Génération des variables aléatoires

Génération des variables aléatoires

À partir d'une bonne source de nombres aléatoires, on peut générer des variables aléatoires en utilisant différents techniques:

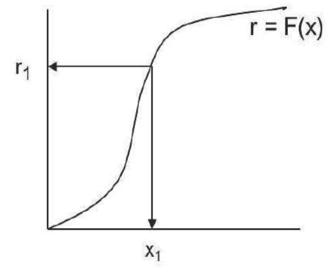
- Méthode de la transformée inverse
- Méthode de convolution
- Méthode de composition
- Méthode de rejet

Méthode de la transformée inverse

- Transformation d'intégrale de probabilité

Si une variable aléatoire X suit une distribution dont la fonction de distribution F est continue et croissante, la variable aléatoire $R = F(X)$ suit une loi uniforme $U(0,1)$.

$$R = F(X) \sim U(0,1)$$



Méthode de la transformée inverse

- Démonstration

Soit X une variable aléatoire.

Si $F(X)$ est une fonction bijective et croissante, pour $r \in (0,1)$

$$\begin{aligned} F_R(r) &= P(R \leq r) = P(F(X) \leq r) \\ &= P\left(F^{-1}(F(X)) \leq F^{-1}(r)\right) \\ &= P\left(X \leq F^{-1}(r)\right) = F\left(F^{-1}(r)\right) \\ &= r \end{aligned}$$

La cdf de R est celle de la loi $U(0,1)$.

Méthode de la transformée inverse

- Remarque : $F(X)$ peut ne pas être 1-1 ou strictement croissante, i.e. il existe un intervalle (a,b) avec $0 \leq a \leq b$, tel que $P\big(X \in (a,b)\big) = 0$.
- Soit

$$G(r) = \min\{x | r \leq F(x)\}$$

et $G(r)$ est non-décroissante.

- Alors

$$\begin{aligned} F_R(r) &= P(R \leq r) = P(F(X) \leq r) \\ &= P(X \leq \min\{x | r \leq F(x)\}) \\ &= P\big(X \leq G(r)\big) = r \end{aligned}$$

Méthode de la transformée inverse

Génération de X avec fonction de répartition $F(x)$

- Algorithme:

1. Générer $U \sim Uniforme(0,1)$
2. Retourner $X = F^{-1}(U)$

- Remarque

Si F n'est pas continue ou strictement croissante (non bijective), on pourra utiliser la fonction inverse généralisée

$$X = F^{-1}(U) = \min\{x : F(x) \geq U\}$$

Méthode de la transformée inverse: distribution triangulaire

- Soit X une variable aléatoire suivant une distribution triangulaire à droit $\mathrm{T}ri(0,1,0)$.
- Sa fonction de densité s'écrit

$$f(x) = \begin{cases} -2(1-x), & 0 \le x \le 1 \\ 0, & sinon \end{cases}$$

- Et sa fonction de distribution

$$F(x) = \begin{cases} 0 & x < 0 \\ 2x - x^2 & 0 \le x \le 1 \\ 1 & x > 1. \end{cases}$$

Méthode de la transformée inverse: distribution triangulaire

$$F(x) = \begin{cases} 0, & x < 0 \\ 1 - (1-x)^2, & 0 \le x \le 1 \\ 1, & x > 1 \end{cases}$$

En considérant $F(x) = u$, on a

$$F(x) = u$$
$$1 - (1-x)^2 = u$$
$$(1-x)^2 = 1 - u$$
$$1 - x = \sqrt{1-u}$$
$$X = 1 - \sqrt{1-u}$$

Méthode de la transformée inverse: distribution triangulaire

- Fonction de répartition inverse

$$F^{-1}(x) = 1 - \sqrt{1-x}$$

- Algorithme

1. Générer
$$U \sim U(0,1)$$
2. Retourner
$$\mathrm{X} = F^{-1}(U) = 1 - \sqrt{1-U}$$

Méthode de la transformée inverse: distribution exponentielle

- Distribution exponentielle

$$X \sim Exp(\lambda)$$

- Fonction de densité

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & si \;\; x \ge 0 \\ 0 & si \; x < 0 \end{cases}$$

- Fonction de répartition

$$F(x) = \begin{cases} 1 - e^{-\lambda x} & si \;\; x \ge 0 \\ 0 & si \; x < 0 \end{cases}$$

