References

[R805], [R806]

`sympy.stats.Coin(name, p=1 / 2)`

Create a Finite Random Variable representing a Coin toss.

Parameters

p : Rational Numeber between 0 and 1

Represents probability of getting "Heads", by default is Half

Returns

RandomSymbol

Examples

```
>>> from sympy.stats import Coin, density
>>> from sympy import Rational
```

```
>>> C = Coin('C') # A fair coin toss
>>> density(C).dict
{H: 1/2, T: 1/2}
```

```
>>> C2 = Coin('C2', Rational(3, 5)) # An unfair coin
>>> density(C2).dict
{H: 3/5, T: 2/5}
```

See also:

sympy.stats.Binomial (page 2877)

References

[R807]

`sympy.stats.Binomial(name, n, p, succ=1, fail=0)`

Create a Finite Random Variable representing a binomial distribution.

Parameters

n : Positive Integer

Represents number of trials

p : Rational Number between 0 and 1

Represents probability of success

succ : Integer/symbol/string

Represents event of success, by default is 1

fail : Integer/symbol/string

Represents event of failure, by default is 0

Returns

RandomSymbol

Examples

```
>>> from sympy.stats import Binomial, density
>>> from sympy import S, Symbol
```

```
>>> X = Binomial('X', 4, S.Half) # Four "coin flips"
>>> density(X).dict
{0: 1/16, 1: 1/4, 2: 3/8, 3: 1/4, 4: 1/16}
```

```
>>> n = Symbol('n', positive=True, integer=True)
>>> p = Symbol('p', positive=True)
>>> X = Binomial('X', n, S.Half) # n "coin flips"
>>> density(X).dict
Density(BinomialDistribution(n, 1/2, 1, 0))
>>> density(X).dict.subs(n, 4).doit()
{0: 1/16, 1: 1/4, 2: 3/8, 3: 1/4, 4: 1/16}
```

References

[R808], [R809]

sympy.stats.**BetaBinomial**(*name, n, alpha, beta*)

Create a Finite Random Variable representing a Beta-binomial distribution.

Parameters

n : Positive Integer

Represents number of trials

alpha : Real positive number

beta : Real positive number

Returns

RandomSymbol

Examples

```
>>> from sympy.stats import BetaBinomial, density
```

```
>>> X = BetaBinomial('X', 2, 1, 1)
>>> density(X).dict
{0: 1/3, 1: 2*beta(2, 2), 2: 1/3}
```

References

[R810], [R811]

sympy.stats.**Hypergeometric**(*name*, *N*, *m*, *n*)

Create a Finite Random Variable representing a hypergeometric distribution.

Parameters

N : Positive Integer

Represents finite population of size N.

m : Positive Integer

Represents number of trials with required feature.

n : Positive Integer

Represents numbers of draws.

Returns

RandomSymbol

Examples

```
>>> from sympy.stats import Hypergeometric, density
```

```
>>> X = Hypergeometric('X', 10, 5, 3) # 10 marbles, 5 white (success), 3␣
↪draws
>>> density(X).dict
{0: 1/12, 1: 5/12, 2: 5/12, 3: 1/12}
```


References

[R812], [R813]

sympy.stats.**FiniteRV**(*name*, *density*, ***kwargs*)

Create a Finite Random Variable given a dict representing the density.

Parameters

name : Symbol

Represents name of the random variable.

density : dict

Dictionary containing the pdf of finite distribution

check : bool

If True, it will check whether the given density integrates to 1 over the given set. If False, it will not perform this check. Default is False.

Returns

RandomSymbol

Examples

```
>>> from sympy.stats import FiniteRV, P, E
```

```
>>> density = {0: .1, 1: .2, 2: .3, 3: .4}
>>> X = FiniteRV('X', density)
```

```
>>> E(X)
2.00000000000000
>>> P(X >= 2)
0.700000000000000
```

sympy.stats.**Rademacher**(*name*)

Create a Finite Random Variable representing a Rademacher distribution.

Returns

RandomSymbol

Examples

```
>>> from sympy.stats import Rademacher, density
```

```
>>> X = Rademacher('X')
>>> density(X).dict
{-1: 1/2, 1: 1/2}
```

See also:

sympy.stats.Bernoulli (page 2876)

References

[R814]

Discrete Types

sympy.stats.**Geometric**(*name, p*)

Create a discrete random variable with a Geometric distribution.

Parameters

p : A probability between 0 and 1

Returns

RandomSymbol

Explanation

The density of the Geometric distribution is given by

$$f(k) := p(1-p)^{k-1}$$

Examples

```
>>> from sympy.stats import Geometric, density, E, variance
>>> from sympy import Symbol, S
```

```
>>> p = S.One / 5
>>> z = Symbol("z")
```

```
>>> X = Geometric("x", p)
```

```
>>> density(X)(z)
(4/5)**(z - 1)/5
```

```
>>> E(X)
5
```

```
>>> variance(X)
20
```


References

[R815], [R816]

sympy.stats.**Hermite**(*name*, *a1*, *a2*)

Create a discrete random variable with a Hermite distribution.

Parameters

a1 : A Positive number greater than equal to 0.

a2 : A Positive number greater than equal to 0.

Returns

RandomSymbol

Explanation

The density of the Hermite distribution is given by

$$f(x) := e^{-a_1-a_2} \sum_{j=0}^{\lfloor x/2 \rfloor} \frac{a_1^{x-2j} a_2^j}{(x-2j)!j!}$$

Examples

```
>>> from sympy.stats import Hermite, density, E, variance
>>> from sympy import Symbol
```

```
>>> a1 = Symbol("a1", positive=True)
>>> a2 = Symbol("a2", positive=True)
>>> x = Symbol("x")
```

```
>>> H = Hermite("H", a1=5, a2=4)
```

```
>>> density(H)(2)
33*exp(-9)/2
```

```
>>> E(H)
13
```

```
>>> variance(H)
21
```

References

[R817]

sympy.stats.**Poisson**(*name, lamda*)

Create a discrete random variable with a Poisson distribution.

