

# Package: convdistr (via r-universe)

September 2, 2024

**Type** Package

**Title** Convolute Probabilistic Distributions

**Version** 1.6.2.9000

**URL** <https://github.com/johnaponte/convdistr>,  
<https://johnaponte.github.io/convdistr/>

**Description** Convolute probabilistic distributions using the random generator function of each distribution. A new random number generator function is created that perform the mathematical operation on the individual random samples from the random generator function of each distribution. See the documentation for examples.

**License** GPL (>= 3)

**Encoding** UTF-8

**Suggests** spelling, testthat, knitr, rmarkdown

**Imports** stats, graphics, pryr, extraDistr, dplyr, tidyr, ggplot2, RColorBrewer, SHELF, MASS, shiny

**RoxygenNote** 7.3.1

**VignetteBuilder** knitr

**Language** en-US

**Repository** <https://johnaponte.r-universe.dev>

**RemoteUrl** <https://github.com/johnaponte/convdistr>

**RemoteRef** HEAD

**RemoteSha** c1862226840d9c7e1869cd738e2b1943c0664977

## Contents

add_total	2
BETA	3
BETABINOMIAL	4
BINOMIAL	6

cinqnum . . . . .	6
CONVOLUTION . . . . .	7
CONVOLUTION_assoc . . . . .	9
CONVOLUTION_comb . . . . .	10
DIRAC . . . . .	12
DIRICHLET . . . . .	12
DISCRETE . . . . .	13
DISTRIBUTION . . . . .	14
DISTRIBUTION_factory . . . . .	15
EXPONENTIAL . . . . .	16
fitbeta . . . . .	17
fitdirichlet . . . . .	18
ggDISTRIBUTION . . . . .	19
LOGNORMAL . . . . .	19
metadata . . . . .	20
NA_DISTRIBUTION . . . . .	21
new_MIXTURE . . . . .	21
new_MULTINORMAL . . . . .	22
NORMAL . . . . .	23
omit_NA . . . . .	24
plot.DISTRIBUTION . . . . .	24
POISSON . . . . .	25
restrict_environment . . . . .	26
rfunc . . . . .	27
rfunc.default . . . . .	27
rfunc.DISTRIBUTION . . . . .	28
same_dimensions . . . . .	28
set_seed . . . . .	29
summary.DISTRIBUTION . . . . .	29
TRIANGULAR . . . . .	30
TRUNCATED . . . . .	31
UNIFORM . . . . .	32
<b>Index</b>	<b>33</b>

---

add_total	<i>Adds a total dimension</i>
-----------	-------------------------------

---

### Description

This function returns a [DISTRIBUTION](#) with a new dimension created by row sum of the dimensions of the distribution.

### Usage

```
add_total(p_distribution, p_totalname = "TOTAL")
```

**Arguments**

`p_distribution` an object of class [DISTRIBUTION](#)  
`p_totalname` the name of the new dimension

**Details**

Only works with multidimensional distributions.

**Value**

a [DISTRIBUTION](#)

**Author(s)**

John J. Aponte

**Examples**

```
d1 <- new_DIRICHLET(c(0.2,0.5,0.3))
d2 <- add_total(d1)
```

---

BETA

*Factory for a BETA distribution object*

---

**Description**

Returns an BETA distribution object that produce random numbers from a beta distribution using the [rbeta](#) function

**Usage**

```
new_BETA(p_shape1, p_shape2, p_dimnames = "rvar")
new_BETA_lci(p_mean, p_lci, p_uci, p_dimnames = "rvar")
new_BETA_lci2(p_mean, p_lci, p_uci, p_dimnames = "rvar")
```

**Arguments**

`p_shape1` non-negative parameters of the Beta distribution  
`p_shape2` non-negative parameters of the Beta distribution  
`p_dimnames` A character that represents the name of the dimension  
`p_mean` A numeric that represents the expected value of the proportion  
`p_lci` A numeric for the lower 95% confidence interval  
`p_uci` A numeric for the upper 95% confidence interval

**Value**

An object of class DISTRIBUTION, BETA

**Functions**

- `new_BETA_lci()`: Constructor based on confidence intervals. Preserve expected value.
- `new_BETA_lci2()`: Constructor based on ML confidence intervals

**Note**

When using confidence intervals, the shape parameters are obtained using the following formula:

$$varp = (p_{uci} - p_{lci})/4^2$$

$$shape1 = p_{mean} * (p_{mean} * (1 - p_{mean})/varp - 1)$$

$$shape2 = (1 - p_{mean}) * (p_{mean} * (1 - p_{mean})/varp - 1)$$

`new_BETA_lci2` estimate parameters using maximum likelihood `myDistr <- new_BETA_lci2(0.30,0.25,0.35)`  
`myDistr$rfunc(10)`

**Author(s)**

John J. Aponte

**Examples**

```
myDistr <- new_BETA(1,1)
myDistr$rfunc(10)
myDistr <- new_BETA_lci(0.30,0.25,0.35)
myDistr$rfunc(10)
```

---

BETABINOMIAL

*Factory for a BETABINOMIAL distribution object*

---

**Description**

Returns an BETABINOMIAL distribution object that produce random numbers from a betabinomial distribution using the `rbbinom` function