Méthode de transformation inverse: distribution exponentielle

En considérant $F(x) = u$, on a

$$u = 1 - e^{-\lambda x}$$

$$1 - u = e^{-\lambda x}$$

$$\ln(1-u) = -\lambda x$$

$$x = -\frac{\ln(1-u)}{\lambda}$$

Méthode de transformation inverse: distribution exponentielle

- Fonction de répartition inverse

$$F^{-1}(x) = -\frac{\ln(1-x)}{\lambda} \qquad 0 \le x \le 1$$

- Algorithme

1. Générer
$$U \sim U(0,1)$$
2. Retourner X
$$X = -\frac{\ln(U)}{\lambda} \sim Exp\,(\lambda)$$

Méthode de transformation inverse: distribution de Weibull

- Distribution de Weibull

$$X \sim Weibull(k,\lambda) \qquad k,\lambda > 0$$

- Fonction de densité

$$f(x) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-\left(\frac{x}{\lambda}\right)^k} & si\ \ x \geq 0 \\ 0 & si\ x < 0 \end{cases}$$

- Fonction de répartition

$$F(x) = \begin{cases} 1 - e^{-\left(\frac{x}{\lambda}\right)^k} & si\ \ x \geq 0 \\ 0 & si\ x < 0 \end{cases}$$

Méthode de la transformée inverse: distribution de Weibull

- Fonction de répartition inverse

$$F^{-1}(x) = \frac{(-\ln(U))^{\frac{1}{k}}}{\lambda} \qquad 0 \leq x \leq 1$$

- Algorithme

1. Générer

$$U \sim U(0,1)$$

2. Retourner X

$$X = \frac{(-\ln(U))^{\frac{1}{a}}}{\lambda} \sim Weibull(a,\lambda)$$

Méthode de la transformée inverse: distributions discrètes

- Condition d'application:

Toutes les lois peuvent être générées par la méthode de la transformée inverse.

- Exemples
  - Distributions empiriques
  - Loi uniforme discrète
  - Loi de Poisson
  - …

Méthode de la transformée inverse: distributions discrètes

Considérons X une variable aléatoire avec probabilité

$$P(X = x_i) = p_i, \qquad i = 0,1,\ldots.. \quad \sum_{i=0}^{\infty} p_i = 1.$$

- Algorithme:

1. Générer

$$U \sim U(0,1)$$

2. Retourner $X = x_i$ si

$$\sum_{j=0}^{i-1} p_j \leq U < \sum_{j=0}^{i} p_j$$

Méthode de la transformée inverse: distributions discrètes

Considérons X une variable aléatoire avec probabilité

$$P(X = x_i) = p_i, \qquad i = 0,1,\ldots.. \quad \sum_{i=0}^{\infty} p_i = 1.$$

- Algorithme:

1. Générer

$$U \sim U(0,1)$$

2. Retourner

$$X = F^{-1}(U) = \min\{x: F(x) \geq U\}$$

Méthode de la transformée inverse: Loi de Bernoulli

- Distribution de Bernoulli

$$X \sim B(p)$$

- Fonction de masse

$$P(X = 1) = p \qquad P(X = 0) = 1 - p$$

$$P(X = i) = p^i(1-p)^{1-i}$$

- Fonction de répartition

$$F(x) = \begin{cases} 0 & x < 0 \\ 1 - p & 0 \leq x < 1 \\ 1 & x \geq 1 \end{cases}$$

Méthode de la transformée inverse: Loi de Bernoulli

- Fonction de répartition inverse

$$F^{-1}(x) = \begin{cases} 1 & si\ x \le p \\ 0 & si\ x > p \end{cases}$$

- Algorithme

1. Générer

$$U \sim U(0,1)$$

2. Si $U \le p$, alors retourner $X = 1$; sinon, retourner $X = 0$.

Méthode de la transformée inverse: Loi uniforme discrète

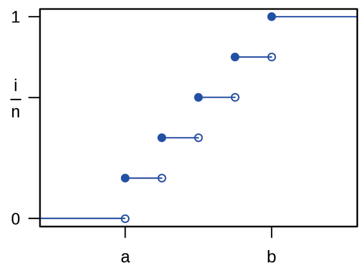
- Distribution uniforme discrète
- Fonction de masse

Pour $i = a, a+1, \ldots, b$

$$P(X = i) = \frac{1}{b - a + 1}$$

- Fonction de répartition

$$F(x) = \begin{cases} 1 & x > b \\ \frac{\lfloor x \rfloor - a + 1}{b - a + 1} & a \le x \le b \\ 0 & x < a \end{cases}$$



