

---

## **AIMMS Function Reference - Distribution and Combinatoric Functions**

This file contains only one chapter of the book. For a free download of the complete book in pdf format, please visit [www.aimms.com](http://www.aimms.com)

# Distribution and Combinatoric Functions

AIMMS supports several functions to obtain random numbers from discrete or continuous distribution, and additionally some combinatoric functions.

The functions for discrete distributions are:

- `Binomial`
- `Geometric`
- `HyperGeometric`
- `NegativeBinomial`
- `Poisson`

The functions for continuous distributions are:

- `Beta`
- `Exponential`
- `ExtremeValue`
- `Gamma`
- `Logistic`
- `LogNormal`
- `Normal`
- `Pareto`
- `Triangular`
- `Uniform`
- `Weibull`

The following functions that operate on distributions are available:

- `DistributionCumulative`
- `DistributionInverseCumulative`
- `DistributionDensity`
- `DistributionInverseDensity`
- `DistributionMean`
- `DistributionDeviation`
- `DistributionVariance`
- `DistributionSkewness`
- `DistributionKurtosis`

The combinatoric functions are:

- `Combination`
- `Factorial`
- `Permutation`

---

## Binomial

The function `Binomial` draws a random value from a binomial distribution.

```
Binomial(  
    ProbabilityOfSuccess, ! (input) numerical expression  
    NumberOfTries        ! (input) integer expression  
)
```

### Arguments:

*ProbabilityOfSuccess*

A scalar numerical expression in range (0, 1).

*NumberOfTries*

An integer numerical expression > 0.

### Return value:

The function `Binomial` returns a random value drawn from a binomial distribution with a probability of success *ProbabilityOfSuccess* and number of tries *NumberOfTries*

### See also:

The Binomial distribution is discussed in full detail in [Appendix A](#) of the Language Reference.

---

## Geometric

The function `Geometric` draws a random value from a geometric distribution.

```
Geometric(  
    ProbabilityOfSuccess ! (input) numerical expression  
)
```

### Arguments:

*ProbabilityOfSuccess*

A scalar numerical expression in the range (0, 1).

### Return value:

The function `Geometric` returns a random value drawn from a geometric distribution with a probability of success *ProbabilityOfSuccess*.

### See also:

The Geometric distribution is discussed in full detail in [Appendix A](#) of the Language Reference.

---

## HyperGeometric

The function `HyperGeometric` draws a random value from a hypergeometric distribution.

```
HyperGeometric(  
  ProbabilityOfSuccess, ! (input) numerical expression  
  NumberOfTries,       ! (input) integer expression  
  PopulationSize       ! (input) integer expression  
)
```

### Arguments:

*ProbabilityOfSuccess*

A scalar numerical expression in the range  $(0, 1)$ .

*NumberOfTries*

A integer numerical expression in the range  $1, \dots, PopulationSize$ .

*PopulationSize*

A integer numerical expression  $> 0$ .

### Return value:

The function `HyperGeometric` returns a random value drawn from a hypergeometric distribution with a probability of success *ProbabilityOfSuccess*, number of tries *NumberOfTries* and population size *PopulationSize*.

### Remarks:

The probability of success *ProbabilityOfSuccess* must assume one of the values  $i/size$ , where  $i$  is in the range  $1, \dots, PopulationSize - 1$ .

### See also:

The HyperGeometric distribution is discussed in full detail in [Appendix A](#) of the Language Reference.

---

## NegativeBinomial

The function `NegativeBinomial` draws a random value from a negative binomial distribution.

```
NegativeBinomial(  
  ProbabilityOfSuccess, ! (input) numerical expression  
  NumberOfSuccesses    ! (input) integer expression  
)
```

### Arguments:

*ProbabilityOfSuccess*

A scalar numerical expression in the range (0, 1).

*NumberOfSuccesses*

A integer numerical expression  $> 0$ .

### Return value:

The function `NegativeBinomial` returns a random value drawn from a negative binomial distribution with probability *ProbabilityOfSuccess* and number of successes *NumberOfSuccesses*.

### See also:

The `NegativeBinomial` distribution is discussed in full detail in [Appendix A](#) of the Language Reference.

---

## Poisson

The function `Poisson` draws a random value from a Poisson distribution.

```
Poisson(  
    AverageNumberOfSuccesses    ! (input) numerical expression  
)
```

### Arguments:

*lambda*

A scalar numerical expression  $> 0$ .