Parameters

lamda : Positive number, a rate

Returns

RandomSymbol

Explanation

The density of the Poisson distribution is given by

$$f(k) := \frac{\lambda^k e^{-\lambda}}{k!}$$

Examples

```
>>> from sympy.stats import Poisson, density, E, variance
>>> from sympy import Symbol, simplify
```

```
>>> rate = Symbol("lambda", positive=True)
>>> z = Symbol("z")
```

```
>>> X = Poisson("x", rate)
```

```
>>> density(X)(z)
lambda**z*exp(-lambda)/factorial(z)
```

```
>>> E(X)
lambda
```

```
>>> simplify(variance(X))
lambda
```

References

[R818], [R819]

sympy.stats.**Logarithmic**(*name*, *p*)

Create a discrete random variable with a Logarithmic distribution.

Parameters

p : A value between 0 and 1

Returns

RandomSymbol

Explanation

The density of the Logarithmic distribution is given by

$$f(k) := \frac{-p^k}{k \ln{(1 - p)}}$$

Examples

```
>>> from sympy.stats import Logarithmic, density, E, variance
>>> from sympy import Symbol, S
```

```
>>> p = S.One / 5
>>> z = Symbol("z")
```

```
>>> X = Logarithmic("x", p)
```

```
>>> density(X)(z)
-1/(5**z*z*log(4/5))
```

```
>>> E(X)
-1/(-4*log(5) + 8*log(2))
```

```
>>> variance(X)
-1/((-4*log(5) + 8*log(2))*(-2*log(5) + 4*log(2))) + 1/(-
↪64*log(2)*log(5) + 64*log(2)**2 + 16*log(5)**2) - 10/(-32*log(5) +
↪64*log(2))
```

References

[R820], [R821]

sympy.stats.**NegativeBinomial**(*name*, *r*, *p*)

Create a discrete random variable with a Negative Binomial distribution.

Parameters

r : A positive value

p : A value between 0 and 1

Returns

RandomSymbol

Explanation

The density of the Negative Binomial distribution is given by

$$f(k) := \binom{k + r - 1}{k} (1 - p)^r p^k$$

Examples

```
>>> from sympy.stats import NegativeBinomial, density, E, variance
>>> from sympy import Symbol, S
```

```
>>> r = 5
>>> p = S.One / 5
>>> z = Symbol("z")
```

```
>>> X = NegativeBinomial("x", r, p)
```

```
>>> density(X)(z)
1024*binomial(z + 4, z)/(3125*5**z)
```

```
>>> E(X)
5/4
```

```
>>> variance(X)
25/16
```

References

[R822], [R823]

sympy.stats.**Skellam**(*name*, *mu1*, *mu2*)

Create a discrete random variable with a Skellam distribution.

Parameters

mu1 : A non-negative value

mu2 : A non-negative value

Returns

RandomSymbol

Explanation

The Skellam is the distribution of the difference N1 - N2 of two statistically independent random variables N1 and N2 each Poisson-distributed with respective expected values mu1 and mu2.

The density of the Skellam distribution is given by

$$f(k) := e^{-(\mu_1+\mu_2)}(\frac{\mu_1}{\mu_2})^{k/2}I_k(2\sqrt{\mu_1\mu_2})$$

Examples

```
>>> from sympy.stats import Skellam, density, E, variance
>>> from sympy import Symbol, pprint
```

```
>>> z = Symbol("z", integer=True)
>>> mu1 = Symbol("mu1", positive=True)
>>> mu2 = Symbol("mu2", positive=True)
>>> X = Skellam("x", mu1, mu2)
```

```
>>> pprint(density(X)(z), use_unicode=False)
     z
     -
     2
/mu1\   -mu1 - mu2        /       _____   _____\
|---| *e          *besseli\z, 2*\/ mu1 *\/ mu2 /
\mu2/
>>> E(X)
mu1 - mu2
>>> variance(X).expand()
mu1 + mu2
```

References

[R824]

sympy.stats.**YuleSimon**(*name*, *rho*)

Create a discrete random variable with a Yule-Simon distribution.

Parameters
rho : A positive value

Returns
RandomSymbol

Explanation

The density of the Yule-Simon distribution is given by

$$f(k) := \rho B(k, \rho + 1)$$

Examples

```
>>> from sympy.stats import YuleSimon, density, E, variance
>>> from sympy import Symbol, simplify
```

```
>>> p = 5
>>> z = Symbol("z")
```

```
>>> X = YuleSimon("x", p)
```

```
>>> density(X)(z)
5*beta(z, 6)
```

```
>>> simplify(E(X))
5/4
```

```
>>> simplify(variance(X))
25/48
```

References

[R825]

sympy.stats.**Zeta**(*name*, *s*)

Create a discrete random variable with a Zeta distribution.

Parameters
s : A value greater than 1

Returns
RandomSymbol

Explanation

The density of the Zeta distribution is given by

$$f(k) := \frac{1}{k^s \zeta(s)}$$

Examples

```
>>> from sympy.stats import Zeta, density, E, variance
>>> from sympy import Symbol
```

```
>>> s = 5
>>> z = Symbol("z")
```

```
>>> X = Zeta("x", s)
```

```
>>> density(X)(z)
1/(z**5*zeta(5))
```

```
>>> E(X)
pi**4/(90*zeta(5))
```

```
>>> variance(X)
-pi**8/(8100*zeta(5)**2) + zeta(3)/zeta(5)
```


References

[R826]

Continuous Types

sympy.stats.**Arcsin**(*name*, *a=0*, *b=1*)

Create a Continuous Random Variable with an arcsin distribution.

The density of the arcsin distribution is given by

$$f(x) := \frac{1}{\pi\sqrt{(x-a)(b-x)}}$$

with $x \in (a, b)$. It must hold that $-\infty < a < b < \infty$.

Parameters

a : Real number, the left interval boundary

b : Real number, the right interval boundary

Returns

RandomSymbol

Examples

```
>>> from sympy.stats import Arcsin, density, cdf
>>> from sympy import Symbol
```

```
>>> a = Symbol("a", real=True)
>>> b = Symbol("b", real=True)
>>> z = Symbol("z")
```

```
>>> X = Arcsin("x", a, b)
```

```
>>> density(X)(z)
1/(pi*sqrt((-a + z)*(b - z)))
```

```
>>> cdf(X)(z)
Piecewise((0, a > z),
        (2*asin(sqrt((-a + z)/(-a + b)))/pi, b >= z),
        (1, True))
```

References

[R827]

sympy.stats.**Benini**(*name*, *alpha*, *beta*, *sigma*)

Create a Continuous Random Variable with a Benini distribution.

