randvar A class to manage a random variable.

* [Help chapter 1](index.html)
* [Help chapter 1](section_b95303b40d80bf443b9812f77074bc91.html)
* [Overview](nisp_overview.html)
* [randvar](nisp_randvar.html)
* [setrandvar](nisp_setrandvar.html)
* [polychaos](nisp_polychaos.html)
* [Tutorials](section_92e5b1d134f462b45362dd31534e5eb9.html)
* [Configuration](section_7d3c5583ddd6f69cccc9568e109f6b70.html)
* [Sensitivity Analysis](section_dcab75b909a8f13b16ecd51f42f2874a.html)
* [Test Functions](section_116d444de7bacb73d7d34d22d4e07942.html)

 [setrandvar >>](nisp_setrandvar.html) [<< Overview](nisp_overview.html)

# randvar

A class to manage a random variable.

### Static Functions

tokenmatrix=randvar\_tokens()
nb=randvar\_size()

### Creation/Destruction

rv=randvar\_new(name,a,b)
rv=randvar\_new(name,a)
rv=randvar\_new(name)
randvar\_destroy(rv)

### Methods

value=randvar\_getvalue(rv)
value=randvar\_getvalue(rv,rv2,value2)
randvar\_getlog(rv)

### Description

This class manages a random variable.

The randvar\_new function allows to create a new random variable. The randvar\_destroy function allows to destroy an existing random variable. The type of random variable is chosen at the creation of the random variable to the randvar\_new function. Several random variable types are available, including Normal and Uniform variables (see below for a complete list). Once a random variable is created, the randvar\_getvalue function allows to generate random numbers from the associated distribution function.

The randvar\_new function returns a token which is a unique identifier for the new random variable. Indeed, when a new random variable is created, a counter is updated which corresponds to the returned token. This way, each token is unique and can correspond only to one single random variable.

The functions randvar\_tokens and randvar\_size allow to manage the random variables which have already been created. The function randvar\_size returns the number of random variables, while the randvar\_tokens function returns the list of current random variables.

### Static Functions

tokenmatrix=randvar\_tokens()

Returns a 1-by-n matrix of floating point integers representing the current randvar tokens, where n is the number of tokens.

n=randvar\_size()

Returns a 1-by-1 matrix containing the current number of randvar tokens currently in use, where n is the number of tokens.

### Creation/Destruction

rv=randvar\_new(name)

Returns a new random variable rv which corresponds to the distribution function given by name, with default parameters.

The following is the list of possible values for the name input argument. See below for the details of a and b.

* "Normale" with a=0 and b=1
* "Uniforme" with a=0 and b=1
* "Exponentielle" with a=1
* "LogNormale" with a=0 and b=1
* "LogUniforme" with a=0 and b=1

rv=randvar\_new(name,a)

Returns a new random variable rv with first parameter a.

The following is the list of possible values for the name input argument.

* "Exponentielle" with scale parameter a (i.e. the mean of the random variable is 1/a).

rv=randvar\_new(name,a,b)

Returns a new random variable rv with first parameter a and second parameter b.

The following is the list of possible values for the name input argument.

* "Normale" where X is a normal random variable with mean a and standard deviation b.
* "Uniforme" where X i uniform in [a,b].
* "LogNormale" where log(X) is a normal random variable with mean a and standard deviation b.
* "LogUniforme" where log(X) is uniform in [a,b].

randvar\_destroy(rv)

Destroys the current random variable.

### Methods

value = randvar getvalue(rv)

Returns a random value from the distribution function of the current random variable.

value = randvar getvalue(rv,rv2,value2)

Returns a random value from the distribution function of the random variable rv by transformation of value2 from the distribution function of random variable rv2.

randvar getlog(rv)

Prints a log for the current random variable.

### Distributions

In this section, we present the mean M and variance V of the random variables available in the randvar class. We also present the suppport of the random variables, i.e. the interval which contains the outcomes.

"Normale"

The support is (-%inf,+%inf).

 M=a
 V=b^2

"Uniforme"

The support is [a,b].

 M=(a+b)/2
 V=(b-a)^2/12

"Exponentielle"

The support is [0,+%inf).

 M=1/a
 V=1/a^2

"LogNormale"

The support is [0,+%inf). The mean and standard deviation of log(X) are a and b.

 M=exp(a+b^2/2)
 V=(exp(b^2)-1)\*exp(2\*a+b^2)

If X is a LogNormale random variable with mean M and variance V, then:

 a=log(E)-0.5\*log(1+V/E^2)
 b=sqrt(log(1+V/E^2))

"LogUniforme"

The support is [exp(a),exp(b)].