Méthode de la transformée inverse: Loi uniforme discrète

- Fonction de répartition inverse

$$F^{-1}(x) = a + \lfloor (b - a + 1)x \rfloor \qquad 0 \le x \le 1$$

- Algorithme

1. Générer

$$U \sim U(0,1)$$

2. Retourner X

$$X = a + \lfloor (b - a + 1)U \rfloor$$

Méthode de la transformée inverse: Loi géométrique

- Loi géométrique

$$X \sim Geométrique(p)$$

- Fonction de masse

$$P(X = i) = p(1-p)^i, \qquad i = 0,1,2,\ldots$$

- Fonction de répartition

$$F(x) = \sum_{i=0}^{x} p(1-p)^i = 1 - (1-p)^{x+1}$$

Méthode de la transformée inverse: Loi géométrique

$$F(x-1) < r \le F(x)$$

$$1 - (1-p)^x < r \le 1 - (1-p)^{x+1}$$

$$(1-p)^x < 1 - r \le (1-p)^{x+1}$$

$$xln(1-p) < \ln(1-r) \le (x+1)ln(1-p)$$

$$\frac{\ln(1-r)}{\ln(1-p)} - 1 \le x < \frac{\ln(1-r)}{\ln(1-p)}$$

Méthode de la transformée inverse: Loi géométrique

- Fonction de répartition inverse

$$F^{-1}(x) = \left\lceil \frac{\ln(1-x)}{\ln(1-p)} - 1 \right\rceil$$

- Algorithme

1. Générer $U \sim U(0,1)$
2. Retourner $X = \lfloor \ln(U) \,/\, \ln(1-p) \rfloor$

- Algorithme

1. Générer indépendamment et successivement variables aléatoires $Y_1, Y_2, \ldots \sim Bernoulli(p)$ jusqu'à l'arrivée de la première $Y_I = 1$
2. Retourner $X = I - 1$

Méthode de la transformée inverse

- Problème

L'expression de la fonction inverse de F n'est pas toujours facile à obtenir.

Méthode de composition

- Condition d'application:

Si la fonction de distribution F peut être exprimée ainsi

$$F(x) = \sum_{i=1}^{\infty} p_i F_i(x)$$

où $F_1, F_2, \ldots$ sont des fonctions de distributions et

$$p_i \geq 0$$

et

$$\sum_{i=1}^{\infty} p_i = 1$$

Méthode de composition

- Algorithme:

1. Générer un indexe I tel que, $i = 1,2, \ldots$.

$$P(I = i) = p_i$$

2. Générer X avec la fonction de distribution F_I

Méthode de composition Distribution hyper-exponentielle

- Distribution hyper-exponentielle

$$F(x) = \begin{cases} 1 - \sum_{i=1}^{k} p_i e^{-\lambda_i x} & si\ x \geq 0 \\ 0 & sinon \end{cases}$$

$$= p_1 F_1(x) + p_2 F_2(x) + \cdots + p_k F_k(x), \qquad x \geq 0$$

où $F_i(x)$ est une fonction de distribution exponentielle avec paramètre $\mu_i, i = 1, \ldots, k$, i.e.

$$F_i(x) = 1 - e^{-\lambda_i x}$$

Méthode de composition Distribution hyper-exponentielle

- Algorithme:

1. Générer un indexe I tel que

$$P(I = i) = p_i$$

2. Générer

$$X \sim Exp(\mu_I)$$

3. Retourner X

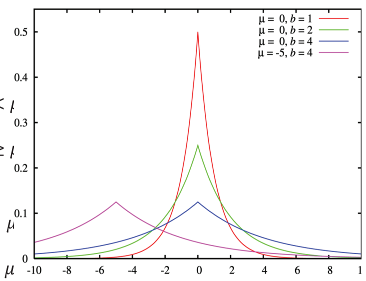
Méthode de composition distribution de Laplace

- Distribution de Laplace (ou double-exponentielle)
- Fonction de densité

$$f(x) = \frac{1}{2b} \begin{cases} \frac{1}{2} \exp\left(-\frac{\mu - x}{b}\right) & si\ x < \mu \\ \frac{1}{2} \exp\left(-\frac{x - \mu}{b}\right) & si\ x \geq \mu \end{cases}$$