**Usage**

```
new_BETABINOMIAL(p_size, p_shape1, p_shape2, p_dimnames = "rvar")
```

```
new_BETABINOMIAL_od(p_size, p_mu, p_od, p_dimnames = "rvar")
```

```
new_BETABINOMIAL_icc(p_size, p_mu, p_icc, p_dimnames = "rvar")
```

**Arguments**

<code>p_size</code>	a non-negative parameter for the number of trials
<code>p_shape1</code>	non-negative parameters of the Betabinomial distribution
<code>p_shape2</code>	non-negative parameters of the Betabinomial distribution
<code>p_dimnames</code>	A character that represents the name of the dimension
<code>p_mu</code>	mean proportion for the binomial part of the distribution
<code>p_od</code>	over dispersion parameter
<code>p_icc</code>	intra-class correlation parameter

**Value**

An object of class DISTRIBUTION, BETADISTRIBUTION

**Functions**

- `new_BETABINOMIAL_od()`: parametrization based on dispersion
- `new_BETABINOMIAL_icc()`: parametrization based on intra-class correlation

**Note**

There are several parametrization for the betabinomial distribution. The one based on `shape1` and `shape2` are parameters alpha and beta of the beta part of the distribution, but it can be parametrized as `mu`, and `od` where `mu` is the expected mean proportion and `od` is a measure of the overdispersion.

$$p_mu = p_shape1 / (p_shape1 + p_shape2)$$

$$p_od = p_shape1 + p_shape2$$

$$p_shape1 = p_mu * p_od$$

$$p_shape2 < -(1 - p_mu) * p_od$$

Another parametrization is based on `mu` and the `icc` where `mu` is the mean proportion and `icc` is the intra-class correlation.

$$p_mu = p_shape1 / (p_shape1 + p_shape2)$$

$$p_icc = 1 / (p_shape1 + p_shape2 + 1)$$

$$p_shape1 = p_mu * (1 - p_icc) / p_icc$$

$$p_shape2 = (1 - p_mu) * (1 - p_icc) / p_icc$$

**Author(s)**

John J. Aponte

**Examples**

```
myDistr <- new_BETABINOMIAL(10,1,1)
myDistr$rfunc(10)
```

BINOMIAL

*Factory for a BINOMIAL distribution object*

---

**Description**

Returns a BINOMIAL distribution object that produce random numbers from a binomial distribution using the `rbinom` function

**Usage**

```
new_BINOMIAL(p_size, p_prob, p_dimnames = "rvar")
```

**Arguments**

<code>p_size</code>	integer that represent the number of trials
<code>p_prob</code>	probability of success
<code>p_dimnames</code>	A character that represents the name of the dimension

**Value**

An object of class `DISTRIBUTION`, `BINOMIAL`

**Author(s)**

John J. Aponte

**Examples**

```
myDistr <- new_BINOMIAL(1000,0.3)
myDistr$rfunc(10)
```

---

cinqnum*cinqnum*

---

**Description**

Produce 5 numbers of the distribution (`mean_`, `sd_`, `lci_`, `uci_`, `median_`).

**Usage**

```
cinqnum(x, ...)  
  
## S3 method for class 'DISTRIBUTION'  
cinqnum(x, n, ...)  
  
## S3 method for class '^NA^'  
cinqnum(x, n, ...)  
  
## S3 method for class 'DIRAC'  
cinqnum(x, n, ...)
```

**Arguments**

x	an object of class <a href="#">DISTRIBUTION</a>
...	further parameters
n	number of draws

**Details**

Uses the stored seed to have the same sequence always and produce the same numbers This is an internal function for the summary function

**Value**

a vector with the mean, sd, lci, uci and median values

**Methods (by class)**

- `cinqnum(DISTRIBUTION)`: Generic method for a [DISTRIBUTION](#)
- `cinqnum(^NA^)`: Generic method for optimized for a NA distribution
- `cinqnum(DIRAC)`: Generic method optimized for a [DIRAC](#) distribution

**Author(s)**

John J. Aponte

---

CONVOLUTION

*Make the convolution of two or more [DISTRIBUTION](#) objects*

---

**Description**

The convolution of the simple algebraic operations is made by the operation of individual draws of the distributions. The [DISTRIBUTION](#) objects must have the same dimensions.

**Usage**

```

new_CONVOLUTION(listdistr, op, omit_NA = FALSE)

new_SUM(..., omit_NA = FALSE)

## S3 method for class 'DISTRIBUTION'
e1 + e2

new_SUBTRACTION(..., omit_NA = FALSE)

## S3 method for class 'DISTRIBUTION'
e1 - e2

new_MULTIPLICATION(..., omit_NA = FALSE)

## S3 method for class 'DISTRIBUTION'
e1 * e2

new_DIVISION(..., omit_NA = FALSE)

## S3 method for class 'DISTRIBUTION'
e1 / e2

```

**Arguments**

<code>listdistr</code>	a list of <a href="#">DISTRIBUTION</a> objects
<code>op</code>	a function to convolute '+', '-', '*', '\'
<code>omit_NA</code>	if TRUE, NA distributions will be omitted
<code>...</code>	<a href="#">DISTRIBUTION</a> objects or a list of distribution objects
<code>e1</code>	object of class <a href="#">DISTRIBUTION</a>
<code>e2</code>	object of class <a href="#">DISTRIBUTION</a>

