BINOMIAL DISTRIBUTION

Cumulative Binomial Probability

public static double binomalCDF(**double** p, **int** n, **int** k)

Usage: **bp = Stat.binomalCDF(p, n, k);**

Returns the cumulative binomial probability, *P*, of an event, which occurs with a probability, *p*, occuring *k* or more times in *n* trials:

$$P = \sum_{j=k}^{n} \frac{n!}{j!(n-j)!} p^j (1-p)^{n-j}$$

A regularised incomplete beta function is used to calculate *P*

$$P = I_p(k, n-k+1)$$

Binomial Probability Mass Function

public static double binomialPDF(**double** p, **int** n, **int** k)

Usage: **bpmf = Stat.binomialPDF(p, n, k);**

Returns the binomial mass probability function, $p(j|n,p)$, idefined as:

$$f(j|n,p) = \frac{n!}{j!(n-j)!} p^j (1-p)^{n-j}$$

where *j* is the number of successes in *n* trials of a Bernoulli process with probability of success *p*.

Generation of Binomial random deviates

NB. A more extensive set of methods concerning the generation of this deviate may be found in the class PsRandom.

public double[ ] binomialRand(**double** prob, **int** nTrials, **int** nArray)

public double[ ] binomialRand(**double** prob, **int** nTrials, **int** nArray, **long** seed)

Usage: **deviates = Stat.binomialRand(prob, nTrials, nArray);**

Returns an array, of length **nArray**, of integer random deviates (returned as an array of doubles) drawn from a Binomial distribution of *n* [**nTrials**] each of probabilty *p* [**prob**].

The Java class, **Random**, is used, as the source of the generator's required standard uniform random deviates, taking the initial seed from the clock.

Usage: **deviates = Stat.binomialRand(prob, nTrials, nArray, seed);**

Returns an array, of length **nArray**, of integer random deviates (returned as an array of doubles) drawn from a Binomial distribution of *n* [**nTrials**] each of probabilty *p* [**prob**].

The Java class, **Random**, is used, as the source of the generator's required standard uniform random deviates, using a user supplied seed.

Binomial Coefficient

public static double binomialCoeff(**int** n, **int** k)

Usage: **bc = Stat.binomialCoeff(n, k);**

Returns the binomial coefficient, n!/(k!(n-k)!), for 0<=k<=n, as a double.

If a series of related binomial coefficients are required the recurrence relationships

coeff(n+1, k) = coeff(n, k)*(n+1)/(n-k+1) = coeff(n, k) + coeff(n, k-1)

coeff(n, k+1) = coeff(n, k)*(n-k)/(k+1)

may be useful.