- Fonction de répartition

$$F(x) = \begin{cases} \frac{1}{2} \exp\left(\frac{x - \mu}{b}\right) & si\ x < \mu \\ 1 - \frac{1}{2} \exp\left(\frac{x - \mu}{b}\right) & si\ x \geq \mu \end{cases}$$



Méthode de composition
distribution de Laplace

- Algorithme:

1. Générer un indexe I tel que , $i = 1\ ou\ 2$
$$P(I = i) = \frac{1}{2}$$
2. Générer Y
$$Y \sim Exp(\mu_I)$$
3. Retourner X
$$X = \mu + (-1)^I . Y$$

Méthode de convolution

- Condition d'application

Si une variable aléatoire X peut être exprimée par la somme des variables aléatoires indépendantes $Y_1, Y_2, \dots, Y_n$, (convolution), alors
$$X = Y_1 + Y_2 + \cdots Y_n$$
où la génération (directe) de Y_i est plus facile que celle de X.

Méthode de convolution

- Algorithme:

1. Générer indépendamment
$$Y_1 \sim G_1$$
$$Y_2 \sim G_2$$
$$\dots$$
$$Y_n \sim G_n$$
2. Retourner
$$X = Y_1 + Y_2 + \cdots Y_n$$

Méthode de convolution
Loi binomiale

- Loi binomiale
$$X \sim Binom(n, p)$$
- Fonction de masse
$$P(X = i) = \binom{n}{i} p^i (1-p)^{n-i} \qquad i = 0,1, \dots, n$$
- Propriété : par définition, X est la somme des n variables aléatoires indépendantes de distribution de Bernoulli avec la probabilité de succès p
- Algorithme

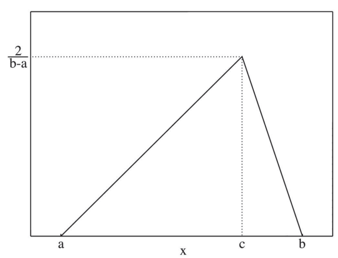
1. Générer $Y_1, \dots, Y_n \sim Bernoulli(p)$
2. Retourner $X = Y_1 + \cdots + Y_n$

Méthode de convolution
distribution triangulaire

- Distribution triangulaire

Soient U_1 et U_2 deux variables aléatoire indépendantes et suivant la distribution uniforme $U(0,1)$.

On a
$$X = U_1 + U_2 \sim Triangulaire(0,2,1)$$
et
$$Y = U_1 + U_2 - 1 \sim Triangulaire(-1,1,0)$$



Méthode de convolution
distribution triangulaire

- Algorithme:

1. Générer indépendamment
$$U_1 \sim U(0,1)$$
$$U_2 \sim U(0,1)$$
2. Retourner
$$X = U_1 + U_2 \sim Tri\ (0,2,1)$$

Méthode de convolution
Distribution d'Erlang

- Composition

$$X = Y_1 + Y_2 + \cdots + Y_n \sim Erlang(n, \mu)$$

où

$Y_1, Y_2, \ldots, Y_n$ sont les variables aléatoires indépendantes et suivant une distribution exponentielle $Exp(\mu)$

- Algorithme:

1. Générer indépendamment

$$Y_1, Y_2, \ldots, Y_n \sim Exp(1/\mu)$$

2. Retourner

$$X = Y_1 + Y_2 + \cdots + Y_n$$

Méthode de rejet

- Objectif: générer X ayant une densité f.

- Méthode:

S'il existe une fonction de g tel que quelque soit x,

$$g(x) \geq f(x).$$

Alors $h(x) = \frac{g(x)}{c}$ est une fonction de densité avec

$$c = \int_{-\infty}^{\infty} g(x)dx < \infty$$

Méthode de rejet

- Algorithme

1. Générer Y selon densité $h(x)$
2. Générer $U \sim U(0{,}1)$ indépendamment de Y
3. Si

$$U \leq \frac{f(Y)}{g(Y)},$$

alors retourner $X = Y$;

sinon, retourner à l'étape 1

Méthode de rejet

- Remarque

$$P(X \leq x) = P(Y \leq x | Y\ accepté)$$

- Validation

$$P(Y \leq x, Y\ accepté) = \int_{-\infty}^{x} \frac{f(x)}{g(x)} h(y)dy = \frac{1}{c}\int_{-\infty}^{x} f(y)dy$$

Quand $x \to \infty$,

$$P(Y\ accepté) = \frac{1}{c}$$

Donc

$$P(X \leq x) = \frac{P(Y \leq x, Y\ accepté)}{P(Y\ accepté)} = \int_{-\infty}^{x} f(y)dy$$

Méthode de rejet

- Remarque

Le nombre d'itérations est géométriquement distribué avec une moyenne c.