**Details**

If any of the distributions is of class NA ([NA\\_DISTRIBUTION](#)) the result will be a new distribution of class NA unless the `omit_NA` option is set to TRUE

**Value**

and object of class CONVOLUTION, [DISTRIBUTION](#)

**Functions**

- `new_SUM()`: Sum of distributions
- `new_SUBTRACTION()`: Subtraction for distributions
- `new_MULTIPLICATION()`: Multiplication for distributions
- `new_DIVISION()`: DIVISION for distributions



**Author(s)**

John J. Aponte

**Examples**

```

x1 <- new_NORMAL(0,1)
x2 <- new_UNIFORM(1,2)
new_CONVOLUTION(list(x1,x2), `+`)
new_SUM(x1,x2)
x1 + x2
new_SUBTRACTION(x1,x2)
x1 - x2
new_MULTIPLICATION(list(x1,x2))
x1 * x2
new_DIVISION(list(x1,x2))
x1 / x2

```

---

CONVOLUTION\_assoc

*Convolution with association of dimensions*


---

**Description**

In case of different dimensions of the distribution this function perform the operation on the common distributions and add without modifications the other dimensions of the distribution.

**Usage**

```

new_CONVOLUTION_assoc(dist1, dist2, op)

new_SUM_assoc(dist1, dist2)

new_SUBTRACTION_assoc(dist1, dist2)

new_MULTIPLICATION_assoc(dist1, dist2)

new_DIVISION_assoc(dist1, dist2)

```

**Arguments**

dist1	an object of class <a href="#">DISTRIBUTION</a>
dist2	and object of class <a href="#">DISTRIBUTION</a>
op	one of '+', '-', '*', '/'

**Details**

If distribution A have dimensions a and b and distribution B have dimensions b and c, the A + B would produce a distribution with dimensions a, c, b+b,

**Value**

an object of class `DISTRIBUTION`

**Functions**

- `new_SUM_assoc()`: Sum of distributions
- `new_SUBTRACTION_assoc()`: Subtraction of distributions
- `new_MULTIPLICATION_assoc()`: Multiplication of distributions
- `new_DIVISION_assoc()`: Division of distributions

**Author(s)**

John J. Aponte

**Examples**

```
x1 <- new_MULTINORMAL(c(0,1), matrix(c(1,0.5,0.5,1),ncol=2), p_dimnames = c("A","B"))
x2 <- new_MULTINORMAL(c(10,1), matrix(c(1,0.4,0.4,1),ncol=2), p_dimnames = c("B","C"))
new_CONVOLUTION_assoc(x1,x2, `+`)
new_SUM_assoc(x1,x2)
new_SUBTRACTION_assoc(x1,x2)
new_MULTIPLICATION_assoc(x1,x2)
new_DIVISION_assoc(x1,x2)
```

---

CONVOLUTION\_comb

*Convolution with combination of dimensions*

---

**Description**

In case of different dimensions of the distribution this function perform the operation on the combination of the distributions of both distribution.

**Usage**

```
new_CONVOLUTION_comb(dist1, dist2, op, p_dimnames)
```

```
new_SUM_comb(dist1, dist2)
```

```
new_SUBTRACTION_comb(dist1, dist2)
```

```
new_MULTIPLICATION_comb(dist1, dist2)
```

```
new_DIVISION_comb(dist1, dist2)
```

**Arguments**

dist1	an object of class <code>DISTRIBUTION</code>
dist2	and object of class <code>DISTRIBUTION</code>
op	one of '+', '-', '*', '/'
p_dimnames	a character vector with the name of the dimensions. If missing the combination of the individual dimensions will be used

**Details**

If distribution A have dimensions a and b and distribution B have dimensions b and c, the A + B would produce a distribution with dimensions a\_b,a\_c,b\_b, b\_c

**Value**

an object of class `DISTRIBUTION`

**Functions**

- `new_SUM_comb()`: Sum of distributions
- `new_SUBTRACTION_comb()`: Subtraction of distributions
- `new_MULTIPLICATION_comb()`: Multiplication of distributions
- `new_DIVISION_comb()`: Division of distributions

**Note**

In case of the same dimensions, only the first combination is taken

**Author(s)**

John J. Aponte

**Examples**

```
x1 <- new_MULTINORMAL(c(0,1), matrix(c(1,0.5,0.5,1),ncol=2), p_dimnames = c("A","B"))
x2 <- new_MULTINORMAL(c(10,1), matrix(c(1,0.4,0.4,1),ncol=2), p_dimnames = c("B","C"))
new_CONVOLUTION_comb(x1,x2, `+`)
new_SUM_comb(x1,x2)
new_SUBTRACTION_comb(x1,x2)
new_MULTIPLICATION_comb(x1,x2)
new_DIVISION_comb(x1,x2)
```

---

DIRAC *Factory for a DIRAC distribution object*

---

**Description**

Returns an DIRAC distribution object that always return the same number, or the same matrix of numbers in case multiple dimensions are setup

