Package: distributions (via r-universe)

November 13, 2026

Type Package

Title What the Package Does (Title Case)

Version 1.0.0

Author Who wrote it

Maintainer The package maintainer <yourself@somewhere.net>

Description More about what it does (maybe more than one line) Use four spaces when indenting paragraphs within the Description.

License What license is it under?

Encoding UTF-8

LazyData true

Collate generics_definition.R support.R transformations.R master_class.R helpers.R empiric_distribution.R smoothed_empiric_distribution.R joint_independent_distributions.R univariate_canonical_distribution.R univariate_canonical_constructors.R canonical_mixture_distribution.R canonical_mixture_constructors.R multivariate_normal_distribution.R multivariate_constructors.R discrete_set_distributions.R

Imports mvtnorm, tmvtnorm, matrixcalc

RoxygenNote 7.1.1

Repository https://cipher-epi.r-universe.dev

Date/Publication 2025-03-11 20:50:03 UTC

RemoteUrl https://github.com/tfojo1/distributions

RemoteRef HEAD

RemoteSha 4d71d9644b4439a59210e804520ac8717ae8f079

Contents

Autoregressive.Multivariate.Normal.Distribution . . . 3
Bernoulli.Distribution . . . 4
Beta.Distribution . . . 5
Binomial.Distribution . . . 6
calculate.cdf . . . 7
calculate.marginal.cdfs . . . 7
calculate.marginal.densities . . . 8
Canonical.Mixture.Distribution . . . 9
Canonical_Mixture_Distribution . . . 10
Compound.Symmetry.Multivariate.Normal.Distribution . . . 11
Constant.Distribution . . . 12
Constant_Distribution . . . 13
Continuous.Support . . . 13
create.transformation . . . 14
Discrete.Set.Distribution . . . 15
Discrete.Set.Support . . . 15
Discrete_Set_Distribution . . . 16
Discrete_Set_Support . . . 16
Distribution . . . 17
distributions.equal . . . 17
do.calculate.density . . . 18
Empiric.Distribution . . . 18
Empiric_Distribution . . . 19
generate.random.marginal.samples . . . 20
generate.random.samples . . . 20
get.covariance.matrix . . . 21
get.defined.transformation . . . 22
get.density.function . . . 22
get.description . . . 23
get.equal.tailed.intervals . . . 23
get.highest.density.intervals . . . 24
get.intervals . . . 25
get.means . . . 26
get.medians . . . 26
get.quantiles . . . 27
get.random.generator.function . . . 28
get.sds . . . 28
get.support.bounds . . . 29
Integer.Support . . . 29
is.supported . . . 30
join.distributions . . . 31
Joint_Independent_Distributions . . . 31
Logitnormal.Distribution . . . 32
Logitnormal.Mixture . . . 33
Logituniform.Distribution . . . 34
Lognormal.Distribution . . . 35

Lognormal.Mixture . . . 36
Loguniform.Distribution . . . 37
Multivariate.Correlated.Uniform.Distribution . . . 37
Multivariate.Logitnormal.Distribution . . . 38
Multivariate.Lognormal.Distribution . . . 39
Multivariate.Normal.Distribution . . . 40
Multivariate.Support . . . 41
Multivariate_Normal_Distribution . . . 41
Multivariate_Support . . . 42
Normal.Distribution . . . 42
Normal.Mixture . . . 43
Range_Support . . . 44
separate.independent.distributions . . . 44
set.var.names . . . 44
Smoothed.Empiric.Distribution . . . 45
Smoothed_Empiric_Distribution . . . 46
subset.distribution . . . 46
subset.support . . . 47
Support . . . 47
Support.Union . . . 47
transformation . . . 48
Transformed.Multivariate.Normal.Distribution . . . 48
Transformed.Normal.Distribution . . . 49
Transformed.Normal.Mixture . . . 50
Uniform.Distribution . . . 51
Union_Support . . . 52
Univariate.Canonical.Distribution . . . 52
Univariate_Canonical_Distribution . . . 53

Index **55**

`Autoregressive.Multivariate.Normal.Distribution`

Create a multivariate normal distribution with a Continuous Autoregressive 1 (CAR-1) correlation structure

---

Description

Creates a distribution where all the variables, after arbitrary transformations, follow a normal distribution with a Continuous Autoregressive 1 (CAR-1) correlation structure

Usage

```
Autoregressive.Multivariate.Normal.Distribution(
  rho,
  times,
  mu = 0,
  sds = 1,
```

```
  lower = -Inf,
  upper = Inf,
  var.names = NULL,
  transformations = NULL
)
```

