcja odczytana z dystrybuanty rozkładu normalnego: %4g' %p)
# symulacja
N_prob=10000
seria=st.norm.rvs(loc=mu, scale=sig, size=N_prob)
p = sum(seria<=m_kryt)/N_prob
bins = np.arange(60,200,5)
h=py.hist(seria,bins);
py.plot([m_kryt, m_kryt],[, max(h[])],'r')
p1=np.sum(seria<=m_kryt)/N_prob
print('prawdopodobieństwo uzyskane z symulacji: %.4g' %p1)
py.show()
```

Rozkład t

Gdy nie znamy wariancji populacji σ^2 używamy w jego miejsce estymatora wariancji próby s^2 danego wzorem:

$s^2 = \frac{\sum(\bar{x} - x_i)^2}{n-1}$ gdzie n — rozmiar próby

Wtedy rozkład $Y = \frac{\bar{X} - \mu}{\frac{s}{\sqrt{n}}}$ nie podlega rozkładowi normalnemu. Jeśli populacja X podlega rozkładowi normalnemu to Y podlega rozkładowi *t* z $n-1$ stopniami swobody.

Zobaczmy na symulacji co to zmienia. Proszę obejrzeć wyniki dla kilku wartości N

```
# -*- coding: utf-8 -*-
```

```
import numpy as np
import pylab as py
import scipy.stats as st

N=3
N_liczb=10000
mu=2
sigma=1

y_wyidealizowane=np.zeros(N_liczb)
m_populacji=np.zeros(N_liczb)
s_populacji=np.zeros(N_liczb)
y_estymowane=np.zeros(N_liczb)
m_estymowane=np.zeros(N_liczb)
s_estymowane=np.zeros(N_liczb)
for i in range(N_liczb):
    x=st.norm.rvs(loc=mu,scale=sigma,size=N)
    m_populacji[i]=mu
    s_populacji[i]=sigma/np.sqrt(N)

    m_estymowane[i]=np.mean(x)
    s_estymowane[i]=np.std(x,ddof=1)/np.sqrt(N)

    y_wyidealizowane[i]=(m_estymowane[i]-mu)/(s_populacji[i])
    y_estymowane[i]=(m_estymowane[i]-mu)/(s_estymowane[i])

py.subplot(311)
py.title(u'średnia (wartość oczekiwana)')
py.plot(m_estymowane,label="estymowana")
py.plot(m_populacji,'r',label="populacji")
py.legend()

py.subplot(312)
py.title('odchylenie standardowe')
py.plot(s_estymowane,label="estymowane")
py.plot(s_populacji,'r',label='populacji')
py.legend()

py.subplot(325)
py.title(u'rozkład ilorazu Y dla wersji wyidealizowanej')
py.hist(y_wyidealizowane,30,(-4,4))

py.subplot(326)
py.title(u'rozkład ilorazu Y dla wersji estymowanej')
py.hist(y_estymowane,30,(-4,4))
```

```
py.show()
```

Inne rozkłady uzyskiwane przez transformacje

Metoda odwracania dystrybuanty

Niech ξ będzie zmienną losową o rozkładzie jednostajnym na przedziale $[0,1)$. Jej dystrybuanta J dana jest więc worem:

$$J(x) = P(\xi \leq x) = \begin{cases} 0 & \text{dla} \quad x < 0 \\ x & \text{dla} \quad 0 \leq x < 1 \\ 1 & \text{dla} \quad 1 \leq x \end{cases}$$

oraz niech F będzie dystrybuantą interesującego nas rozkładu. Jeśli F jest funkcją odwracalną to zmienna losowa zdefiniowana jako:

$$\eta = F^{-1}(\xi)$$

jest zmienną losową podlegającą rozkładowi o dystrybuancie F. Własność tę łatwo sprawdzić. Ponieważ dla każdego $x \in R$ mamy:

$P(\eta \leq x) = P(F^{-1}(\xi) \leq x) = P(\xi \leq F(x)) = F(x)$ więc F jest dystrybuantą zmiennej losowej η.

Czyli jeśli znamy dystrybuantę interesującego nas rozkładu prawdopodobieństwa i dystrybuanta ta jest odwracalna, to możemy generować liczby z rozkładu o dystrybuancie F.

Przykład

Wygenerować liczby pseudolosowe z rozkładu wykładniczego o dystrybuancie:

$$F(X) = 1 - e^{-\lambda x}$$

Rozwiązanie:

- Znajdujemy funkcję odwrotną do dystrybuanty:

$$G(u) = F^{-1}(u) = -\frac{1}{\lambda}\ln(1-u)$$

- Generujemy liczby losowe u z przedziału $[0,1)$ i stosujemy transformację $G(u)$
- Test generowanego rozkładu robimy przez porównanie dystrybuanty teoretycznej i empirycznej. Proszę zwrócić uwagę na funkcję estymującą dystrybuantę empiryczną

```
# -*- coding: utf-8 -*-
import pylab as py
```

Projekt *Fizyka wobec wyzwań XXI wieku* współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

```
import numpy as np


def gen_wykladniczy(lam, N):
        '''funkcja zwraca N liczb z rozkładu wykładniczego o parametrze lam
        korzystając z metody odwracania dystrybuanty
        Dystrybuanta rozkładu wykładniczego: F(x) = 1- exp(-lam*x)
        Funkcja do niej odwrotna: G(u) = -1/lam ln(1-u)'''

        u = np.random.random(N)
        return -1.0/lam* np.log(1.-u)

def dystrybuanta(X,od=, do=4, krok=0.05):
        '''funkcja zwraca dystrybuantę empiryczną
        zmiennej X na przedziale (od, do) z krokiem krok'''

        os_x = np.arange(od,do,krok)
        nx = len(os_x)
        N_liczb = len(X)
        D=np.zeros(nx) #wyzerowany wektor na naszą dystrybuantę
        for i in range(nx):
                D[i] = sum(X <= os_x[i])/N_liczb
        return (os_x, D)

# testujemy: narysujemy dystrybuantę oczekiwaną i empiryczną
lam=3
N=10000

x = np.arange(,4,0.01)
F = 1-np.exp(-lam*x)
X = gen_wykladniczy(lam, N)
(os_x, F_emp) = dystrybuanta(X,od=, do=4, krok=0.1)
py.plot(x,F,'r',label = 'dystrybuanta teoretyczna')
py.plot(os_x, F_emp,'.b', label = 'dystrybuanta empiryczna')
py.legend()
py.show()
```

Zauważmy, że nie zawsze jest łatwo wyznaczyć efektywnie F^{-1} — jest tak na przykład w przypadku rozkładu normalnego. W takich sytuacjach można szukać przybliżonej wartości F^{-1}, rozwiązując numerycznie równanie: $F(x) = u$, co sprowadza się do znalezienia miejsc zerowych funkcji $H(x) = F(x) - u$.