- Choisir g tel que la variable Y peut être générée facilement

1. La probabilité de rejet dans l'étape 3 doit être petit
2. Comment choisir la fonction g ?
   - Choisir un c proche de 1,
   - g doit être proche de f

Méthode de rejet: distribution de beta

- La densité de distribution $Beta(4{,}3)$

$$f(x) = 60x^3(1-x)^2 \qquad 0 \leq x \leq 1$$

On a $argmax_x f(x) = 0{,}6$

$$f(0{,}6) = 2{,}0736$$

On définit

$$g(x) = 2{,}0736 \qquad 0 \leq x \leq 1$$

Alors

$$g(x) \geq f(x)$$

Méthode de rejet: distribution de beta

- Algorithme:

1. Générer Y et U selon $U(0,1)$
2. Si

$$U \leq \frac{60Y^3(1-Y)^2}{2{,}0736}$$

alors retourner $X = Y$;
sinon, retourner à l'étape 1.

Méthode de rejet: distribution normale

Si $X \sim N(0,1)$ on a la densité de $|X|$

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

Alors

$$g(x) = \sqrt{2e/\pi} e^{-x} \geq f(x)$$

Méthode de rejet: distribution normale

- Algorithme

1. Générer $Y \sim Exp(1)$
2. Générer $U \sim U(0,1)$ indépendante de Y
3. Si

$$U \leq e^{-(Y-1)^2/2}$$

retourner $X = Y$ ou $X = -Y$ avec probabilité $\frac{1}{2}$:

3.1 Générer $R \sim U(0,1)$

3.2 Si $R \leq \frac{1}{2}$, retourner $X = Y$; sinon retourner $X = -Y$

sinon, retourner à l'étape 1.

Fonctions intégrées random

random.triangular(low, high, mode)
random.betavariate(alpha, beta)
random.betavariate(alpha, beta)
random.gammavariate(alpha, beta)
random.gauss(mu, sigma)

Fonctions intégrées random

random.lognormvariate(mu, sigma)
random.normalvariate(mu, sigma)
random.vonmisesvariate(mu, kappa)
random.paretovariate(alpha)
random.weibullvariate(alpha, beta)

Fonctions intégrées numpy.random

beta(a, b[, size]) Draw samples from a Beta distribution.
binomial(n, p[, size]) Draw samples from a binomial distribution.
chisquare(df[, size]) Draw samples from a chi-square distribution.
dirichlet(alpha[, size]) Draw samples from the Dirichlet distribution.
exponential([scale, size]) Draw samples from an exponential distribution.
f(dfnum, dfden[, size]) Draw samples from an F distribution.
gamma(shape[, scale, size]) Draw samples from a Gamma distribution.

Fonctions intégrées numpy.random

geometric(p[, size]) Draw samples from the geometric distribution.

gumbel([loc, scale, size]) Draw samples from a Gumbel distribution.

hypergeometric(ngood, nbad, nsample[, size]) Draw samples from a Hypergeometric distribution.

laplace([loc, scale, size]) Draw samples from the Laplace or double exponential distribution with specified location (or mean) and scale (decay).

logistic([loc, scale, size]) Draw samples from a logistic distribution.

lognormal([mean, sigma, size]) Draw samples from a log-normal distribution.

logseries(p[, size]) Draw samples from a logarithmic series distribution.

Fonctions intégrées numpy.random

multinomial(n, pvals[, size]) Draw samples from a multinomial distribution.

multivariate_normal(mean, cov[, size, ...) Draw random samples from a multivariate normal distribution.

negative_binomial(n, p[, size]) Draw samples from a negative binomial distribution.

noncentral_chisquare(df, nonc[, size]) Draw samples from a noncentral chi-square distribution.

noncentral_f(dfnum, dfden, nonc[, size]) Draw samples from the noncentral F distribution.

normal([loc, scale, size]) Draw random samples from a normal (Gaussian) distribution.

Fonctions intégrées numpy.random

pareto(a[, size]) Draw samples from a Pareto II or Lomax distribution with specified shape.

poisson([lam, size]) Draw samples from a Poisson distribution.

power(a[, size]) Draws samples in [0, 1] from a power distribution with positive exponent a - 1.

rayleigh([scale, size]) Draw samples from a Rayleigh distribution.

standard_cauchy([size]) Draw samples from a standard Cauchy distribution with mode = 0.

standard_exponential([size]) Draw samples from the standard exponential distribution.