Arguments

rho	The autoregressive correlation coefficient
times	The times for each observation in the distribution. The correlation between any two observations at times t1 and t2 is rho^(abs(t1-t2))
mu, sds	The mean and standard deviation vectors. If given a scalar value, assumes that the mean/sd is the same for every variable in the distribution
lower	The upper and lower bounds for each variable (on the original scale, NOT the transformed scale). If only one value is given, assumed to apply to all variables in the distribution
upper	The upper and lower bounds for each variable (on the original scale, NOT the transformed scale). If only one value is given, assumed to apply to all variables in the distribution
var.names	The names of the variables in the distribution. If NULL, uses the names of mu, the row names of sigma, or the column names of sigma, in that order
transformations	Objects of class transformation to be applied to each variable. Can be either an object of class transformation, the character name of a defined transformation, or a list of transformation objects. If only one transformation is supplied, assumed to apply to all variables. If this is a named list, any variables missing transformation are assmed to not be transformed

See Also

Other Distribution Constructors: Bernoulli.Distribution(), Beta.Distribution(), Binomial.Distribution(), Canonical.Mixture.Distribution(), Compound.Symmetry.Multivariate.Normal.Distribution(), Constant.Distribution(), Discrete.Set.Distribution(), Empiric.Distribution(), Logitnormal.Distribution(Logitnormal.Mixture(), Logituniform.Distribution(), Lognormal.Distribution(), Lognormal.Mixture(), Loguniform.Distribution(), Multivariate.Correlated.Uniform.Distribution(), Multivariate.Logitnormal.Di Multivariate.Lognormal.Distribution(), Multivariate.Normal.Distribution(), Normal.Distribution(), Normal.Mixture(), Smoothed.Empiric.Distribution(), Transformed.Multivariate.Normal.Distribution(), Transformed.Normal.Distribution(), Transformed.Normal.Mixture(), Uniform.Distribution(), Univariate.Canonical.Distribution(), join.distributions()

Bernoulli.Distribution

Create a Univariate Bernoulli Distribution Object

Description

Create a Univariate Bernoulli Distribution Object

Usage

```
Bernoulli.Distribution(p = 0.5, var.name = NULL)
```

Arguments

p	The probability of success in the distribution
var.name	The name of the single variable in the distribution, or NULL if no name is specified

See Also

Other Univariate Canonical Distribution Constructors: Beta.Distribution(), Binomial.Distribution(), Canonical.Mixture.Distribution(), Logitnormal.Distribution(), Logituniform.Distribution(), Lognormal.Distribution(), Loguniform.Distribution(), Normal.Distribution(), Transformed.Normal.Distribu Uniform.Distribution(), Univariate.Canonical.Distribution()

Other Distribution Constructors: Autoregressive.Multivariate.Normal.Distribution(), Beta.Distribution(), Binomial.Distribution(), Canonical.Mixture.Distribution(), Compound.Symmetry.Multivariate.Normal.Distr Constant.Distribution(), Discrete.Set.Distribution(), Empiric.Distribution(), Logitnormal.Distribution(Logitnormal.Mixture(), Logituniform.Distribution(), Lognormal.Distribution(), Lognormal.Mixture(), Loguniform.Distribution(), Multivariate.Correlated.Uniform.Distribution(), Multivariate.Logitnormal.D Multivariate.Lognormal.Distribution(), Multivariate.Normal.Distribution(), Normal.Distribution(), Normal.Mixture(), Smoothed.Empiric.Distribution(), Transformed.Multivariate.Normal.Distribution(), Transformed.Normal.Distribution(), Transformed.Normal.Mixture(), Uniform.Distribution(), Univariate.Canonical.Distribution(), join.distributions()

Beta.Distribution *Create a Univariate Beta Distribution Object*

Description

Create a Univariate Beta Distribution Object

Usage

```
Beta.Distribution(alpha = 1, beta = 1, lower = 0, upper = 1, var.name = NULL)
```

Arguments

alpha, beta	The parameters to the beta distribution (AKA shape1, shape2 respectively in dbeta)
lower, upper	The lower and the upper bounds
var.name	The name of the single variable in the distribution, or NULL if no name is specified

See Also

Other Univariate Canonical Distribution Constructors: Bernoulli.Distribution(), Binomial.Distribution(), Canonical.Mixture.Distribution(), Logitnormal.Distribution(), Logituniform.Distribution(), Lognormal.Distribution(), Loguniform.Distribution(), Normal.Distribution(), Transformed.Normal.Distribu Uniform.Distribution(), Univariate.Canonical.Distribution()

Other Distribution Constructors: Autoregressive.Multivariate.Normal.Distribution(), Bernoulli.Distribution(Binomial.Distribution(), Canonical.Mixture.Distribution(), Compound.Symmetry.Multivariate.Normal.Distr Constant.Distribution(), Discrete.Set.Distribution(), Empiric.Distribution(), Logitnormal.Distribution(Logitnormal.Mixture(), Logituniform.Distribution(), Lognormal.Distribution(), Lognormal.Mixture(), Loguniform.Distribution(), Multivariate.Correlated.Uniform.Distribution(), Multivariate.Logitnormal.D Multivariate.Lognormal.Distribution(), Multivariate.Normal.Distribution(), Normal.Distribution(), Normal.Mixture(), Smoothed.Empiric.Distribution(), Transformed.Multivariate.Normal.Distribution(), Transformed.Normal.Distribution(), Transformed.Normal.Mixture(), Uniform.Distribution(), Univariate.Canonical.Distribution(), join.distributions()