The density of the Benini distribution is given by

$$f(x) := e^{-\alpha \log \frac{x}{\sigma} - \beta \log^2 \left[\frac{x}{\sigma}\right]} \left(\frac{\alpha}{x} + \frac{2\beta \log \frac{x}{\sigma}}{x}\right)$$

This is a heavy-tailed distribution and is also known as the log-Rayleigh distribution.

Parameters

alpha : Real number, $\alpha > 0$, a shape

beta : Real number, $\beta > 0$, a shape

sigma : Real number, $\sigma > 0$, a scale

Returns

RandomSymbol

Examples

```
>>> from sympy.stats import Benini, density, cdf
>>> from sympy import Symbol, pprint
```

```
>>> alpha = Symbol("alpha", positive=True)
>>> beta = Symbol("beta", positive=True)
>>> sigma = Symbol("sigma", positive=True)
>>> z = Symbol("z")
```

```
>>> X = Benini("x", alpha, beta, sigma)
```

```
>>> D = density(X)(z)
>>> pprint(D, use_unicode=False)
/                  /  z  \\             /  z  \            2/  z  \
|        2*beta*log|-----||  - alpha*log|-----| - beta*log  |-----|
|alpha             \sigma/|             \sigma/             \sigma/
|----- + -----------------|*e
\  z             z        /
```

```
>>> cdf(X)(z)
Piecewise((1 - exp(-alpha*log(z/sigma) - beta*log(z/sigma)**2), sigma <=␣
↪z),
        (0, True))
```

References

[R828], [R829]

sympy.stats.**Beta**(*name, alpha, beta*)

Create a Continuous Random Variable with a Beta distribution.

The density of the Beta distribution is given by

$$f(x) := \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\mathrm{B}(\alpha,\beta)}$$

with $x \in [0,1]$.

Parameters

alpha : Real number, $\alpha > 0$, a shape

beta : Real number, $\beta > 0$, a shape

Returns

RandomSymbol

Examples

```
>>> from sympy.stats import Beta, density, E, variance
>>> from sympy import Symbol, simplify, pprint, factor
```

```
>>> alpha = Symbol("alpha", positive=True)
>>> beta = Symbol("beta", positive=True)
>>> z = Symbol("z")
```

```
>>> X = Beta("x", alpha, beta)
```

```
>>> D = density(X)(z)
>>> pprint(D, use_unicode=False)
 alpha - 1        beta - 1
```

(continues on next page)

(continued from previous page)

```
z          *(1 - z)
--------------------------
      B(alpha, beta)
```

```
>>> simplify(E(X))
alpha/(alpha + beta)
```

```
>>> factor(simplify(variance(X)))
alpha*beta/((alpha + beta)**2*(alpha + beta + 1))
```

References

[R830], [R831]

sympy.stats.**BetaNoncentral**(*name, alpha, beta, lamda*)

Create a Continuous Random Variable with a Type I Noncentral Beta distribution.

The density of the Noncentral Beta distribution is given by

$$f(x) := \sum_{k=0}^\infty e^{-\lambda/2}\frac{(\lambda/2)^k}{k!} \frac{x^{\alpha+k-1}(1-x)^{\beta-1}}{\mathrm{B}(\alpha+k,\beta)}$$

with $x \in [0,1]$.

Parameters

alpha : Real number, $\alpha > 0$, a shape

beta : Real number, $\beta > 0$, a shape

lamda : Real number, $\lambda \geq 0$, noncentrality parameter

Returns

RandomSymbol

Examples

```
>>> from sympy.stats import BetaNoncentral, density, cdf
>>> from sympy import Symbol, pprint
```

```
>>> alpha = Symbol("alpha", positive=True)
>>> beta = Symbol("beta", positive=True)
>>> lamda = Symbol("lamda", nonnegative=True)
>>> z = Symbol("z")
```

```
>>> X = BetaNoncentral("x", alpha, beta, lamda)
```

```
>>> D = density(X)(z)
>>> pprint(D, use_unicode=False)
  oo
 _____
```

(continues on next page)

(continued from previous page)

```
\     `
 \                                               -lamda
  \                        k                     -------
   \    k + alpha - 1 /lamda\         beta - 1     2
    )  z             *|-----| *(1 - z)        *e
   /                  \  2  /
  /    ------------------------------------------------
 /                 B(k + alpha, beta)*k!
/____,
k = 0
```

Compute cdf with specific 'x', 'alpha', 'beta' and 'lamda' values as follows:

```
>>> cdf(BetaNoncentral("x", 1, 1, 1), evaluate=False)(2).doit()
2*exp(1/2)
```

The argument evaluate=False prevents an attempt at evaluation of the sum for general x, before the argument 2 is passed.

References

[R832], [R833]

sympy.stats.**BetaPrime**(*name*, *alpha*, *beta*)

Create a continuous random variable with a Beta prime distribution.

The density of the Beta prime distribution is given by

$$f(x) := \frac{x^{\alpha-1}(1+x)^{-\alpha-\beta}}{B(\alpha,\beta)}$$

with $x > 0$.

Parameters

alpha : Real number, $\alpha > 0$, a shape

beta : Real number, $\beta > 0$, a shape

Returns

RandomSymbol

Examples

```
>>> from sympy.stats import BetaPrime, density
>>> from sympy import Symbol, pprint
```

```
>>> alpha = Symbol("alpha", positive=True)
>>> beta = Symbol("beta", positive=True)
>>> z = Symbol("z")
```

```
>>> X = BetaPrime("x", alpha, beta)
```

```
>>> D = density(X)(z)
>>> pprint(D, use_unicode=False)
 alpha - 1        -alpha - beta
z         *(z + 1)
-------------------------------
         B(alpha, beta)
```

References

[R834], [R835]

sympy.stats.**BoundedPareto**(*name, alpha, left, right*)

Create a continuous random variable with a Bounded Pareto distribution.

