# distributions

Library routines for generating random variates from all of the usual discrete and continuous distributions, e.g., Bern(p), Geom(p), Exp(λ) …

This series of random number generators allows you to create random numbers that fit discrete and continuous distributions given a Unif(0,1) random number.

* ☒ [Standard Normal](#standard-normal)
* ☒ [Erlang](#erlang)
* ☒ [Weibull](#weibull)
* ☒ Geometric
* ☒ [Bernoulli](#bernoulli)
* ☒ [Exponential](#exponential)
* ☒ [Gamma](#gamma)
* ☒ [Negative Binomial](#negative-binomial)
* ☒ [Poisson](#poisson)
* ☒ [Triangular](#triangular)

Go tests have been written against the Mean and Variance of the distributions and the documentation below will show the histograms from samples of random numbers returned by the random number generation functions.

Contributers:

Ni Li Joshua McDonald

## Using this library

To use this library just go get this repository and use the following functions to start generating random numbers.

All functions return a single random variate that fits the distribution selected. The distributions have the following functions:

ExpectedValue([]interface{}) interface{}  
 Variance([]interface{}) float64  
 RandVar(interface{}) ([]interface{}, error)  
}

For most distributions the RandVar() function returns a single number, however, since the standard normal distribution is using the Box-Mueller method it will return a pair of numbers.

ArrayMean([]float64) has been added to allow you to get the mean of an array of numbers easily. You will notice that as you make the array larger it will come closer to the ExpectedValue of the distribution.

## Bernoulli

Bernoulli’s RandVar() takes a probability of p and given a Unif(0,1) number returns a 0 if the random number is less than or equal to p, otherwise returns 1.

The function takes a float64, probability as a parameter. The function will return an error if the probability value is *less than* 0 OR *greater than* 1.

#### Example

var p = .25  
 d := BernoulliDistribution{  
 DistributionType: "Bernoulli",  
 }  
 n, \_ := d.RandVar(p)

#### Mean

E(X) = p

#### Variance

Var[X] = pq = p(1-p) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ## Exponential

The exponential distribution is commonly used to model time between events or time between failures.

The RandVar() function generates random numbers from a Unif(0,1) random number so that the returned value fits an exponential distribution given a scale parameter which is a float64.

The function will return an error if the scale parameter value is *less than* 0 OR *equal to* 0.

#### Example

a := 1.0 // scale parameter  
 d := ExponentialDistribution{  
 DistributionType: "Exponential",  
 }  
 n, \_ := d.RandVar(a)

#### Mean

E[X] = 1/scale parameter

#### Variance

Var(X) = 1/scale parameter^2 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ## Weibull The RandVar function of the Weibull distribution take a scale parameter *a* and a shape parameter *b* and returns a random valriable that fits the Weibull distribution.

The parameter *a* must be > 0 The parameter *b* must be > 0

Weibull is commonly used to model failure rates of electronics where the failure rate :

\_b\_ < 1 ▶️ increases over time \_b\_ > 1 ▶️ decreases over time \_b\_ = 1 ▶️ constant over time

#### Example

a := 1.5  
 b := float64(1)  
  
 d := WeibullDistribution{  
 DistributionType: "Weibull",  
 }  
 n, \_ := d.RandVar(a, b)

#### Expected Value

E(X) = a / b Γ (1/b)

#### Variance

Var(X) = a^2 / b^2 [2 \* b \* Γ(2/b) - {Γ(1/b)}^2}] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ## Standard Normal

The normal distribution’s RandVar() function takes a mean and standard deviation and returns two random variables z1, z2 using the Box-Muller method.

#### Example

mean := .5  
 sd := 1  
 d := NormalDistribution{  
 DistributionType: "Normal",  
 }  
 n, \_ := d.RandVar(m, sd)

#### Expected Value

E[X] = μ

#### Variance

Var(X) = σ^2 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Triangular

A triangular distribution has a lower limit (min), an upper limit (max) and mode and is used to describe a population when there is a limited amount of sample data.

The *min* parameter must be lower than the *max* parameter

min := float64(0)  
 mode := .5  
 max := float64(1)  
  
 d := TriangularDistribution{  
 DistributionType: "Triangular",  
 }  
 n, \_ := d.RandVar(min, mode, max)

#### Expected Value

E(X) = (min + mode + max) / 3 #### Variance Var(X) = (min^2 + max^2 + mode^2 - (min \* max) - (min \* mode) - (max \* mode)) / 18 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Negative Binomial

The negative binomial distribution is used to model the number of failures *x* before the *nth* success. The RandVar() function takes in *p* which is the probability of success and *n* which is the number of successes. This function calculates the sum of *n* geometric variates G(p).

The parameter *p* must be 0 < p < 1 The parameter *n* must be a positive integer

p := .25  
 n := 4  
 d := NegativeBinomialDistribution{  
 DistributionType: "NegativeBinomialDistribution",  
 }  
 x, \_ := d.RandVar(p, n)

#### Expected Value

E(X) = n(1-p)/p #### Variance Var(X) = n(1-p)/p^2 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Poisson

The Poisson distribution is used to model the number of arrivals over a given interval.

Since the function is using the direct method the value of λ has been limited to 20.

The parameter *λ* must be > 0 and < 21

lambda := float64(2)  
 d := PoissonDistribution{  
 DistributionType: "Poisson",  
 }  
 n, \_ := d.RandVar(lambda)

#### Expected Value

E(X) = λ #### Variance Var(X) = λ

## Erlang

Where the events that occur can be modeld by the poisson distribution, the waiting times between k occurrences of the event are Erlang distributed.

lambda := float64(2)  
 k := 1  
  
 d := ErlangDistribution{  
 DistributionType: "Erlang",  
 }  
 n, \_ := d.RandVar(k, lambda)

#### Expected Value

E(X) = k / λ #### Variance Var(X) = k / λ^2

## Gamma

The gamma distribution RandVar() function produces random variables given the shape paramter *k* and scale parameter *s*.

Lots of help on this from

* [www.hongliangjie.com](https://www.hongliangjie.com/2012/12/19/how-to-generate-gamma-random-variables/)
* [Gamma Distribution](https://en.wikipedia.org/wiki/Gamma_distribution)
* [Logarithmic Transformation-Based Gamma Random Number Generators](https://www.stat.purdue.edu/~xbw/research/jss102013.gamma.pdf)

This generator relies on the matlab example which has been translated to Go and uses the standard normal random variate generator function from this same package.

k := 5  
 s := 1  
  
 d := GammaDistribution{  
 DistributionType: "Gamma",  
 }  
 n, \_ := d.RandVar(k, s)

#### Expected Value

E(X) = k / s #### Variance Var(X) = k / s^2