Binomial.Distribution *Create a Univariate Binomial Distribution Object*

Description

Create a Univariate Binomial Distribution Object

Usage

```
Binomial.Distribution(n = 10, p = 0.5, lower = 0, upper = n, var.name = NULL)
```

Arguments

n, p	The size and probability parameters to the distribution (AKA size, prob in dbinom)
lower, upper	The lower and the upper bounds
var.name	The name of the single variable in the distribution, or NULL if no name is specified

See Also

Other Univariate Canonical Distribution Constructors: Bernoulli.Distribution(), Beta.Distribution(), Canonical.Mixture.Distribution(), Logitnormal.Distribution(), Logituniform.Distribution(), Lognormal.Distribution(), Loguniform.Distribution(), Normal.Distribution(), Transformed.Normal.Distribu Uniform.Distribution(), Univariate.Canonical.Distribution()

Other Distribution Constructors: Autoregressive.Multivariate.Normal.Distribution(), Bernoulli.Distribution(Beta.Distribution(), Canonical.Mixture.Distribution(), Compound.Symmetry.Multivariate.Normal.Distribu Constant.Distribution(), Discrete.Set.Distribution(), Empiric.Distribution(), Logitnormal.Distribution(Logitnormal.Mixture(), Logituniform.Distribution(), Lognormal.Distribution(), Lognormal.Mixture(), Loguniform.Distribution(), Multivariate.Correlated.Uniform.Distribution(), Multivariate.Logitnormal.D Multivariate.Lognormal.Distribution(), Multivariate.Normal.Distribution(), Normal.Distribution(),

Normal.Mixture(), Smoothed.Empiric.Distribution(), Transformed.Multivariate.Normal.Distribution(), Transformed.Normal.Distribution(), Transformed.Normal.Mixture(), Uniform.Distribution(), Univariate.Canonical.Distribution(), join.distributions()

`calculate.cdf` *Calculate the cumulative distribution function for a distribution*

Description

Calculate the cumulative distribution function for a distribution

Usage

```
calculate.cdf(dist, q, lower.tail = T, log.p = F, n.sim = 1000 * dist@n.var)
```

Arguments

`dist`	An object of class Distribution or a subclass
`n.sim`	If the values are to be calculated by random sample generation, how many random samples to generate
`x`	The values of the parameters at which to calculate the CDF. May be either a vector, if the CDF at a single point is desired, or a matrix where each row represents one point and each column represents a variable in the distribution
`log`	A logical indicating whether to return the CDF on the log scale

Details

Note to developers: this function should not be overridden. Override `do.calculate.cdf` instead

Value

If x is a vector, a scalar density. If x is a matrix, a numeric vector of densities, one for each row in x

---

`calculate.marginal.cdfs` *Calculate a marginal cumulative distribution function for each variable in a distribution*

Description

Calculate a marginal cumulative distribution function for each variable in a distribution

Usage

```
calculate.marginal.cdfs(dist, q, lower.tail = T, log.p = F, n.sim = 1000)
```

Arguments

dist	An object of class Distribution or a subclass
n.sim	If the values are to be calculated by random sample generation, how many random samples to generate

Details

Note to developers: this function should not be overridden. Override do.calculate.marginal.cdfs instead

Value

If q is a vector, a vector of probabilities with one value for each variable in the distribution. If q is a matrix, a matrix of probabilites, with one for each row in x and one column for each variable in the distribution

---

calculate.marginal.densities

Calculate a marginal density for each variable in a distribution

---

Description

Calculate a marginal density for each variable in a distribution

Usage

```
calculate.marginal.densities(dist, x, log = F, n.sim = 1000)
```

Arguments

dist	An object of class Distribution or a subclass
x	The values of the parameters at which to calculate the density. May be either a vector, if the density at a single point is desired, or a matrix where each row represents one point and each column represents a variable in the distribution
log	A logical indicating whether to return the density on the log scale
n.sim	If the values are to be calculated by random sample generation, how many random samples to generate

Details

Note to developers: this function should not be overridden. Override do.calculate.marginal.densities instead

Value

If x is a vector or a matrix with only one row, a vector of densities with one value for each variable in the distribution. If x is a matrix, a matrix of densities, with one for each row in x and one column for each variable in the distribution

`Canonical.Mixture.Distribution` *Create a mixture distribution object based off of a canonical, univariate distribution*