The density of the Bounded Pareto distribution is given by

$$f(x) := \frac{\alpha L^{\alpha}x^{-\alpha-1}}{1-(\frac{L}{H})^{\alpha}}$$

Parameters

alpha : Real Number, $\alpha > 0$

Shape parameter

left : Real Number, $left > 0$

Location parameter

right : Real Number, $right > left$

Location parameter

Returns

RandomSymbol

Examples

```
>>> from sympy.stats import BoundedPareto, density, cdf, E
>>> from sympy import symbols
>>> L, H = symbols('L, H', positive=True)
>>> X = BoundedPareto('X', 2, L, H)
>>> x = symbols('x')
>>> density(X)(x)
2*L**2/(x**3*(1 - L**2/H**2))
>>> cdf(X)(x)
Piecewise((-H**2*L**2/(x**2*(H**2 - L**2)) + H**2/(H**2 - L**2), L <= x),
↪ (0, True))
>>> E(X).simplify()
2*H*L/(H + L)
```

References

[R836]

sympy.stats.**Cauchy**(*name*, *x0*, *gamma*)

Create a continuous random variable with a Cauchy distribution.

The density of the Cauchy distribution is given by

$$f(x) := \frac{1}{\pi\gamma[1 + (\frac{x-x_0}{\gamma})^2]}$$

Parameters

x0 : Real number, the location

gamma : Real number, $\gamma > 0$, a scale

Returns

RandomSymbol

Examples

```
>>> from sympy.stats import Cauchy, density
>>> from sympy import Symbol
```

```
>>> x0 = Symbol("x0")
>>> gamma = Symbol("gamma", positive=True)
>>> z = Symbol("z")
```

```
>>> X = Cauchy("x", x0, gamma)
```

```
>>> density(X)(z)
1/(pi*gamma*(1 + (-x0 + z)**2/gamma**2))
```

References

[R837], [R838]

sympy.stats.**Chi**(*name*, *k*)

Create a continuous random variable with a Chi distribution.

The density of the Chi distribution is given by

$$f(x) := \frac{2^{1-k/2}x^{k-1}e^{-x^2/2}}{\Gamma(k/2)}$$

with $x \geq 0$.

Parameters

k : Positive integer, The number of degrees of freedom

Returns

RandomSymbol

Examples

```
>>> from sympy.stats import Chi, density, E
>>> from sympy import Symbol, simplify
```

```
>>> k = Symbol("k", integer=True)
>>> z = Symbol("z")
```

```
>>> X = Chi("x", k)
```

```
>>> density(X)(z)
2**(1 - k/2)*z**(k - 1)*exp(-z**2/2)/gamma(k/2)
```

```
>>> simplify(E(X))
sqrt(2)*gamma(k/2 + 1/2)/gamma(k/2)
```

References

[R839], [R840]

sympy.stats.**ChiNoncentral**(*name*, *k*, *l*)

Create a continuous random variable with a non-central Chi distribution.

Parameters

k : A positive Integer, $k > 0$

The number of degrees of freedom.

lambda : Real number, $\lambda > 0$

Shift parameter.

Returns

RandomSymbol

Explanation

The density of the non-central Chi distribution is given by

$$f(x) := \frac{e^{-(x^2+\lambda^2)/2} x^k \lambda}{(\lambda x)^{k/2}} I_{k/2-1}(\lambda x)$$

with $x \geq 0$. Here, $I_\nu(x)$ is the *modified Bessel function of the first kind* (page 499).

Examples

```
>>> from sympy.stats import ChiNoncentral, density
>>> from sympy import Symbol
```

```
>>> k = Symbol("k", integer=True)
>>> l = Symbol("l")
>>> z = Symbol("z")
```

```
>>> X = ChiNoncentral("x", k, l)
```

```
>>> density(X)(z)
l*z**k*exp(-l**2/2 - z**2/2)*besseli(k/2 - 1, l*z)/(l*z)**(k/2)
```

References

[R841]

sympy.stats.**ChiSquared**(*name*, *k*)

Create a continuous random variable with a Chi-squared distribution.

Parameters

k : Positive integer

The number of degrees of freedom.

Returns

RandomSymbol

Explanation

The density of the Chi-squared distribution is given by

$$f(x) := \frac{1}{2^{\frac{k}{2}}\Gamma\left(\frac{k}{2}\right)} x^{\frac{k}{2}-1} e^{-\frac{x}{2}}$$

with $x \geq 0$.

Examples

```
>>> from sympy.stats import ChiSquared, density, E, variance, moment
>>> from sympy import Symbol
```

```
>>> k = Symbol("k", integer=True, positive=True)
>>> z = Symbol("z")
```

```
>>> X = ChiSquared("x", k)
```

```
>>> density(X)(z)
z**(k/2 - 1)*exp(-z/2)/(2**(k/2)*gamma(k/2))
```

```
>>> E(X)
k
```

```
>>> variance(X)
2*k
```

```
>>> moment(X, 3)
k**3 + 6*k**2 + 8*k
```

References

[R842], [R843]

`sympy.stats.Dagum(name, p, a, b)`

Create a continuous random variable with a Dagum distribution.

Parameters

p : Real number

$p > 0$, a shape.

a : Real number

$a > 0$, a shape.

b : Real number

$b > 0$, a scale.

Returns

RandomSymbol

Explanation

The density of the Dagum distribution is given by

$$f(x) := \frac{ap}{x} \left(\frac{\left(\frac{x}{b}\right)^{ap}}{\left(\left(\frac{x}{b}\right)^{a} + 1\right)^{p+1}} \right)$$

with $x > 0$.

Examples

```
>>> from sympy.stats import Dagum, density, cdf
>>> from sympy import Symbol
```

```
>>> p = Symbol("p", positive=True)
>>> a = Symbol("a", positive=True)
>>> b = Symbol("b", positive=True)
>>> z = Symbol("z")
```

```
>>> X = Dagum("x", p, a, b)
```

```
>>> density(X)(z)
a*p*(z/b)**(a*p)*((z/b)**a + 1)**(-p - 1)/z
```

```
>>> cdf(X)(z)
Piecewise(((1 + (z/b)**(-a))**(-p), z >= 0), (0, True))
```

References

[R844]

sympy.stats.**Erlang**(*name*, *k*, *l*)

Create a continuous random variable with an Erlang distribution.

Parameters

k : Positive integer

l : Real number, $\lambda > 0$, the rate

Returns

RandomSymbol

Explanation

The density of the Erlang distribution is given by

$$f(x) := \frac{\lambda^k x^{k-1} e^{-\lambda x}}{(k-1)!}$$

with $x \in [0, \infty]$.