**Usage**

```
new_DIRAC(p_scalar, p_dimnames = "rvar")
```

**Arguments**

`p_scalar` A numeric that set the value for the distribution  
`p_dimnames` A character that represents the name of the dimension

**Value**

An object of class DISTRIBUTION, DIRAC

**Author(s)**

John J. Aponte

**Examples**

```
myDistr <- new_DIRAC(1)  
myDistr$rfunc(10)
```

---

DIRICHLET *Factory for a DIRICHLET distribution object*

---

**Description**

Returns an DIRICHLET distribution object that draw random numbers generated by the function `rdirichlet`

**Usage**

```
new_DIRICHLET(p_alpha, p_dimnames)
```

**Arguments**

`p_alpha` k-value vector for concentration parameter. Must be positive  
`p_dimnames` A vector of characters for the names of the k-dimensions

**Details**

A name can be provided for the dimensions. Otherwise `rvar1`, `rvar2`, ..., `rvark` will be assigned

**Value**

An object of class `DISTRIBUTION`, `p_distribution$distribution`, `TRUNCATED`

**Author(s)**

John J. Aponte

**Examples**

```
myDistr <- new_DIRICHLET(c(0.3,0.2,0.5), c("a","b","c"))
myDistr$rfunc(10)
```

---

DISCRETE

*Factory for a DISCRETE distribution object*

---

**Description**

Returns an `DISCRETE` distribution object that sample from the vector `p_supp` of options with probability the vector of probabilities `p_prob`.

**Usage**

```
new_DISCRETE(p_supp, p_prob, p_dimnames = "rvar")
```

**Arguments**

<code>p_supp</code>	A numeric vector of options
<code>p_prob</code>	A numeric vector of probabilities.
<code>p_dimnames</code>	A character that represents the name of the dimension

**Value**

An object of class `DISTRIBUTION`, `DISCRETE`

**Note**

If the second argument is missing, all options will be sample with equal probability. If provided, the second argument would add to 1 and must be the same length that the first argument

**Author(s)**

John J. Aponte

**Examples**

```
myDistr <- new_DISCRETE(p_supp=c(1,2,3,4), p_prob=c(0.40,0.30,0.20,0.10))
myDistr$rfunc(10)
```

---

DISTRIBUTION

*DISTRIBUTION class*


---

**Description**

DISTRIBUTION is a kind of abstract class (or interface) that the specific constructors should implement.

**Details**

It contains 4 fields

**distribution** A character with the name of the distribution implemented

**seed** A numerical that is used for `details` to produce reproducible details of the distribution

**oval** Observed value. Is the value expected. It is used as a number for the mathematical operations of the distributions as if they were a simple scalar

**rfunc** A function that generate random numbers from the distribution. Its only parameter `n` is the number of draws of the distribution. It returns a matrix with as many rows as `n`, and as many columns as the dimensions of the distributions

The DISTRIBUTION objects could support multidimensional distributions for example [DIRICHLET](#). The names of the dimensions should coincides with the names of the `oval` vector. If only one dimension, the default name is `rvar`.

It is expected that the `rfunc` is included in the creation of new distributions by convolution so the environment should be carefully controlled to avoid reference leaking that is possible within the R language. For that reason, `rfunc` should be created within a [restrict\\_environment](#) function

Once the object is instanced, the fields are immutable and should not be changed. If the seed needs to be modified, a new object can be created using the [set\\_seed](#) function

Objects are defined for the following distributions

- [UNIFORM](#)
- [NORMAL](#)
- [BETA](#)
- [TRIANGULAR](#)
- [POISSON](#)
- [EXPONENTIAL](#)
- [DISCRETE](#)
- [DIRAC](#)
- [DIRICHLET](#)
- [TRUNCATED](#)
- [NA\\_DISTRIBUTION](#)

**Value**

a DISTRIBUTION object

**Author(s)**

John J. Aponte

---

DISTRIBUTION\_factory *A factory of DISTRIBUTION classes*

---

**Description**

Generate a function that creates DISTRIBUTION objects

**Usage**

```
DISTRIBUTION_factory(distname, rfunction, ovalfunc)
```

**Arguments**

distname	name of the distribution. By convention they are upper case
rfunction	a function to generate random numbers from the distribution
ovalfunc	a function that calculate the oval value, should used only the same arguments that the rfunction

**Value**

A function that is able to create DISTRIBUTION objects.