### Return value:

The function `Poisson` returns a random value drawn from a Poisson distribution with average number of occurrences *AverageNumberOfSuccesses*.

### See also:

The Poisson distribution is discussed in full detail in [Appendix A](#) of the Language Reference.

---

## Beta

The function Beta draws a random value from a beta distribution.

```
Beta(  
  ShapeAlpha,      ! (input) numerical expression  
  ShapeBeta,       ! (input) numerical expression  
  Minimum,         ! (optional) numerical expression  
  Maximum          ! (optional) numerical expression  
)
```

### Arguments:

*ShapeAlpha*

A scalar numerical expression  $> 0$ .

*ShapeBeta*

A scalar numerical expression  $> 0$ .

*Minimum*

A scalar numerical expression.

*Maximum*

A scalar numerical expression  $> min$ .

### Return value:

The function Beta returns a random value drawn from a beta distribution with shapes *ShapeAlpha*, *ShapeBeta*, lower bound *Minimum* and upper bound *Maximum*.

### Remarks:

The prototype of this function has changed with the introduction of AIMMS 3.4. In order to run models that still use the original prototype, the option `Distribution_compatibility` should be set to `Aimms_3_0`. The original function `Beta(ShapeAlpha, ShapeBeta, s)` returns a random value drawn from a beta distribution with shapes *ShapeAlpha*, *ShapeBeta* and scale *s*, where  $s = Maximum$  and  $Minimum = 0$ .

### See also:

The Beta distribution is discussed in full detail in [Appendix A](#) of the Language Reference.

---

## Exponential

The function `Exponential` draws a random value from an exponential distribution.

```
Exponential(  
  lowerbound    ! (optional) numerical expression  
  scale         ! (optional) numerical expression  
)
```

### Arguments:

*lowerbound*

A scalar numerical expression.

*scale*

A scalar numerical expression  $> 0$ .

### Return value:

The function `Exponential` returns a random value drawn from a exponential distribution with lower bound *lowerbound* and scale *scale*.

### Remarks:

The prototype of this function has changed with the introduction of AIMMS 3.4. In order to run models that still use the original prototype, the option `Distribution_compatibility` should be set to `Aimms_3_0`. The original function `Exponential(lambda)` returns a random value drawn from a exponential distribution with rate  $lambda = 1/scale$  and lower bound 0.

### See also:

The `Exponential` distribution is discussed in full detail in [Appendix A](#) of the Language Reference.

---

## ExtremeValue

The function `ExtremeValue` draws a random value from an extreme value distribution.

```
ExtremeValue(  
    location,    ! (optional) numerical expression  
    scale       ! (optional) numerical expression  
)
```

### Arguments:

*location*

A scalar numerical expression.

*scale*

A scalar numerical expression  $> 0$ .

### Return value:

The function `ExtremeValue` returns a random value drawn from an extreme value distribution with location *location* and scale *scale*.

### See also:

The `ExtremeValue` distribution is discussed in full detail in [Appendix A](#) of the Language Reference.

---

## Gamma

The function `Gamma` draws a random value from a gamma distribution.

```
Gamma(  
  Shape,      ! (input) numerical expression  
  Lowerbound, ! (optional) numerical expression  
  Scale      ! (optional) numerical expression  
)
```

### Arguments:

#### *Shape*

A scalar numerical expression  $> 0$ .

#### *Lowerbound*

A scalar numerical expression  $> 0$ .

#### *Scale*

A scalar numerical expression  $> 0$ .

### Return value:

The function `Gamma` returns a random value drawn from a gamma distribution with shape *Shape*, lower bound *Lowerbound* and scale *Scale*.

### Remarks:

The prototype of this function has changed with the introduction of AIMMS 3.4. In order to run models that still use the original prototype, the option `Distribution_compatibility` should be set to `Aimms_3_0`. The original function `Gamma(alpha, Shape)` returns a random value drawn from a gamma distribution with rate  $alpha = 1/Scale$ , shape *Shape* and lower bound 0.

### See also:

The Gamma distribution is discussed in full detail in [Appendix A](#) of the Language Reference.

---

## Logistic

The function `Logistic` draws a random value from a logistic distribution.

```
Logistic(  
    Location,    ! (optional) numerical expression  
    Scale       ! (optional) numerical expression  
)
```

### Arguments:

*Location*

A scalar numerical expression.

*Scale*

A scalar numerical expression  $> 0$ .