Examples

```
>>> from sympy.stats import Erlang, density, cdf, E, variance
>>> from sympy import Symbol, simplify, pprint
```

```
>>> k = Symbol("k", integer=True, positive=True)
>>> l = Symbol("l", positive=True)
>>> z = Symbol("z")
```

```
>>> X = Erlang("x", k, l)
```

```
>>> D = density(X)(z)
>>> pprint(D, use_unicode=False)
 k  k - 1  -l*z
l *z     *e
---------------
    Gamma(k)
```

```
>>> C = cdf(X)(z)
>>> pprint(C, use_unicode=False)
/lowergamma(k, l*z)
|------------------  for z > 0
<     Gamma(k)
|
\        0           otherwise
```

```
>>> E(X)
k/l
```

```
>>> simplify(variance(X))
k/l**2
```

References

[R845], [R846]

sympy.stats.**ExGaussian**(*name*, *mean*, *std*, *rate*)

Create a continuous random variable with an Exponentially modified Gaussian (EMG) distribution.

Parameters

name : A string giving a name for this distribution

mean : A Real number, the mean of Gaussian component

std : A positive Real number,

math

$\sigma^2 > 0$ the variance of Gaussian component

rate : A positive Real number,

math

$\lambda > 0$ the rate of Exponential component

Returns

RandomSymbol

Explanation

The density of the exponentially modified Gaussian distribution is given by

$$f(x) := \frac{\lambda}{2}e^{\frac{\lambda}{2}(2\mu+\lambda\sigma^2-2x)}\text{erfc}(\frac{\mu + \lambda\sigma^2 - x}{\sqrt{2}\sigma})$$

with $x > 0$. Note that the expected value is $1/\lambda$.

Examples

```
>>> from sympy.stats import ExGaussian, density, cdf, E
>>> from sympy.stats import variance, skewness
>>> from sympy import Symbol, pprint, simplify
```

```
>>> mean = Symbol("mu")
>>> std = Symbol("sigma", positive=True)
>>> rate = Symbol("lamda", positive=True)
>>> z = Symbol("z")
>>> X = ExGaussian("x", mean, std, rate)
```

```
>>> pprint(density(X)(z), use_unicode=False)
             /           2              \
       lamda*\lamda*sigma  + 2*mu - 2*z/
       ---------------------------------     /  ___ /           2        ␣
↪ \\
                      2                      |\/ 2 *\lamda*sigma  + mu -␣
↪z/|
lamda*e                                  *erfc|--------------------------
↪--|
                                             \           2*sigma         ␣
↪  /
-------------------------------------------------------------------------
↪---
                                    2
```

```
>>> cdf(X)(z)
-(erf(sqrt(2)*(-lamda**2*sigma**2 + lamda*(-mu + z))/(2*lamda*sigma))/2␣
↪+ 1/2)*exp(lamda**2*sigma**2/2 - lamda*(-mu + z)) + erf(sqrt(2)*(-mu +␣
↪z)/(2*sigma))/2 + 1/2
```

```
>>> E(X)
(lamda*mu + 1)/lamda
```

```
>>> simplify(variance(X))
sigma**2 + lamda**(-2)
```

```
>>> simplify(skewness(X))
2/(lamda**2*sigma**2 + 1)**(3/2)
```

References

[R847]

sympy.stats.**Exponential**(*name*, *rate*)

Create a continuous random variable with an Exponential distribution.

Parameters

rate : A positive Real number, $\lambda > 0$, the rate (or inverse scale/inverse mean)

Returns

RandomSymbol

Explanation

The density of the exponential distribution is given by

$$f(x) := \lambda \exp(-\lambda x)$$

with $x > 0$. Note that the expected value is $1/\lambda$.

Examples

```
>>> from sympy.stats import Exponential, density, cdf, E
>>> from sympy.stats import variance, std, skewness, quantile
>>> from sympy import Symbol
```

```
>>> l = Symbol("lambda", positive=True)
>>> z = Symbol("z")
>>> p = Symbol("p")
>>> X = Exponential("x", l)
```

```
>>> density(X)(z)
lambda*exp(-lambda*z)
```

```
>>> cdf(X)(z)
Piecewise((1 - exp(-lambda*z), z >= 0), (0, True))
```

```
>>> quantile(X)(p)
-log(1 - p)/lambda
```

```
>>> E(X)
1/lambda
```

```
>>> variance(X)
lambda**(-2)
```

```
>>> skewness(X)
2
```

```
>>> X = Exponential('x', 10)
```

```
>>> density(X)(z)
10*exp(-10*z)
```

```
>>> E(X)
1/10
```

```
>>> std(X)
1/10
```

References

[R848], [R849]

sympy.stats.**FDistribution**(*name*, *d1*, *d2*)

Create a continuous random variable with a F distribution.

Parameters

d1 : $d_1 > 0$, where d_1 is the degrees of freedom $(n_1 - 1)$

d2 : $d_2 > 0$, where d_2 is the degrees of freedom $(n_2 - 1)$

Returns

RandomSymbol

Explanation

The density of the F distribution is given by

$$f(x) := \frac{\sqrt{\frac{(d_1 x)^{d_1} d_2^{d_2}}{(d_1 x + d_2)^{d_1 + d_2}}}}{x \mathrm{B}\left(\frac{d_1}{2}, \frac{d_2}{2}\right)}$$

with $x > 0$.

Examples

```
>>> from sympy.stats import FDistribution, density
>>> from sympy import Symbol, pprint
```

```
>>> d1 = Symbol("d1", positive=True)
>>> d2 = Symbol("d2", positive=True)
>>> z = Symbol("z")
```

```
>>> X = FDistribution("x", d1, d2)
```

```
>>> D = density(X)(z)
>>> pprint(D, use_unicode=False)
  d2
  --
  2    /       d1             -d1 - d2
d2  *\/  (d1*z)  *(d1*z + d2)
--------------------------------------
                /d1  d2\
             z*B|--, --|
                \2   2 /
```

References

[R850], [R851]

sympy.stats.**FisherZ**(*name, d1, d2*)

Create a Continuous Random Variable with an Fisher's Z distribution.