**Note**

The function return a new function, that have as arguments the formals of the rfunction plus a new argument dimnames for the dimension names. If The distribution is unidimensional, the default value dimnames = "rvar" will works well, but if not, the dimnames argument should be specified when the generated function is used as in the example for the new\_MyDIRICHLET

**Author(s)**

John J. Aponte

**Examples**

```

new_MYDISTR <- DISTRIBUTION_factory("MYDISTR", rnorm, function(){mean})
d1 <- new_MYDISTR(0,1)
summary(d1)
require(extraDistr)
new_MyDIRICHLET <- DISTRIBUTION_factory('rdirichlet',
                                       rdirichlet,
                                       function() {
                                         salpha = sum(alpha)
                                         alpha / salpha
                                       })
d2 <- new_MyDIRICHLET(c(10, 20, 70), dimnames = c("A", "B", "C"))
summary(d2)

```

---

EXPONENTIAL

*Factory for a EXPONENTIAL distribution using confidence intervals*


---

**Description**

Returns an EXPONENTIAL distribution object that produce random numbers from an exponential distribution using the [rexp](#) function

**Usage**

```
new_EXPONENTIAL(p_rate, p_dimnames = "rvar")
```

**Arguments**

`p_rate`            A numeric that represents the rate of events  
`p_dimnames`        A character that represents the name of the dimension

**Value**

An object of class DISTRIBUTION, EXPONENTIAL

**Author(s)**

John J. Aponte

**Examples**

```

myDistr <- new_EXPONENTIAL(5)
myDistr$rfunc(10)

```



---

`fitbeta`*Fits a beta distribution based on quantiles*

---

**Description**

Fits a beta distribution based on quantiles

**Usage**

```
fitbeta_ml(point, lci, uci)
```

```
fitbeta(point, lci, uci)
```

**Arguments**

<code>point</code>	Point estimates corresponding to the median
<code>lci</code>	Lower limit (quantile 0.025)
<code>uci</code>	Upper limit (quantile 0.975)

**Value**

parameters `shape1` and `shape2` of a beta distribution

**Functions**

- `fitbeta_ml()`: using ML to estimate parameters
- `fitbeta()`: preserve the expected value

**Note**

This is a wrap of the [fitdist](#) to obtain the best parameters for a beta distribution based on quantiles.

When using confidence intervals (not ML), the shape parameters are obtained using the following formula:

$$varp = (p_{uci} - p_{lci})/4^2$$

$$shape1 = p_{mean} * (p_{mean} * (1 - p_{mean})/varp - 1)$$

$$shape2 = (1 - p_{mean}) * (p_{mean} * (1 - p_{mean})/varp - 1)$$

**Author(s)**

John J. Aponte

**See Also**

[fitdist](#)

**Examples**

```
fitbeta_ml(0.45,0.40,0.50)
fitbeta(0.45,0.40,0.50)
```

---

fitdirichlet                    *Fits a Dirichlet distribution,*

---

**Description**

Fits a Dirichlet distribution based on the parameters of Beta distributions

**Usage**

```
fitdirichlet(..., plotBeta = FALSE, n.fitted = "opt")
```

**Arguments**

...	named vectors with the distribution parameters shape1, shape2
plotBeta	if TRUE a ggplot of the densities are plotted
n.fitted	Method to fit the values

**Details**

Each one of the arguments is a named vector with values for shape1, shape2. Values from [fitbeta](#) are suitable for this. This is a wrap of [fitDirichlet](#)

**Value**

a vector with the parameters for a Dirichlet distribution

**Author(s)**

John J. Aponte

**See Also**

[fitDirichlet](#)

**Examples**

```
a <- fitbeta(0.3, 0.2, 0.4)
c <- fitbeta(0.2, 0.1, 0.3)
b <- fitbeta(0.5, 0.4, 0.6)
fitdirichlet(cat1=a,cat2=b,cat3=c)
```

---

ggDISTRIBUTION      *Plot of DISTRIBUTION objects using ggplot2*

---

**Description**

Plot of [DISTRIBUTION](#) objects using [ggplot2](#)

**Usage**

```
ggDISTRIBUTION(x, n = 10000)
```

**Arguments**

x                    an object of class [DISTRIBUTION](#)  
n                    number of observation

**Value**

a [ggplot](#) object with the density of the distribution

**Examples**

```
x <- new_NORMAL(0, 1)
ggDISTRIBUTION(x)
y <- new_DIRICHLET(c(10, 20, 70))
ggDISTRIBUTION(x)
```

---

LOGNORMAL              *Factory for a LOGNORMAL distribution object*

---

**Description**

Returns a LOGNORMAL distribution object that produce random numbers from a log normal distribution using the [rlnorm](#) function

**Usage**

```
new_LOGNORMAL(p_meanlog, p_sdlog, p_dimnames = "rvar")
```

**Arguments**

p\_meanlog            mean of the distribution on the log scale  
p\_sdlog              A numeric that represents the standard deviation on the log scale  
p\_dimnames          A character that represents the name of the dimension

**Value**

An object of class `DISTRIBUTION`, `LOGNORMAL`

**Author(s)**

John J. Aponte

**Examples**

```
myDistr <- new_LOGNORMAL(0,1)
myDistr$rfunc(10)
```

---

metadata

*Metadata for a DISTRIBUTION*

---

**Description**

Shows the distribution and the oval values of a `DISTRIBUTION` object

**Usage**

```
metadata(x)

## S3 method for class 'DISTRIBUTION'
metadata(x)

## Default S3 method:
metadata(x)
```

**Arguments**

x a `DISTRIBUTION` object

**Value**

A `data.frame` with the metadata of the distributions

**Methods (by class)**

- `metadata(DISTRIBUTION)`: Metadata for `DISTRIBUTION` objects
- `metadata(default)`: Metadata for other objects

**Note**

The number of columns depends on the dimensions of the distribution. There will be one column `distribution` with the name of the distribution and one column for each dimension with the names from the oval field.