Usage

```
Canonical.Mixture.Distribution(
  name,
  dist.name,
  parameters,
  weights = 1,
  var.name = NULL,
  support = Continuous.Support(),
  density.function.name = paste0("d", dist.name),
  cdf.function.name = paste0("p", dist.name),
  quantile.function.name = paste0("q", dist.name),
  transformation = NULL,
  is.improper = F,
  mean.values = as.numeric(NA),
  variance.values = as.numeric(NA)
)
```

Arguments

`name`	A short, descriptive name for the distribution
`dist.name`	The root name of the distribution as used by d/r/p functions in R. For example, to create a normal distribution, dist.name='norm' (like dnorm, rnorm, pnorm). To create a beta distribution, dist.name='beta' (dbeta, rbeta, qbeta). This argument is only necessary if density.function, cdf.function, and quantile.function are not passed explicitly
`parameters`	A list of vectors, where each element has parameter values that would be passed to d/r/p functions in R. For example, for a mixture of normal(0,1) and normal(1,1) distributions, parameters=list(mean=c(0,1), sd=1) or parameters=list(mean=c(0,1), sd=c(1,1))
`weights`	A weight for each component distribution in the mixture. The weights need not sum to 1. If a scalar is passed, all components are given the same weight
`var.name`	The name of the variable in this distribution
`transformation`	A transformation object, if the named distribution operates on a transformation of the random variable (can also pass NULL for no transformation or the name of a predefined transformation - see `get.defined.transformation`). For example, to create a log-normal distribution, use dist.name='norm' with transformation='log
`is.improper`, `is.discrete`	Logicals indicating whether the distribution is improper/discrete

`mean.values`, `variance.values`
: If known, vectors of values for the means and variances of each component of the distribution

`lower.bound`, `upper.bound`
: If this is a bounded distribution, the lower and upper bounds

`density.function`, `cdf.function`, `quantile.function`
: The functions that calculate the density, cdf, and quantile respectively. These should take the standard parameters of such functions in R. density.function should take 'x', distribution-specific parameters, and 'log'. cdf.function should take 'q', distribution-specific parameters, 'lower.tail' and 'log.p'. quantile.function should take 'p', distribution-specific parameters, and 'lower.tail' and 'log.p'

See Also

`get.defined.transformation`, `create.transformation`

Other Univariate Canonical Distribution Constructors: `Bernoulli.Distribution()`, `Beta.Distribution()`, `Binomial.Distribution()`, `Logitnormal.Distribution()`, `Logituniform.Distribution()`, `Lognormal.Distribution()`, `Loguniform.Distribution()`, `Normal.Distribution()`, `Transformed.Normal.Distribu`
`Uniform.Distribution()`, `Univariate.Canonical.Distribution()`

Other Distribution Constructors: `Autoregressive.Multivariate.Normal.Distribution()`, `Bernoulli.Distribution(`
`Beta.Distribution()`, `Binomial.Distribution()`, `Compound.Symmetry.Multivariate.Normal.Distribution()`, `Constant.Distribution()`, `Discrete.Set.Distribution()`, `Empiric.Distribution()`, `Logitnormal.Distribution(`
`Logitnormal.Mixture()`, `Logituniform.Distribution()`, `Lognormal.Distribution()`, `Lognormal.Mixture()`, `Loguniform.Distribution()`, `Multivariate.Correlated.Uniform.Distribution()`, `Multivariate.Logitnormal.D`
`Multivariate.Lognormal.Distribution()`, `Multivariate.Normal.Distribution()`, `Normal.Distribution()`, `Normal.Mixture()`, `Smoothed.Empiric.Distribution()`, `Transformed.Multivariate.Normal.Distribution()`, `Transformed.Normal.Distribution()`, `Transformed.Normal.Mixture()`, `Uniform.Distribution()`, `Univariate.Canonical.Distribution()`, `join.distributions()`

Canonical_Mixture_Distribution

The Canonical_Mixture_Distribution class

Description

A class that represents a univariate mixture of different instances of a canonical distribution (the same distribution, but with different parameters)

Slots

`name` A short, descriptive name for the distribution

`n.components` The number of component distributions in this mixture

`weights` The weights for the component distributions in the mixture (sum(weights)=1)

`density.function.name`,`cdf.function.name`,`quantile.function.name` Names of the (elsewhere defined) functions that calculate density, cdf, and quantiles (eg 'dnorm', 'pnorm', and 'qnorm)

`parameters` A list of vectors; each element of the list represents a parameter to the distribution, and the values of the vector represent the parameter values for each component of the mixture

`p.in.bounds,log.p.in.bounds` A vector cumulative probabilities and of the log cumulative probabilities between the lower and upper bounds for each component of the mixture

`p.lower.bound` A vector of values of the cdf evaluated at the lower bound for each components of the distribution

`transformation` An object of class transformation, defining the transformation applied to the variable before applying the distribution (eg,)

`means,variances` If known, the means and variances of each component of the distribution. May be NA

See Also

The family of functions that create specific mixture distributions, such as normal.mixture