Parameters

d1 : $d_1 > 0$

Degree of freedom.

d2 : $d_2 > 0$

Degree of freedom.

Returns

RandomSymbol

Explanation

The density of the Fisher's Z distribution is given by

$$f(x) := \frac{2d_1^{d_1/2} d_2^{d_2/2}}{\mathrm{B}(d_1/2, d_2/2)} \frac{e^{d_1 z}}{\left(d_1 e^{2z} + d_2\right)^{\left(d_1 + d_2\right)/2}}$$

Examples

```
>>> from sympy.stats import FisherZ, density
>>> from sympy import Symbol, pprint
```

```
>>> d1 = Symbol("d1", positive=True)
>>> d2 = Symbol("d2", positive=True)
>>> z = Symbol("z")
```

```
>>> X = FisherZ("x", d1, d2)
```

```
>>> D = density(X)(z)
>>> pprint(D, use_unicode=False)
                                d1   d2
    d1   d2                   - -- - --
    --   --                     2    2
    2    2  /     2*z      \           d1*z
2*d1  *d2  *\d1*e     + d2/          *e
-----------------------------------------
                 /d1  d2\
                B|--, --|
                 \2   2 /
```

References

[R852], [R853]

sympy.stats.**Frechet**(*name*, *a*, *s=1*, *m=0*)

Create a continuous random variable with a Frechet distribution.

Parameters

a : Real number, $a \in (0, \infty)$ the shape

s : Real number, $s \in (0, \infty)$ the scale

m : Real number, $m \in (-\infty, \infty)$ the minimum

Returns

RandomSymbol

Explanation

The density of the Frechet distribution is given by

$$f(x) := \frac{\alpha}{s} \left(\frac{x-m}{s}\right)^{-1-\alpha} e^{-(\frac{x-m}{s})^{-\alpha}}$$

with $x \geq m$.

Examples

```
>>> from sympy.stats import Frechet, density, cdf
>>> from sympy import Symbol
```

```
>>> a = Symbol("a", positive=True)
>>> s = Symbol("s", positive=True)
>>> m = Symbol("m", real=True)
>>> z = Symbol("z")
```

```
>>> X = Frechet("x", a, s, m)
```

```
>>> density(X)(z)
a*((-m + z)/s)**(-a - 1)*exp(-1/((-m + z)/s)**a)/s
```

```
>>> cdf(X)(z)
Piecewise((exp(-1/((-m + z)/s)**a), m <= z), (0, True))
```

References

[R854]

sympy.stats.**Gamma**(*name*, *k*, *theta*)

Create a continuous random variable with a Gamma distribution.

Parameters

k : Real number, $k > 0$, a shape

theta : Real number, $\theta > 0$, a scale

Returns

RandomSymbol

Explanation

The density of the Gamma distribution is given by

$$f(x) := \frac{1}{\Gamma(k)\theta^k} x^{k-1} e^{-\frac{x}{\theta}}$$

with $x \in [0, 1]$.

Examples

```
>>> from sympy.stats import Gamma, density, cdf, E, variance
>>> from sympy import Symbol, pprint, simplify
```

```
>>> k = Symbol("k", positive=True)
>>> theta = Symbol("theta", positive=True)
>>> z = Symbol("z")
```

```
>>> X = Gamma("x", k, theta)
```

```
>>> D = density(X)(z)
>>> pprint(D, use_unicode=False)
                  -z
                -----
     -k  k - 1  theta
theta  *z     *e
---------------------
       Gamma(k)
```

```
>>> C = cdf(X, meijerg=True)(z)
>>> pprint(C, use_unicode=False)
/            /     z  \
|k*lowergamma|k, -----|
|            \   theta/
<----------------------  for z >= 0
|     Gamma(k + 1)
|
\          0             otherwise
```

```
>>> E(X)
k*theta
```

```
>>> V = simplify(variance(X))
>>> pprint(V, use_unicode=False)
       2
k*theta
```

References

[R855], [R856]

sympy.stats.**GammaInverse**(*name*, *a*, *b*)

Create a continuous random variable with an inverse Gamma distribution.

Parameters

a : Real number, $a > 0$, a shape

b : Real number, $b > 0$, a scale

Returns

RandomSymbol

Explanation

The density of the inverse Gamma distribution is given by

$$f(x) := \frac{\beta^\alpha}{\Gamma(\alpha)} x^{-\alpha - 1} \exp\left(\frac{-\beta}{x}\right)$$

with $x > 0$.

Examples

```
>>> from sympy.stats import GammaInverse, density, cdf
>>> from sympy import Symbol, pprint
```

```
>>> a = Symbol("a", positive=True)
>>> b = Symbol("b", positive=True)
>>> z = Symbol("z")
```

```
>>> X = GammaInverse("x", a, b)
```

```
>>> D = density(X)(z)
>>> pprint(D, use_unicode=False)
             -b
             ---
 a   -a - 1   z
b *z       *e
---------------
   Gamma(a)
```

```
>>> cdf(X)(z)
Piecewise((uppergamma(a, b/z)/gamma(a), z > 0), (0, True))
```

References

[R857]

sympy.stats.**Gompertz**(*name*, *b*, *eta*)

Create a Continuous Random Variable with Gompertz distribution.

Parameters

b : Real number, $b > 0$, a scale

eta : Real number, $\eta > 0$, a shape

Returns

RandomSymbol

Explanation

The density of the Gompertz distribution is given by

$$f(x) := b\eta e^{bx} e^{\eta} \exp\left(-\eta e^{bx}\right)$$

with $x \in [0, \infty)$.

Examples

```
>>> from sympy.stats import Gompertz, density
>>> from sympy import Symbol
```

```
>>> b = Symbol("b", positive=True)
>>> eta = Symbol("eta", positive=True)
>>> z = Symbol("z")
```

```
>>> X = Gompertz("x", b, eta)
```

```
>>> density(X)(z)
b*eta*exp(eta)*exp(b*z)*exp(-eta*exp(b*z))
```

References

[R858]

sympy.stats.**Gumbel**(*name*, *beta*, *mu*, *minimum=False*)

Create a Continuous Random Variable with Gumbel distribution.