**Author(s)**

John J. Aponte

---

NA\_DISTRIBUTION      *Factory for a NA distribution object*

---

**Description**

Returns an NA distribution object that always return NA\_real\_ This is useful to handle NA. By default only one dimension rvar is produced, but if several names are provided more columns will be added to the return matrix

**Usage**

```
new_NA(p_dimnames = "rvar")
```

**Arguments**

p\_dimnames      A character that represents the the names of the dimensions. By default only one dimension with name rvar

**Value**

An object of class DISTRIBUTION, NA

**Author(s)**

John J. Aponte

**Examples**

```
myDistr <- new_NA(p_dimnames = "rvar")
myDistr$rfunc(10)
```

---

new\_MIXTURE      *Mixture of [DISTRIBUTION](#) objects*

---

**Description**

Produce a new distribution that obtain random draws of the mixture of the [DISTRIBUTION](#) objects

**Usage**

```
new_MIXTURE(listdistr, mixture)
```

**Arguments**

`listdistr` a list of [DISTRIBUTION](#) objects

`mixture` a vector of probabilities to mixture the distributions. Must add 1 If missing the draws are obtained from the distributions with the same probability

**Value**

an object of class MIXTURE, [DISTRIBUTION](#)

**Author(s)**

John J. Aponte

**Examples**

```
x1 <- new_NORMAL(0,1)
x2 <- new_NORMAL(4,1)
x3 <- new_NORMAL(6,1)
new_MIXTURE(list(x1,x2,x3))
```

---

new\_MULTINORMAL

*Multivariate Normal Distribution*

---

**Description**

Return a [DISTRIBUTION](#) object that draw random numbers from a multivariate normal distribution using the `mvrnorm` function.

**Usage**

```
new_MULTINORMAL(p_mu, p_sigma, p_dimnames, tol = 1e-06, empirical = FALSE)
```

**Arguments**

`p_mu` a vector of means

`p_sigma` a positive-definite symmetric matrix for the covariance matrix

`p_dimnames` A character that represents the name of the dimension

`tol` tolerance (relative to largest variance) for numerical lack of positive-definiteness in `p_sigma`.

`empirical` logical. If true, `mu` and `Sigma` specify the empirical not population mean and covariance matrix.

**Value**

An object of class [DISTRIBUTION](#), MULTINORMAL

**Author(s)**

John J. Aponte

**See Also**

[mvrnorm](#)

**Examples**

```
msigma <- matrix(c(1,0,0,1), ncol=2)
d1 <- new_MULTINORMAL(c(0,1), msigma)
rfunc(d1, 10)
```

---

NORMAL

*Factory for a NORMAL distribution object*

---

**Description**

Returns a NORMAL distribution object that produce random numbers from a normal distribution using the [rnorm](#) function

**Usage**

```
new_NORMAL(p_mean, p_sd, p_dimnames = "rvar")
```

**Arguments**

<code>p_mean</code>	A numeric that represents the mean value
<code>p_sd</code>	A numeric that represents the standard deviation
<code>p_dimnames</code>	A character that represents the name of the dimension

**Value**

An object of class [DISTRIBUTION](#), NORMAL

**Author(s)**

John J. Aponte

**Examples**

```
myDistr <- new_NORMAL(0,1)
myDistr$rfunc(10)
```

---

omit_NA	<i>Omit NA distributions from a list of distributions</i>
---------	---

---

**Description**

Omit NA distributions from a list of distributions

**Usage**

```
omit_NA(listdistr)
```

**Arguments**

listdistr      a list of [DISTRIBUTION](#) objects

**Value**

the list without the [NA\\_DISTRIBUTION](#)

**Author(s)**

John J. Aponte

---

plot.DISTRIBUTION	<i>plot of <a href="#">DISTRIBUTION</a> objects</i>
-------------------	---

---

**Description**

Plot an histogram of the density of the distribution using random numbers from the distribution

**Usage**

```
## S3 method for class 'DISTRIBUTION'
plot(x, n = 10000, ...)
```

**Arguments**

x                      an object of class [DISTRIBUTION](#)  
n                        number of observations  
...                      other parameters to the [hist](#) function

**Value**

No return value. Side effect plot the histogram.



**Examples**

```
x <- new_NORMAL(0,1)
plot(x)
y <- new_DIRICHLET(c(10,20,70))
plot(x)
```

---

POISSON

*Factory for a POISSON distribution using confidence intervals*

---

**Description**

Returns an POISSON distribution object that produce random numbers from a Poisson distribution using the [rpois](#) function

**Usage**

```
new_POISSON(p_lambda, p_dimnames = "rvar")
```

**Arguments**

`p_lambda`            A numeric that represents the expected number of events  
`p_dimnames`        A character that represents the name of the dimension

**Value**

An object of class DISTRIBUTION, POISSON

**Author(s)**

John J. Aponte

**Examples**

```
myDistr <- new_POISSON(5)
myDistr$rfunc(10)
```

---

restrict\_environment *Build a new function with a smaller environment*

---

### Description

As standard feature, R include in the environment of a function all the variables that are available when the function is created. This, however is prompt to leak reference when you have a factory of function and they are created within a list.. it will include all the component of the list in the function environment. To prevent that, the random generator functions are encapsulated with a restricted environment where only the variables that the function requires to work are included

### Usage

```
restrict_environment(f, ...)
```

### Arguments

f	input function
...	define the set of variables to be included as variable = value.