---

`Compound.Symmetry.Multivariate.Normal.Distribution`

Create a multivariate normal distribution with a Compound Symmetry correlation structure

---

Description

Creates a distribution where all the variables, after arbitrary transformations, follow a normal distribution with a Compound Symmetry (AKA Exchangeable) correlation structure

Usage

```
Compound.Symmetry.Multivariate.Normal.Distribution(
  rho,
  n = max(length(mu), length(sds)),
  mu = 0,
  sds = 1,
  lower = -Inf,
  upper = Inf,
  var.names = NULL,
  transformations = NULL
)
```

Arguments

`rho`	The correlation coefficient
`n`	The number of variables in the distribution
`mu, sds`	The mean and standard deviation vectors. If given a scalar value, assumes that the mean/sd is the same for every variable in the distribution

lower — The upper and lower bounds for each variable (on the original scale, NOT the transformed scale). If only one value is given, assumed to apply to all variables in the distribution

upper — The upper and lower bounds for each variable (on the original scale, NOT the transformed scale). If only one value is given, assumed to apply to all variables in the distribution

var.names — The names of the variables in the distribution. If NULL, uses the names of mu, the row names of sigma, or the column names of sigma, in that order

transformations — Objects of class transformation to be applied to each variable. Can be either an object of class transformation, the character name of a defined transformation, or a list of transformation objects. If only one transformation is supplied, assumed to apply to all variables. If this is a named list, any variables missing transformation are assmed to not be transformed

See Also

Other Distribution Constructors: Autoregressive.Multivariate.Normal.Distribution(), Bernoulli.Distribution(), Beta.Distribution(), Binomial.Distribution(), Canonical.Mixture.Distribution(), Constant.Distribution(), Discrete.Set.Distribution(), Empiric.Distribution(), Logitnormal.Distribution(), Logitnormal.Mixture(), Logituniform.Distribution(), Lognormal.Distribution(), Lognormal.Mixture(), Loguniform.Distribution(), Multivariate.Correlated.Uniform.Distribution(), Multivariate.Logitnormal.Distribution(), Multivariate.Lognormal.Distribution(), Multivariate.Normal.Distribution(), Normal.Distribution(), Normal.Mixture(), Smoothed.Empiric.Distribution(), Transformed.Multivariate.Normal.Distribution(), Transformed.Normal.Distribution(), Transformed.Normal.Mixture(), Uniform.Distribution(), Univariate.Canonical.Distribution(), join.distributions()

Constant.Distribution *Create a distribution representing a Constant Distribution*

Description

Creates an object representing a degenerate distribution that only supports a single, discrete value

Usage

```
Constant.Distribution(value, var.name = NULL)
```

Arguments

value — The single value supported by the distribution

var.name — The name of the single variable in the distribution, or NULL if no name is specified

See Also

Discrete.Set.Distribution

Other Discrete Distribution Constructors: Discrete.Set.Distribution()

Other Distribution Constructors: Autoregressive.Multivariate.Normal.Distribution(), Bernoulli.Distribution(), Beta.Distribution(), Binomial.Distribution(), Canonical.Mixture.Distribution(), Compound.Symmetry.Multi Discrete.Set.Distribution(), Empiric.Distribution(), Logitnormal.Distribution(), Logitnormal.Mixture(), Logituniform.Distribution(), Lognormal.Distribution(), Lognormal.Mixture(), Loguniform.Distribution(), Multivariate.Correlated.Uniform.Distribution(), Multivariate.Logitnormal.Distribution(), Multivariate.Lognormal.Distribution(), Multivariate.Normal.Distribution(), Normal.Distribution(), Normal.Mixture(), Smoothed.Empiric.Distribution(), Transformed.Multivariate.Normal.Distribution(), Transformed.Normal.Distribution(), Transformed.Normal.Mixture(), Uniform.Distribution(), Univariate.Canonical.Distribution(), join.distributions()

Constant_Distribution *The Constant_Distribution class*

Description

A class that represents a probility distribution with support for exactly one value

Slots

`value` The single value supported by this distribution

See Also

The Discrete.Set.Distribution

Continuous.Support *Create an object which represents support over a range of integers*

Description

Create an object which represents support over a range of integers

Usage

```
Continuous.Support(
  lower = -Inf,
  upper = Inf,
  lower.inclusive = F,
  upper.inclusive = F
)
```

Arguments

`lower`, `upper` The lower and upper bounds of support. For a single range of support, these should both be of length 1. If support exists over multiple disjoint ranges, then lower and upper should be numeric vectors of the same length, with the lower and upper bounds of each of the disjoint ranges

`lower.inclusive`, `upper.inclusive`
Logical vectors indicating whether the lower and upper bounds are inclusive (vs. exclusive). If length(lower)/length(upper) are >1, these can be either logical vectors of the same length, or logical scalars (in which case the value applies to each element of lower and upper)