### Return value:

The function `Logistic` returns a random value drawn from a logistic distribution with mean *Location* and scale *Scale*.

### See also:

The Logistic distribution is discussed in full detail in [Appendix A](#) of the Language Reference.

---

## LogNormal

The function `LogNormal` draws a random value from a lognormal distribution.

```
LogNormal(  
  Shape,      ! (input) numerical expression  
  Lowerbound, ! (optional) numerical expression  
  Scale      ! (optional) numerical expression  
)
```

### Arguments:

#### *Shape*

A scalar numerical expression  $> 0$ .

#### *Lowerbound*

A scalar numerical expression.

#### *Scale*

A scalar numerical expression  $> 0$ .

### Return value:

The function `LogNormal` returns a random value drawn from a lognormal distribution with shape *Shape*, lower bound *Lowerbound* and scale *Scale*.

### Remarks:

The prototype of this function has changed with the introduction of AIMMS 3.4. In order to run models that still use the original prototype, the option `Distribution_compatibility` should be set to `Aimms_3_0`. The original function `LogNormal(m, sd)` returns a random value drawn from a lognormal distribution with mean  $m > 0$  and standard deviation  $sd > 0$ . The same result should now be obtained by setting  $Shape = sd/m$ ,  $Lowerbound = 0$  and  $Scale = m$ .

### See also:

The `LogNormal` distribution is discussed in full detail in [Appendix A](#) of the Language Reference.

---

## Normal

The function `Normal` draws a random value from a normal distribution.

```
Normal(  
  Mean,      ! (optional) numerical expression  
  Deviation ! (optional) numerical expression  
)
```

### Arguments:

*Mean*

A scalar numerical expression.

*Deviation*

A scalar numerical expression  $> 0$ .

### Return value:

The function `Normal` returns a random value drawn from a normal distribution with mean *Mean* and standard deviation *Deviation*.

### See also:

The Normal distribution is discussed in full detail in [Appendix A](#) of the Language Reference.

---

## Pareto

The function Pareto draws a random value from a Pareto distribution.

```
Pareto(  
  Shape,          ! (input) numerical expression  
  Location,       ! (optional) numerical expression  
  Scale           ! (optional) numerical expression  
)
```

### Arguments:

#### *Shape*

A scalar numerical expression  $> 0$ .

#### *Location*

A scalar numerical expression.

#### *Scale*

A scalar numerical expression  $> 0$ .

### Return value:

The function Pareto returns a random value drawn from a Pareto distribution with shape *Shape*, location *Location* and scale *Scale*.

### Remarks:

The prototype of this function has changed with the introduction of AIMMS 3.4. In order to run models that still use the original prototype, the option `Distribution_compatibility` should be set to `Aimms_3_0`. The original function `Pareto(s, beta)` returns a random value drawn from a Pareto distribution with shape *beta*, location 0 and scale *s*.

### See also:

The Pareto distribution is discussed in full detail in [Appendix A](#) of the Language Reference.

---

## Triangular

The function `Triangular` draws a random value from a triangular distribution.

```
Triangular(  
    Shape,      ! (input) numerical expression  
    Minimum,    ! (optional) numerical expression  
    Maximum     ! (optional) numerical expression  
)
```

### Arguments:

*Shape*

A scalar numerical expression.

*Minimum*

A scalar numerical expression.

*Maximum*

A scalar numerical expression.

### Return value:

The function `Triangular` returns a random value drawn from a triangular distribution with shape *Shape*, lower bound *Minimum* and upper bound *Maximum*. The argument *Shape* must satisfy the relation  $0 < Shape < 1$ .

### Remarks:

The prototype of this function has changed with the introduction of AIMMS 3.4. In order to run models that still use the original prototype, the option `Distribution_compatibility` should be set to `Aimms_3_0`. The original function `Triangular(a, b, c)` returns a random value drawn from a triangular distribution with a lower bound *a*, likeliest value *b* and upper bound *c*. The arguments must satisfy the relation  $a < b < c$ . The relation between the arguments *Shape* and *b* is given by  $Shape = (b - a) / (c - a)$ .

### See also:

The Triangular distribution is discussed in full detail in Appendix A of the Language Reference.

---

## Uniform

The function `Uniform` draws a random value from a uniform distribution.

```
Uniform(  
    Minimum,      ! (optional) numerical expression  
    Maximum       ! (optional) numerical expression  
)
```

### Arguments:

*Minimum*

A scalar numerical expression.