Parameters

mu : Real number, μ, a location

beta : Real number, $\beta > 0$, a scale

minimum : Boolean, by default `False`, set to `True` for enabling minimum distribution

Returns

RandomSymbol

Explanation

The density of the Gumbel distribution is given by

For Maximum

$$f(x) := \frac{1}{\beta} \exp\left(-\frac{x-\mu}{\beta} - \exp\left(-\frac{x-\mu}{\beta}\right)\right)$$

with $x \in [-\infty, \infty]$.

For Minimum

$$f(x) := \frac{e^{-e^{\frac{-\mu+x}{\beta}} + \frac{-\mu+x}{\beta}}}{\beta}$$

with $x \in [-\infty, \infty]$.

Examples

```
>>> from sympy.stats import Gumbel, density, cdf
>>> from sympy import Symbol
>>> x = Symbol("x")
>>> mu = Symbol("mu")
>>> beta = Symbol("beta", positive=True)
>>> X = Gumbel("x", beta, mu)
>>> density(X)(x)
exp(-exp(-(-mu + x)/beta) - (-mu + x)/beta)/beta
>>> cdf(X)(x)
exp(-exp(-(-mu + x)/beta))
```

References

[R859], [R860], [R861], [R862]

sympy.stats.**Kumaraswamy**(*name*, *a*, *b*)

Create a Continuous Random Variable with a Kumaraswamy distribution.

Parameters

a : Real number, $a > 0$, a shape

b : Real number, $b > 0$, a shape

Returns

RandomSymbol

Explanation

The density of the Kumaraswamy distribution is given by

$$f(x) := abx^{a-1}(1 - x^a)^{b-1}$$

with $x \in [0, 1]$.

Examples

```
>>> from sympy.stats import Kumaraswamy, density, cdf
>>> from sympy import Symbol, pprint
```

```
>>> a = Symbol("a", positive=True)
>>> b = Symbol("b", positive=True)
>>> z = Symbol("z")
```

```
>>> X = Kumaraswamy("x", a, b)
```

```
>>> D = density(X)(z)
>>> pprint(D, use_unicode=False)
                     b - 1
     a - 1 /     a\
a*b*z     *\1 - z /
```

```
>>> cdf(X)(z)
Piecewise((0, z < 0), (1 - (1 - z**a)**b, z <= 1), (1, True))
```

References

[R863]

sympy.stats.**Laplace**(*name*, *mu*, *b*)

Create a continuous random variable with a Laplace distribution.

Parameters

mu : Real number or a list/matrix, the location (mean) or the location vector

b : Real number or a positive definite matrix, representing a scale or the covariance matrix.

Returns

RandomSymbol

Explanation

The density of the Laplace distribution is given by

$$f(x) := \frac{1}{2b}\exp\left(-\frac{|x-\mu|}{b}\right)$$

Examples

```
>>> from sympy.stats import Laplace, density, cdf
>>> from sympy import Symbol, pprint
```

```
>>> mu = Symbol("mu")
>>> b = Symbol("b", positive=True)
>>> z = Symbol("z")
```

```
>>> X = Laplace("x", mu, b)
```

```
>>> density(X)(z)
exp(-Abs(mu - z)/b)/(2*b)
```

```
>>> cdf(X)(z)
Piecewise((exp((-mu + z)/b)/2, mu > z), (1 - exp((mu - z)/b)/2, True))
```

```
>>> L = Laplace('L', [1, 2], [[1, 0], [0, 1]])
>>> pprint(density(L)(1, 2), use_unicode=False)
 5        /     ____\
e *besselk\0, \/ 35 /
---------------------
          pi
```

References

[R864], [R865]

sympy.stats.**Levy**(*name*, *mu*, *c*)

Create a continuous random variable with a Levy distribution.

The density of the Levy distribution is given by

$$f(x) := \sqrt{(}\frac{c}{2\pi})\frac{\exp -\frac{c}{2(x-\mu)}}{(x-\mu)^{3/2}}$$

Parameters

mu : Real number

The location parameter.

c : Real number, $c > 0$

A scale parameter.

Returns

RandomSymbol

Examples

```
>>> from sympy.stats import Levy, density, cdf
>>> from sympy import Symbol
```

```
>>> mu = Symbol("mu", real=True)
>>> c = Symbol("c", positive=True)
>>> z = Symbol("z")
```

```
>>> X = Levy("x", mu, c)
```

```
>>> density(X)(z)
sqrt(2)*sqrt(c)*exp(-c/(-2*mu + 2*z))/(2*sqrt(pi)*(-mu + z)**(3/2))
```

```
>>> cdf(X)(z)
erfc(sqrt(c)*sqrt(1/(-2*mu + 2*z)))
```

References

[R866], [R867]

sympy.stats.**Logistic**(*name*, *mu*, *s*)

Create a continuous random variable with a logistic distribution.

Parameters

mu : Real number, the location (mean)

s : Real number, $s > 0$, a scale

Returns

RandomSymbol

Explanation

The density of the logistic distribution is given by

$$f(x) := \frac{e^{-(x-\mu)/s}}{s\left(1+e^{-(x-\mu)/s}\right)^2}$$

Examples

```
>>> from sympy.stats import Logistic, density, cdf
>>> from sympy import Symbol
```

```
>>> mu = Symbol("mu", real=True)
>>> s = Symbol("s", positive=True)
>>> z = Symbol("z")
```

```
>>> X = Logistic("x", mu, s)
```

```
>>> density(X)(z)
exp((mu - z)/s)/(s*(exp((mu - z)/s) + 1)**2)
```

```
>>> cdf(X)(z)
1/(exp((mu - z)/s) + 1)
```


References

[R868], [R869]

`sympy.stats.LogLogistic(name, alpha, beta)`

Create a continuous random variable with a log-logistic distribution. The distribution is unimodal when `beta > 1`.