### Value

new function with a restricted environment

### Author(s)

John J. Aponte

### Examples

```
a = 0
b = 1
myfunc <- restrict_environment(
  function(n) {
    rnorm(meanvalue, sdvalue)
  },
  meanvalue = a, sdvalue = b)

myfunc(10)
ls(envir=environment(myfunc))
```

---

rfunc	<i>Generate random numbers from a <a href="#">DISTRIBUTION</a> object</i>
-------	---

---

**Description**

This is a generic method that calls the rfunc slot of the object

**Usage**

```
rfunc(x, n)
```

**Arguments**

x	an object
n	the number of random samples

**Value**

a matrix with as many rows as n and as many columns as dimensions have distribution

**Author(s)**

John J. Aponte

---

rfunc.default	<i>Default function</i>
---------------	-------------------------

---

**Description**

Default function

**Usage**

```
## Default S3 method:  
rfunc(x, n)
```

**Arguments**

x	an object of class different from <a href="#">DISTRIBUTION</a>
n	the number of random samples

**Value**

No return value. Raise an error message.

**Author(s)**

John J. Aponte

---

`rfunc.DISTRIBUTION`      *Generic function for a [DISTRIBUTION](#) object*

---

**Description**

Generic function for a [DISTRIBUTION](#) object

**Usage**

```
## S3 method for class 'DISTRIBUTION'  
rfunc(x, n)
```

**Arguments**

`x`                    an object of class [DISTRIBUTION](#)  
`n`                    the number of random samples

**Value**

a matrix with as many rows as `n` and as many columns as

**Author(s)**

John J. Aponte

---

`same_dimensions`      *Check the dimensions of a list of distributions*

---

**Description**

Check the dimensions of a list of distributions

**Usage**

```
same_dimensions(listdistr)
```

**Arguments**

`listdistr`           a list of [DISTRIBUTION](#) objects

**Value**

return TRUE if all the dimensions are the same

---

set_seed	<i>Modify a the seed of a Distribution object</i>
----------	---

---

**Description**

This create a new [DISTRIBUTION](#) object but with the specified seed

**Usage**

```
set_seed(distribution, seed)
```

**Arguments**

distribution	a <a href="#">DISTRIBUTION</a> object
seed	the new seed

**Value**

a [DISTRIBUTION](#) object of the same class

**Author(s)**

John J. Aponte

---

summary.DISTRIBUTION	<i>Summary of Distributions</i>
----------------------	---------------------------------

---

**Description**

Summary of Distributions

**Usage**

```
## S3 method for class 'DISTRIBUTION'
summary(object, n = 10000, ...)
```

**Arguments**

object	object of class <a href="#">DISTRIBUTION</a>
n	the number of random samples from the distribution
...	other parameters. Not used

**Value**

A `data.frame` with as many rows as dimensions had the distribution and with the following columns

- distribution name
- varname name of the dimension
- oval value
- nsample number of random samples
- mean\_ mean value of the sample
- sd\_ standard deviation of the sample
- lci\_ lower 95
- median\_ median value of the sample
- uci\_ upper 95

**Note**

The sample uses the seed saved in the object those it will provide the same values for an n value

**Author(s)**

John J. Aponte

---

TRIANGULAR

*Factory for a TRIANGULAR distribution object*

---

**Description**

Returns an TRIANGULAR distribution object that produce random numbers from a triangular distribution using the `rtriang` function

**Usage**

```
new_TRIANGULAR(p_min, p_max, p_mode, p_dimnames = "rvar")
```

**Arguments**

<code>p_min</code>	A numeric that represents the lower limit
<code>p_max</code>	A numeric that represents the upper limit
<code>p_mode</code>	A numeric that represents the mode
<code>p_dimnames</code>	A character that represents the name of the dimension

**Value**

An object of class DISTRIBUTION, TRIANGULAR

**Author(s)**

John J. Aponte

**Examples**

```
myDistr <- new_TRIANGULAR(-1,1,0)
myDistr$rfunc(10)
```

---

TRUNCATED

*Factory for a TRUNCATED distribution object*

---

**Description**

Returns an TRUNCATED distribution object that limits the values that are generated by the distribution to be in the limits `p_min`, `p_max`

**Usage**

```
new_TRUNCATED(p_distribution, p_min = -Inf, p_max = Inf)
```

**Arguments**

`p_distribution` An object of class DISTRIBUTION to truncate  
`p_min` A numeric that set the lower limit of the distribution  
`p_max` A numeric that set the upper limit of the distribution

**Value**

An object of class DISTRIBUTION, `p_distribution$distribution`, TRUNCATED

**Note**

The expected value of a truncated distribution could be very different from the expected value of the unrestricted distribution. Be careful as the `oval` field is not changed and may not represent any more the expected value of the distribution.

If the distribution is multidimensional, the limits will apply to all dimensions.