Fonctions intégrées numpy.random

- standard_gamma(shape[, size]) Draw samples from a standard Gamma distribution.
- standard_normal([size]) Draw samples from a standard Normal distribution (mean=0, stdev=1).
- standard_t(df[, size]) Draw samples from a standard Student’s t distribution with df degrees of freedom.
- triangular(left, mode, right[, size]) Draw samples from the triangular distribution over the interval [left, right].
- uniform([low, high, size]) Draw samples from a uniform distribution.

Fonctions intégrées numpy.random

vonmises(mu, kappa[, size]) Draw samples from a von Mises distribution.

wald(mean, scale[, size]) Draw samples from a Wald, or inverse Gaussian, distribution.

weibull(a[, size]) Draw samples from a Weibull distribution.

zipf(a[, size]) Draw samples from a Zipf distribution.

Fonctions intégrées scipy.stats: lois continues

rvs(*args, **kwds) Random variates of given type.

pdf(x, *args, **kwds) Probability density function at x of the given RV.

logpdf(x, *args, **kwds) Log of the probability density function at x of the given RV.

cdf(x, *args, **kwds) Cumulative distribution function of the given RV.

logcdf(x, *args, **kwds) Log of the cumulative distribution function at x of the given RV.

sf(x, *args, **kwds) Survival function (1 - cdf) at x of the given RV.

logsf(x, *args, **kwds) Log of the survival function of the given RV.

Fonctions intégrées
scipy.stats: lois continues

ppf(q, *args, **kwds)   Percent point function (inverse of cdf) at q of the given RV.

isf(q, *args, **kwds)   Inverse survival function (inverse of sf) at q of the given RV.

moment(n, *args, **kwds)   n-th order non-central moment of distribution.

stats(*args, **kwds)   Some statistics of the given RV.

entropy(*args, **kwds)   Differential entropy of the RV.

expect([func, args, loc, scale, lb, ub, ...])   Calculate expected value of a function with respect to the distribution.

median(*args, **kwds)   Median of the distribution.

Fonctions intégrées
scipy.stats: lois continues

mean(*args, **kwds)   Mean of the distribution.

std(*args, **kwds)   Standard deviation of the distribution.

var(*args, **kwds)   Variance of the distribution.

interval(alpha, *args, **kwds)   Confidence interval with equal areas around the median.

__call__(*args, **kwds)   Freeze the distribution for the given arguments.

fit(data, *args, **kwds)   Return MLEs for shape (if applicable), location, and scale parameters from data.

fit_loc_scale(data, *args)   Estimate loc and scale parameters from data using 1st and 2nd moments.

nnlf(theta, x)   Return negative loglikelihood function.

Fonctions intégrées
scipy.stats: lois discrètes

rvs(*args, **kwargs)   Random variates of given type.

pmf(k, *args, **kwds)   Probability mass function at k of the given RV.

logpmf(k, *args, **kwds)   Log of the probability mass function at k of the given RV.

cdf(k, *args, **kwds)   Cumulative distribution function of the given RV.

logcdf(k, *args, **kwds)   Log of the cumulative distribution function at k of the given RV.

sf(k, *args, **kwds)   Survival function (1 - cdf) at k of the given RV.

Fonctions intégrées
scipy.stats: lois discrètes

logsf(k, *args, **kwds)   Log of the survival function of the given RV.

ppf(q, *args, **kwds)   Percent point function (inverse of cdf) at q of the given RV.

isf(q, *args, **kwds)   Inverse survival function (inverse of sf) at q of the given RV.

moment(n, *args, **kwds)   n-th order non-central moment of distribution.

stats(*args, **kwds)   Some statistics of the given RV.

entropy(*args, **kwds)   Differential entropy of the RV.

expect([func, args, loc, lb, ub, ...])   Calculate expected value of a function with respect to the distribution for discrete distribution.

Fonctions intégrées
scipy.stats: lois discrètes

median(*args, **kwds)   Median of the distribution.

mean(*args, **kwds)   Mean of the distribution.

std(*args, **kwds)   Standard deviation of the distribution.

var(*args, **kwds)   Variance of the distribution.

interval(alpha, *args, **kwds)   Confidence interval with equal areas around the median.

__call__(*args, **kwds)   Freeze the distribution for the given arguments.