Parameters

alpha : Real number, $\alpha > 0$, scale parameter and median of distribution

beta : Real number, $\beta > 0$, a shape parameter

Returns

RandomSymbol

Explanation

The density of the log-logistic distribution is given by

$$f(x) := \frac{(\frac{\beta}{\alpha})(\frac{x}{\alpha})^{\beta-1}}{(1+(\frac{x}{\alpha})^{\beta})^2}$$

Examples

```
>>> from sympy.stats import LogLogistic, density, cdf, quantile
>>> from sympy import Symbol, pprint
```

```
>>> alpha = Symbol("alpha", positive=True)
>>> beta = Symbol("beta", positive=True)
>>> p = Symbol("p")
>>> z = Symbol("z", positive=True)
```

```
>>> X = LogLogistic("x", alpha, beta)
```

```
>>> D = density(X)(z)
>>> pprint(D, use_unicode=False)
              beta - 1
       /  z  \
  beta*|-----|
       \alpha/
------------------------
                       2
      /       beta    \
      |/  z  \        |
alpha*||-----|     + 1|
      \\alpha/        /
```

```
>>> cdf(X)(z)
1/(1 + (z/alpha)**(-beta))
```

```
>>> quantile(X)(p)
alpha*(p/(1 - p))**(1/beta)
```


References

[R870]

sympy.stats.**LogNormal**(*name, mean, std*)

Create a continuous random variable with a log-normal distribution.

Parameters

mu : Real number

The log-scale.

sigma : Real number

A shape. ($\sigma^2 > 0$)

Returns

RandomSymbol

Explanation

The density of the log-normal distribution is given by

$$f(x) := \frac{1}{x\sqrt{2\pi\sigma^2}} e^{-\frac{(\ln x-\mu)^2}{2\sigma^2}}$$

with $x \geq 0$.

Examples

```
>>> from sympy.stats import LogNormal, density
>>> from sympy import Symbol, pprint
```

```
>>> mu = Symbol("mu", real=True)
>>> sigma = Symbol("sigma", positive=True)
>>> z = Symbol("z")
```

```
>>> X = LogNormal("x", mu, sigma)
```

```
>>> D = density(X)(z)
>>> pprint(D, use_unicode=False)
                      2
       -(-mu + log(z))
       -----------------
                   2
           2*sigma
  ___
\/ 2 *e
------------------------

        ____
    2*\/ pi *sigma*z
```

```
>>> X = LogNormal('x', 0, 1) # Mean 0, standard deviation 1
```

```
>>> density(X)(z)
sqrt(2)*exp(-log(z)**2/2)/(2*sqrt(pi)*z)
```

References

[R871], [R872]

sympy.stats.**Lomax**(*name*, *alpha*, *lamda*)

Create a continuous random variable with a Lomax distribution.

Parameters

alpha : Real Number, $\alpha > 0$

Shape parameter

lamda : Real Number, $\lambda > 0$

Scale parameter

Returns
RandomSymbol

Explanation

The density of the Lomax distribution is given by

$$f(x) := \frac{\alpha}{\lambda}\left[1+\frac{x}{\lambda}\right]^{-(\alpha+1)}$$

Examples

```
>>> from sympy.stats import Lomax, density, cdf, E
>>> from sympy import symbols
>>> a, l = symbols('a, l', positive=True)
>>> X = Lomax('X', a, l)
>>> x = symbols('x')
>>> density(X)(x)
a*(1 + x/l)**(-a - 1)/l
>>> cdf(X)(x)
Piecewise((1 - 1/(1 + x/l)**a, x >= 0), (0, True))
>>> a = 2
>>> X = Lomax('X', a, l)
>>> E(X)
l
```

References

[R873]

sympy.stats.**Maxwell**(*name*, *a*)

Create a continuous random variable with a Maxwell distribution.

Parameters
a : Real number, $a > 0$

Returns
RandomSymbol

Explanation

The density of the Maxwell distribution is given by

$$f(x) := \sqrt{\frac{2}{\pi}} \frac{x^2 e^{-x^2/(2a^2)}}{a^3}$$

with $x \geq 0$.

Examples

```
>>> from sympy.stats import Maxwell, density, E, variance
>>> from sympy import Symbol, simplify
```

```
>>> a = Symbol("a", positive=True)
>>> z = Symbol("z")
```

```
>>> X = Maxwell("x", a)
```

```
>>> density(X)(z)
sqrt(2)*z**2*exp(-z**2/(2*a**2))/(sqrt(pi)*a**3)
```

```
>>> E(X)
2*sqrt(2)*a/sqrt(pi)
```

```
>>> simplify(variance(X))
a**2*(-8 + 3*pi)/pi
```

References

[R874], [R875]

sympy.stats.**Moyal**(*name*, *mu*, *sigma*)

Create a continuous random variable with a Moyal distribution.

Parameters

mu : Real number

Location parameter

sigma : Real positive number

Scale parameter

Returns

RandomSymbol

Explanation

The density of the Moyal distribution is given by

$$f(x) := \frac{\exp -\frac{1}{2}\exp -\frac{x-\mu}{\sigma} - \frac{x-\mu}{2\sigma}}{\sqrt{2\pi}\sigma}$$

with $x \in \mathbb{R}$.

Examples

```
>>> from sympy.stats import Moyal, density, cdf
>>> from sympy import Symbol, simplify
>>> mu = Symbol("mu", real=True)
>>> sigma = Symbol("sigma", positive=True, real=True)
>>> z = Symbol("z")
>>> X = Moyal("x", mu, sigma)
>>> density(X)(z)
sqrt(2)*exp(-exp((mu - z)/sigma)/2 - (-mu + z)/(2*sigma))/
↪(2*sqrt(pi)*sigma)
>>> simplify(cdf(X)(z))
1 - erf(sqrt(2)*exp((mu - z)/(2*sigma))/2)
```

References

[R876], [R877]

sympy.stats.**Nakagami**(*name*, *mu*, *omega*)

Create a continuous random variable with a Nakagami distribution.

Parameters

mu : Real number, $\mu \geq \frac{1}{2}$, a shape

omega : Real number, $\omega > 0$, the spread

Returns

RandomSymbol

Explanation

The density of the Nakagami distribution is given by

$$f(x) := \frac{2\mu^\mu}{\Gamma(\mu)\omega^\mu} x^{2\mu-1} \exp\left(-\frac{\mu}{\omega}x^2\right)$$

with $x > 0$.

Examples

```
>>> from sympy.stats import Nakagami, density, E, variance, cdf
>>> from sympy import Symbol, simplify, pprint
```

```
>>> mu = Symbol("mu", positive=True)
>>> omega = Symbol("omega", positive=True)
>>> z = Symbol("z")
```

```
>>> X = Nakagami("x", mu, omega)
```