*Maximum*

A scalar numerical expression.

### Return value:

The function `Uniform` returns a random value drawn from a uniform distribution with lower bound *Minimum* and upper bound *Maximum*.

### Remarks:

The arguments must satisfy the relation  $Minimum < Maximum$ .

### See also:

The `Uniform` distribution is discussed in full detail in [Appendix A](#) of the Language Reference.

---

## Weibull

The function `Weibull` draws a random value from a Weibull distribution.

```
Weibull(  
    Shape,      ! (input) numerical expression  
    Lowerbound, ! (optional) numerical expression  
    Scale       ! (optional) numerical expression  
)
```

### Arguments:

#### *Shape*

A scalar numerical expression  $> 0$ .

#### *Lowerbound*

A scalar numerical expression.

#### *Scale*

A scalar numerical expression  $> 0$ .

### Return value:

The function `Weibull` returns a random value drawn from a Weibull distribution with shape *Shape* lower bound *Lowerbound*, and scale *Scale*.

### Remarks:

The prototype of this function has changed with the introduction of AIMMS 3.4. In order to run models that still use the original prototype, the option `Distribution_compatibility` should be set to `Aimms_3_0`. In the original function `Weibull(Lowerbound, Shape, Scale)`, the arguments were ordered differently.

### See also:

The Weibull distribution is discussed in full detail in [Appendix A](#) of the Language Reference.

---

## DistributionCumulative

The function `DistributionCumulative` computes the cumulative probability value of a given distribution.

```
DistributionCumulative(
    distribution,      ! (input) distribution
    x                 ! (input) numerical expression
)
```

### Arguments:

*distribution*

An expression representing any distribution (such as `Normal(0,1)`).

*x*

A scalar numerical expression.

### Return value:

The function `CumulativeDistribution(distribution,x)`, for  $x \in (-\infty, \infty)$  returns the probability  $P(X \leq x)$  where the stochastic variable  $X$  is distributed according to the given *distribution*.

### Remarks:

For continuous distributions AIMMS can compute the derivatives of the cumulative and inverse cumulative distribution functions. As a consequence, you may use these functions in the constraints of a nonlinear model when the second argument is a variable.

### See also:

The function `DistributionInverseCumulative`. The function `DistributionCumulative` is discussed in full detail in Appendix A of the Language Reference.

---

## DistributionInverseCumulative

The function `DistributionInverseCumulative` computes the inverse cumulative probability value of a given distribution.

```
DistributionInverseCumulative(  
    distribution,      ! (input) distribution  
    alpha             ! (input) numerical expression  
)
```

### Arguments:

*distribution*

An expression representing any distribution (such as `Normal(0,1)`).

*alpha*

A scalar numerical expression within the interval  $[0, 1]$ .

### Return value:

The function `DistributionInverseCumulative(distribution, $\alpha$ )`, for  $\alpha \in [0, 1]$  computes the largest  $x \in (-\infty, \infty)$  such that the probability  $P(X \leq x) \leq \alpha$  where the stochastic variable  $X$  is distributed according to the given *distribution*.

### Remarks:

For continuous distributions AIMMS can compute the derivatives of the cumulative and inverse cumulative distribution functions. As a consequence, you may use these functions in the constraints of a nonlinear model when the second argument is a variable.

### See also:

The function `DistributionCumulative`. The function `DistributionInverseCumulative` is discussed in full detail in [Appendix A](#) of the Language Reference.

---

## DistributionDensity

The function `DistributionDensity` computes the density of a given distribution.

```
DistributionDensity(  
    distribution,      ! (input) distribution  
    x                 ! (input) numerical expression  
)
```

### Arguments:

*distribution*

An expression representing any distribution (such as `Normal(0,1)`).

*x*

A scalar numerical expression.

### Return value:

The function `DistributionDensity(distribution,x)`, for  $x \in (-\infty, \infty)$  returns the expected density around  $x$  of sample points from *distribution*. It is the derivative of `DistributionCumulative(distr,x)`.

### See also:

The functions `DistributionCumulative`, `DistributionInverseDensity`. The function `DistributionDensity` is discussed in full detail in [Appendix A](#) of the Language Reference.

---

## DistributionInverseDensity

The function `DistributionInverseDensity` computes the density of the inverse cumulative function of a given distribution.

```
DistributionInverseDensity(  
    distribution,      ! (input) distribution  
    alpha             ! (input) numerical expression  
)
```

### Arguments:

*distribution*

An expression representing any distribution (such as `Normal(0,1)`).