**Author(s)**

John J. Aponte

**Examples**

```
myDistr <- new_TRUNCATED(p_distribution = new_NORMAL(0,1), p_min = -1, p_max = 1)
myDistr$rfunc(10)
```

---

**UNIFORM***Factory for a UNIFORM distribution object*

---

**Description**

Returns an UNIFORM distribution object that produce random numbers from a uniform distribution using the `runif` function

**Usage**

```
new_UNIFORM(p_min, p_max, p_dimnames = "rvar")
```

**Arguments**

<code>p_min</code>	A numeric that represents the lower limit
<code>p_max</code>	A numeric that represents the upper limit
<code>p_dimnames</code>	A character that represents the name of the dimension

**Value**

An object of class DISTRIBUTION, UNIFORM

**Author(s)**

John J. Aponte

**Examples**

```
myDistr <- new_UNIFORM(0,1)
myDistr$rfunc(10)
```



# Index

- \* **DISTRIBUTION**
  - DISTRIBUTION\_factory, 15
- \*.DISTRIBUTION (CONVOLUTION), 7
- +.DISTRIBUTION (CONVOLUTION), 7
- .DISTRIBUTION (CONVOLUTION), 7
- /.DISTRIBUTION (CONVOLUTION), 7
  
- add\_total, 2
  
- BETA, 3, 14
- BETABINOMIAL, 4
- BINOMIAL, 6
  
- cinqnum, 6
- CONVOLUTION, 7
- CONVOLUTION\_assoc, 9
- CONVOLUTION\_comb, 10
  
- data.frame, 20, 30
- DIRAC, 12, 14
- DIRICHLET, 12, 14
- DISCRETE, 13, 14
- DISTRIBUTION, 2, 3, 6–11, 14, 15, 19–24, 27–29
- DISTRIBUTION\_factory, 15
  
- EXPONENTIAL, 14, 16
  
- fitbeta, 17, 18
- fitbeta\_ml (fitbeta), 17
- fitDirichlet, 18
- fitdirichlet, 18
- fitdist, 17
  
- ggDISTRIBUTION, 19
- ggplot, 19
- ggplot2, 19
  
- hist, 24
  
- LOGNORMAL, 19
  
- metadata, 20
- mvrnorm, 22, 23
  
- NA\_DISTRIBUTION, 8, 14, 21, 24
- new\_BETA (BETA), 3
- new\_BETA\_lci (BETA), 3
- new\_BETA\_lci2 (BETA), 3
- new\_BETABINOMIAL (BETABINOMIAL), 4
- new\_BETABINOMIAL\_icc (BETABINOMIAL), 4
- new\_BETABINOMIAL\_od (BETABINOMIAL), 4
- new\_BINOMIAL (BINOMIAL), 6
- new\_CONVOLUTION (CONVOLUTION), 7
- new\_CONVOLUTION\_assoc (CONVOLUTION\_assoc), 9
- new\_CONVOLUTION\_comb (CONVOLUTION\_comb), 10
- new\_DIRAC (DIRAC), 12
- new\_DIRICHLET (DIRICHLET), 12
- new\_DISCRETE (DISCRETE), 13
- new\_DIVISION (CONVOLUTION), 7
- new\_DIVISION\_assoc (CONVOLUTION\_assoc), 9
- new\_DIVISION\_comb (CONVOLUTION\_comb), 10
- new\_EXPONENTIAL (EXPONENTIAL), 16
- new\_LOGNORMAL (LOGNORMAL), 19
- new\_MIXTURE, 21
- new\_MULTINORMAL, 22
- new\_MULTIPLICATION (CONVOLUTION), 7
- new\_MULTIPLICATION\_assoc (CONVOLUTION\_assoc), 9
- new\_MULTIPLICATION\_comb (CONVOLUTION\_comb), 10
- new\_NA (NA\_DISTRIBUTION), 21
- new\_NORMAL (NORMAL), 23
- new\_POISSON (POISSON), 25
- new\_SUBTRACTION (CONVOLUTION), 7
- new\_SUBTRACTION\_assoc (CONVOLUTION\_assoc), 9
- new\_SUBTRACTION\_comb (CONVOLUTION\_comb), 10

new\_SUM (CONVOLUTION), 7  
new\_SUM\_assoc (CONVOLUTION\_assoc), 9  
new\_SUM\_comb (CONVOLUTION\_comb), 10  
new\_TRIANGULAR (TRIANGULAR), 30  
new\_TRUNCATED (TRUNCATED), 31  
new\_UNIFORM (UNIFORM), 32  
NORMAL, 14, 23

omit\_NA, 24

plot.DISTRIBUTION, 24  
POISSON, 14, 25

rbinom, 4  
rbeta, 3  
rbinom, 6  
rdirichlet, 12  
restrict\_environment, 14, 26  
rexp, 16  
rfunc, 27  
rfunc.default, 27  
rfunc.DISTRIBUTION, 28  
rlnorm, 19  
rnorm, 23  
rpois, 25  
rtriang, 30  
runif, 32

same\_dimensions, 28  
set\_seed, 14, 29  
summary.DISTRIBUTION, 29

TRIANGULAR, 14, 30  
TRUNCATED, 14, 31

UNIFORM, 14, 32