Value

An object of class Range_Support, such that is.supported is true if x falls within any of the ranges lower[i] to upper[i]

`create.transformation` *Create an object representing a transformation*

Description

Create an object representing a transformation

Usage

```
create.transformation(
  transform.function,
  reverse.transform.function,
  transformation.derivative,
  log.abs.transformation.derivative = NULL,
  name
)
```

Arguments

`transform.function`, `reverse.transform.function`, `transformation.derivative`, `log.transformation.derivative`
Functions that compute the transformation, inverse transformation, derivative of the transformation function, and log of the derivative of the transformation function, directly. All should take one argument, 'x', which can be either a numeric vector or numeric scalar. log.transformation.derivative may be supplied as NULL; the other three functions must be defined

`name` A character value with a descriptive name of the transformation

`Discrete.Set.Distribution` *Create a distribution representing a Discrete Set*

Description

Creates an object representing a discrete probability distribution over an arbitrary set of numbers

Usage

```
Discrete.Set.Distribution(values, weights, var.name = NULL)
```

Arguments

`values`	The numeric set of values
`weights`	The weight given to the corresponding value. Need not sum to 1
`var.name`	The name of the single variable in the distribution, or NULL if no name is specified

See Also

Constant.Distribution

Other Discrete Distribution Constructors: `Constant.Distribution()`

Other Distribution Constructors: `Autoregressive.Multivariate.Normal.Distribution()`, `Bernoulli.Distribution()`, `Beta.Distribution()`, `Binomial.Distribution()`, `Canonical.Mixture.Distribution()`, `Compound.Symmetry.Multi`, `Constant.Distribution()`, `Empiric.Distribution()`, `Logitnormal.Distribution()`, `Logitnormal.Mixture()`, `Logituniform.Distribution()`, `Lognormal.Distribution()`, `Lognormal.Mixture()`, `Loguniform.Distribution()`, `Multivariate.Correlated.Uniform.Distribution()`, `Multivariate.Logitnormal.Distribution()`, `Multivariate.Lognormal.Distribution()`, `Multivariate.Normal.Distribution()`, `Normal.Distribution()`, `Normal.Mixture()`, `Smoothed.Empiric.Distribution()`, `Transformed.Multivariate.Normal.Distribution()`, `Transformed.Normal.Distribution()`, `Transformed.Normal.Mixture()`, `Uniform.Distribution()`, `Univariate.Canonical.Distribution()`, `join.distributions()`

`Discrete.Set.Support` *Create a support over an arbitrary set of values*

Description

Create a support over an arbitrary set of values

Usage

```
Discrete.Set.Support(supported.values)
```

Arguments

`supported.values`
: A numeric vector of the allowed values in this support

Value

An object of class Discrete_Set_Support

`Discrete_Set_Distribution` *The Discrete_Set_Distribution class*

Description

A class that represents a discrete probability distribution over an arbitrary set of numbers

Slots

`values` The numeric set of values

`weights` The probability of each value in the set. Sum to 1

See Also

The Constant.Distribution

`Discrete_Set_Support` *An object representing support over an arbitrary set of real numbers*

Description

An object representing support over an arbitrary set of real numbers

Distribution *The abstract superclass for all Distribution objects*

Description

Distribution objects represent a distribution of one or more variables

Slots

`var.names` A character vector of the names of the variables in the distribution (optional)

`n.var` The number of variables in the distribution

`support` A Support object defining the support for each variable in the distribution

`is.improper` A logical vectors indicating whether each variable in the distribution is improper

See Also

Specific subclasses which can be instantiated:

- Empiric_Distribution
- Univariate_Canonical_Distribution
- Joint_Independent_Distributions

distributions.equal *Tests whether two Distribution objects represent the same distribution*

Description

Evaluates to true if the two distributions produce the same random samples

Usage

```
distributions.equal(dist1, dist2)
```

Arguments

`dist1`, `dist2` The distributions to compare

Details

The variable order must be the same for two multivariate distributions, otherwise the equality test will return false

Value

True if the two distributions produce the same random samples

do.calculate.density *Internal function for subclasses to calculate the density for a distribution*

Description

This function is provided to be overriden by developers. End users should instead call `calculate.density`

Usage

```
do.calculate.density(dist, x, log = F, n.sim = 1000)
```

Arguments

`dist`	An object of class Distribution or a subclass
`x`	A matrix where each row represents one point and each column represents a variable in the distribution (the wrapper matches variable names before passing to this function)
`log`	A logical indicating whether to return the density on the log scale
`n.sim`	If the values are to be calculated by random sample generation, how many random samples to generate

Details

The wrapper `calculate.density` takes care of matching variable names and of formatting the return value

Value

If x is a vector, a scalar density. If x is a matrix, a numeric vector of densities, one for each row in x

Empiric.Distribution *Create an empiric distribution from a set of samples*

Description

This is a convenience alias for the formal constructor, new('Empiric_Distribution', ...)