*alpha*

A scalar numerical expression within the interval  $[0, 1]$ .

### Return value:

The function `DistributionInverseDensity(distribution, $\alpha$ )`, for  $\alpha \in [0, 1]$  returns the inverse density from *distribution*. It is the derivative of `DistributionInverseCumulative(distr,alpha)`.

### See also:

The function `DistributionDensity`. The function `DistributionInverseDensity` is discussed in full detail in Appendix [A](#) of the Language Reference.

---

## DistributionMean

The function `DistributionMean` computes the mean of a given distribution.

```
DistributionMean(  
    distribution      ! (input) distribution  
)
```

### Arguments:

*distribution*

An expression representing any distribution (such as `Normal(0,1)`).

### Return value:

The function `DistributionMean(distribution)` returns the mean of the given *distribution*.

### See also:

You can find more information about the mean of a distribution in Appendix [A](#) of the Language Reference.

---

## DistributionDeviation

The function `DistributionDeviation` computes the expected deviation of the given distribution .

```
DistributionDeviation(  
    distribution      ! (input) distribution  
)
```

### Arguments:

*distribution*

An expression representing any distribution (such as `Normal(0,1)`).

### Return value:

The function `DistributionDeviation(distribution)` returns the expected deviation (distance from the mean) of the *distribution*.

### See also:

You can find more information about the deviation of a distribution in [Appendix A](#) of the Language Reference.

---

## DistributionVariance

The function `DistributionVariance` computes the variance of a given distribution.

```
DistributionVariance(  
    distribution          ! (input) distribution  
)
```

### Arguments:

*distribution*

An expression representing any distribution (such as `Normal(0,1)`).

### Return value:

The function `DistributionVariance(distribution)` returns the variance of the given *distribution*.

### See also:

You can find more information about the variance of a distribution in Appendix [A](#) of the Language Reference.

---

## DistributionSkewness

The function `DistributionSkewness` computes the skewness of a given distribution.

```
DistributionSkewness(  
    distribution          ! (input) distribution  
)
```

### Arguments:

*distribution*

An expression representing any distribution (such as `Normal(0,1)`).

### Return value:

The function `DistributionSkewness(distribution)` returns the skewness of the given *distribution*.

### See also:

You can find more information about the skewness of a distribution in [Appendix A](#) of the Language Reference.

---

## DistributionKurtosis

The function `DistributionKurtosis` computes the kurtosis of a given distribution.

```
DistributionKurtosis(  
    distribution      ! (input) distribution  
)
```

### Arguments:

*distribution*

An expression representing any distribution (such as `Normal(0,1)`).

### Return value:

The function `DistributionKurtosis(distribution)` returns the kurtosis of the given *distribution*.

### See also:

You can find more information about the kurtosis of a distribution in Appendix [A](#) of the Language Reference.

---

## Combination

The function `Combination` computes the number of combinations of length  $m$  in  $n$  items.

```
Combination(  
    n,          ! (input) integer expression  
    m          ! (input) integer expression  
)
```

### Arguments:

$n$   
An integer numerical expression  $\geq 0$ .

$m$   
An integer numerical expression in the range  $0, \dots, n$ .

### Return value:

The function `Combination` returns  $\binom{n}{m}$ , the number of combinations of length  $m$  in a given number of items  $n$ .

### See also:

Combinatoric functions are discussed in full detail in Section [6.1.7](#).

---

## Factorial

The function `Factorial` returns the factorial of an integer number.

```
Factorial(  
    n          ! (input) integer expression  
)
```

### Arguments:

*n*  
An integer numerical expression  $\geq 0$ .

### Return value:

The function `Factorial` returns the factorial value  $n!$ .

### See also:

Combinatoric functions are discussed in full detail in Section [6.1.7](#).

---

## Permutation

The function `Permutation` computes the number of permutations of length  $m$  in  $n$  items.

```
Permutation(  
    n,          ! (input) integer expression  
    m          ! (input) integer expression  
)
```

### Arguments:

$n$   
An integer numerical expression  $\geq 0$ .

$m$   
An integer numerical expression in the range  $0, \dots, n$ .

### Return value:

The function `Permutation` returns  $m! \cdot \binom{n}{m}$ , the number of permutations of length  $m$  in a given number of items  $n$ .

### See also:

Combinatoric functions are discussed in full detail in Section [6.1.7](#).