 M=(exp(b)-exp(a))/(b-a)
 V=0.5\*(exp(b)^2-exp(a)^2)/(b-a)-M

If X is a LogUniforme random variable with minimum A and maximum B, then:

 a=log(A)
 b=log(B)

### Examples

In the following example, we call the randvar\_new function in order to create a Normal random variable with mean 1.0 and standard deviation 0.5. Then, we perform a loop so that we get 1000 values from this random variable. In order to check that these values are associated with a Normal distribution function, we compute the mean and variance and check that this corresponds to the expected results. Finally, the random variable is destroyed with the randvar\_destroy function.

nisp\_initseed(0);
mu = 1.0;
sigma = 0.5;
rv=randvar\_new("Normale",mu,sigma);

nbshots = 1000;
values = zeros(nbshots);
for i=1:nbshots
values(i)=randvar\_getvalue(rv);
end

mymean = mean (values);
mysigma = st\_deviation(values);
myvariance = variance (values);

mprintf("Mean=%f (exact=%f)\n", mymean, mu);
mprintf("Std. dev.=%f (exact=%f)\n", mysigma, sigma);
mprintf("Var.=%f (exact=%f)\n", myvariance,sigma^2);

randvar\_destroy(rv);

scf();
histplot(50,values)
xtitle("Histogram of X","X","P(x)")

The previous script produces the following output.



### Example of randvar\_getlog

In the following session, we use the randvar\_getlog function and print a Normal variable in the console.

mu = 1.0;
sigma = 0.5;
rv=randvar\_new("Normale",mu,sigma);
randvar\_getlog(rv);
randvar\_destroy(rv);

The previous script produces the following output.

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
 Nisp(RandomVariable::GetLog) for RandomVariable
 Normale : 1 : 0.5
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### Example of Log-Normal variable.

In the following example, we create a Log-Normal variable. We compute the parameters from the parameters of the underlying Normal variable.

nisp\_initseed(0);
a = 1.0;
b = 0.5;
rv=randvar\_new("LogNormale",a,b);
nbshots = 1000;
values = zeros(nbshots);
for i=1:nbshots
values(i)=randvar\_getvalue(rv);
end
// Check the mean of log(X)
mymean = mean (log(values))
expected = a
// Check the variance of log(X)
myvariance = variance (log(values))
expected = b^2
randvar\_destroy(rv);
// Draw the histogram
scf();
histplot(50,values);
xtitle("LogNormal","X","P(x)");
// Compare with PDF
x = linspace(0,10,100);
p = distfun\_lognpdf(x,a,b);
plot(x,p,"b-");
legend(["Empirical" "PDF"]);

The previous script produces the following output.



### Example of Log-Uniform variable.

In the following example, we create a Log-Uniform variable. We compute the parameters from the parameters of the underlying Uniform variable.

nisp\_initseed(0);
a = 4;
b = 6;
rv=randvar\_new("LogUniforme",a,b);
nbshots = 1000;
values = zeros(nbshots);
for i=1:nbshots
values(i)=randvar\_getvalue(rv);
end
// Check the mean of log(X)
mymean = mean (log(values))
expected = (a+b)/2
// Check the variance of log(X)
myvariance = variance (log(values))
expected = (b-a)^2/12
randvar\_destroy(rv);
// Draw the histogram
scf();
histplot(20,values);
xtitle("LogUniform","X","P(x)");
x=linspace(exp(a),exp(b),100);
p=distfun\_logupdf(x,a,b);
plot(x,p,"b-");
legend(["Empirical" "PDF"]);

The previous script produces the following output.



### Example of Exponential variable.

nisp\_initseed(0);
// Define a Exponential variable.
scale = 12.0;
rv=randvar\_new("Exponentielle",scale);
// Perform Monte-Carlo simulations
nbshots = 1000;
values = zeros(nbshots);
for i=1:nbshots
values(i)=randvar\_getvalue(rv);
end
// Check the mean
mymean = mean (values)
expected = 1/scale
// Check the variance
myvariance = variance (values)
expected = 1/scale^2
randvar\_destroy(rv);
// Draw the histogram
scf();
histplot(20,values);
xtitle("Histogram of X","X","P(x)");
// Compare with PDF
x = linspace(0,0.5,100);
p = distfun\_exppdf(x,1/scale);
plot(x,p,"b-");
legend(["Empirical" "PDF"]);

The previous script produces the following output.



### Authors

Copyright (C) 2012-2013 - Michael Baudin

Copyright (C) 2008-2011 - CEA - Jean-Marc Martinez

Copyright (C) 2008-2011 - INRIA - Michael Baudin

 [setrandvar >>](nisp_setrandvar.html) [<< Overview](nisp_overview.html)

scilab-5.5.2

[Scilab Enterprises](http://www.scilab.org/)
Copyright (c) 2011-2014 (Scilab Enterprises)
Copyright (c) 1989-2012 (INRIA)
Copyright (c) 1989-2007 (ENPC)
with [contributors](https://www.scilab.org/development/contribute/contributors)

Last updated:
Tue Jan 01 16:22:26 CET 2019