See Also

The **BinarySearch** and **FindLevel** operations. See **Indexing and Subranges** on page II-76.

binomial

binomial(*n*, *k*)

The binomial function returns the ratio:

$$\frac{n!}{k!(n-k)!}$$

It is assumed that n and k are integers and $0 \le k \le n$ and ! denotes the factorial function.

Note that although the binomial function is an integer-valued function, a double-precision number has 53 bits for the mantissa. This means that numbers over 2^{52} (about 4.5×10^{15}) will be accurate to about one part in 2×10^{16}.

If you encounter overflow when the arguments are large you can use **APMath** or **binomialln**. For example:

```
• Print binomial(2800,333)
  inf
• APMath/V result = binomial(2800,333)
  9.00198266850214464998502850373044733821917603122330E+441
• Print/D binomialln(2800,333)
  1017.63747085995
• APMath/V result = log(binomial(2800,333))
  1.01763747085994889343514158007045064106357813268781E+3
```

binomialln

binomialln(*a*, *b*)

The binomialln function returns the natural log of the binomial coefficient for a and b.

$$\text{binomialln}(a,b) = \ln(a!) - \ln(b!) - \ln((a-b)!)$$

See Also

Chapter III-12, **Statistics** for an overview of the various functions and operations; **binomial**, **StatsBinomialPDF**, **StatsBinomialCDF**, and **StatsInvBinomialCDF**.

binomialNoise

binomialNoise(*n*, *p*)

The binomialNoise function returns a pseudo-random value from the binomial distribution

$$f(x) = \begin{pmatrix} n \\ x \end{pmatrix} p^x (1-p)^{n-x}, \qquad \begin{matrix} 0 \le p \le 1 \\ x = 0,1,2...n \end{matrix}$$

whose mean is np and variance is $np(1-p)$.

When n is large such that p^n is zero to machine accuracy the function returns NaN. When n is large such that $np(1-p)>5$ and $0.1<p<0.9$ you can replace the binomial variate with a normal variate with mean np and standard deviation sqrt($n*p*(1-p)$).

The random number generator initializes using the system clock when Igor Pro starts. This almost guarantees that you will never repeat the same sequence. For repeatable "random" numbers, use **SetRandomSeed**. The algorithm uses the Mersenne Twister random number generator.

See Also

The **SetRandomSeed** operation.

Noise Functions on page III-390.

Chapter III-12, **Statistics** for an overview of the various functions and operations.