Usage

```
Empiric.Distribution(points, weights = 1, var.names = NULL)
```

Arguments

points	A matrix of values, where each row represents one sample and each column represents one variable. If a numeric vector is passed, assumed to be samples from a univariate distribution
weights	A numeric vector of weights, one for each row in points
var.names	The names of the variables in the distribution. If NULL, uses the column names from points

See Also

Other Distribution Constructors: Autoregressive.Multivariate.Normal.Distribution(), Bernoulli.Distribution(), Beta.Distribution(), Binomial.Distribution(), Canonical.Mixture.Distribution(), Compound.Symmetry.Multi, Constant.Distribution(), Discrete.Set.Distribution(), Logitnormal.Distribution(), Logitnormal.Mixture(), Logituniform.Distribution(), Lognormal.Distribution(), Lognormal.Mixture(), Loguniform.Distribution(), Multivariate.Correlated.Uniform.Distribution(), Multivariate.Logitnormal.D, Multivariate.Lognormal.Distribution(), Multivariate.Normal.Distribution(), Normal.Distribution(), Normal.Mixture(), Smoothed.Empiric.Distribution(), Transformed.Multivariate.Normal.Distribution(), Transformed.Normal.Distribution(), Transformed.Normal.Mixture(), Uniform.Distribution(), Univariate.Canonical.Distribution(), join.distributions()

Empiric_Distribution *The Empiric_Distribution Class*

Description

A class that defines a distribution based on a set of samples from that distribution. Treats the samples as coming from a discrete distribution. For a continuous representation, see Smoothed_Empiric_Distribution

Slots

points A matrix containing the samples

weights A numeric vector with the weight given to each row in points. Sum to 1

n.points The number of rows in points

See Also

Smoothed_Empiric_Distribution, create.empiric.distribution

generate.random.marginal.samples

Generate marginal random samples for each variable in a distribution

Description

Generates random samples for each variable in the distribution, but does NOT preserve the correlations across variables within samples. For some distribution implementations, this can be substantially faster than generating true, multivariate random samples, so this method is preferred if random samples are only needed for one variable at a time

Usage

```
generate.random.marginal.samples(dist, n)
```

Arguments

dist	An object of class Distribution or a subclass
n	The number of random samples to be generated

Details

Note to developers: this function should not be overridden. Override do.generate.random.marginal.samples instead

Value

If n==1, a numeric vector with one value for each variable in the distribution. If there is only one variable in the distribution, a numeric vector with n values. If n>1 and there is more than one variable in the distribution, a matrix with n rows, where each row represents one random sample, and each column represents a variable in the distribution

generate.random.samples

Generate random samples from a distribution

Description

Generate random samples from a distribution

Usage

```
generate.random.samples(dist, n)
```

Arguments

dist	An object of class Distribution or a subclass
n	The number of random samples to be generated

Details

Note to developers: This function should not be overridden directly. Insteady override do.generate.random.samples

Value

If n==1, a numeric vector with one value for each variable in the distribution. If there is only one variable in the distribution, a numeric vector with n values. If n>1 and there is more than one variable in the distribution, a matrix with n rows, where each row represents one random sample, and each column represents a variable in the distribution

---

get.covariance.matrix *Get the variance-covariance matrix for variables in a distribution*

Description

Get the variance-covariance matrix for variables in a distribution

Usage

```
get.covariance.matrix(dist, n.sim = 1000)
```

Arguments

dist	An object of class Distribution or a subclass
n.sim	If the values are to be calculated by random sample generation, how many random samples to generate

Details

Note to developers: this function should not be overridden. Override do.get.covariance.matrix instead

Value

A square matrix of dimension number variables in distribution representing the variance-covariance matrix for the distribution

get.defined.transformation *Get Pre-Defined Transformation Objects by Name*

Description

Gets a pre-defined transformation object. The available transformations are 'log', 'logit', and 'identity'

Usage

```
get.defined.transformation(transformation.names, throw.error.if.no.match = F)
```

Value

If transformation.names is a single character value, returns an object of class transformation. Otherwise, returns a list of transformation objects

get.density.function *Get a function that calculates the density for the distribution*

Description

A convenience function that packages up a call to `calculate.density` on this distribution into its own stand-alone function (which could be passed along as an argument to other functions)

Usage

```
get.density.function(dist, default.log = T, n.sim = 1000)
```

Arguments

`dist`	An object of class Distribution or a subclass
`default.log`	Whether the returned function should calculate the density on the log scale (by default, the returned function can have a log parameter set explicitly)
`n.sim`	If the values are to be calculated by random sample generation, how many random samples to generate

Value

A function that takes two arguments. The first, mandatory, is 'x' - the values of the parameters at which to calculate the density. May be either a vector, if the density at a single point is desired, or a matrix where each row represents one point and each column represents a variable in the distribution The second, optional, is 'log' - whether to calculate the density on the log scale. By default, log is set to default.log

get.description *Get the description of a Distribution object*

Description

Get the description of a Distribution object

Usage

```
get.description(object)
```

Arguments

object	The object to describe

get.equal.tailed.intervals *Get equal-tailed intervals (confidence/credible intervals) for the variables in a distribution*

Description

Get equal-tailed intervals (confidence/credible intervals) for the variables in a distribution

Usage

```
get.equal.tailed.intervals(dist, coverage = 0.95, n.sim = 1000)
```

Arguments

dist	An object of class Distribution or a subclass
coverage	A vector of probabilities (from 0 to 1) which the intervals should cover
n.sim	If the values are to be calculated by random sample generation, how many random samples to generate

Details

Note to developers: This function should not be overridden directly. Insteady override do.get.equal.tailed.intervals

Value

If length(coverage)==1, and the distribution has one variable, returns a numeric vector with two values: the lower and upper bound of the interval. If length(coverage)==1 and the distribution has more than one variable, returns a matrix with one row for the lower bound and one row for the upper, and one column for each variable. If length(coverage)>1 and the distribution has one variable, returns a matrix with one row for the lower bound and one row for the upper bound, and one column for each value in coverage. If length(coverage)>1, and the distribution has more than one variable returns a 3-dimensional array; the first dimension has values 'lower' and 'upper'. The second dimension has one value for each variable. The third dimension has one value for each value of coverage.

See Also

`get.equal.tailed.intervals`, `get.highest.density.intervals`

`get.highest.density.intervals` *Get highest-density intervals (confidence/credible intervals) for the variables in a distribution*

Description

Get highest-density intervals (confidence/credible intervals) for the variables in a distribution

Usage

```
get.highest.density.intervals(dist, coverage = 0.95, n.sim = 1000)
```

Arguments

`dist`	An object of class Distribution or a subclass
`coverage`	A vector of probabilities (from 0 to 1) which the intervals should cover
`n.sim`	If the values are to be calculated by random sample generation, how many random samples to generate

Details

Note to developers: This function should not be overridden directly. Insteady override `do.get.highest.density.interval`

Value

If length(coverage)==1, and the distribution has one variable, returns a numeric vector with two values: the lower and upper bound of the interval. If length(coverage)==1 and the distribution has more than one variable, returns a matrix with one row for the lower bound and one row for the upper, and one column for each variable. If length(coverage)>1 and the distribution has one variable, returns a matrix with one row for the lower bound and one row for the upper bound, and one column for each value in coverage. If length(coverage)>1, and the distribution has more than

one variable returns a 3-dimensional array; the first dimension has values 'lower' and 'upper'. The second dimension has one value for each variable. The third dimension has one value for each value of coverage.

See Also

get.equal.tailed.intervals, get.highest.density.intervals

`get.intervals` *Get intervals (confidence/credible intervals) for the variables in a distribution*

Description

Get intervals (confidence/credible intervals) for the variables in a distribution

Usage

```
get.intervals(
  dist,
  type = c("equal-tailed", "highest-density")[1],
  coverage = 0.95,
  n.sim = 1000
)
```

Arguments

`dist`	An object of class Distribution or a subclass
`type`	Either 'equal-tailed' or 'highest-density' depending on the type of interval desired (equal-tailed is generally easier to compute)
`coverage`	A vector of probabilities (from 0 to 1) which the intervals should cover
`n.sim`	If the values are to be calculated by random sample generation, how many random samples to generate

Details

Note to developers: this function should not be overridden. Override do.get.equal.tailed.intervals and do.get.highest.density.intervals instead

Value

If length(coverage)==1, and the distribution has one variable, returns a numeric vector with two values: the lower and upper bound of the interval. If length(coverage)==1 and the distribution has more than one variable, returns a matrix with one row for the lower bound and one row for the upper, and one column for each variable. If length(coverage)>1 and the distribution has one variable, returns a matrix with one row for the lower bound and one row for the upper bound, and one column for each value in coverage. If length(coverage)>1, and the distribution has more than

one variable returns a 3-dimensional array; the first dimension has values 'lower' and 'upper'. The second dimension has one value for each variable. The third dimension has one value for each value of coverage.

See Also

`get.equal.tailed.intervals`, `get.highest.density.intervals`

`get.means` *Get means of the variables in a distribution*

Description

Get means of the variables in a distribution

Usage

```
get.means(dist, n.sim = 1000)
```

Arguments

`dist`	An object of class Distribution or a subclass
`n.sim`	If the values are to be calculated by random sample generation, how many random samples to generate

Details

Note to developers: this function should not be overridden. Override `do.get.means` instead

Value

A numeric vector with a mean for each variable in the distribution

---

`get.medians` *Get medians of the variables in a distribution*

Description

Get medians of the variables in a distribution

Usage

```
get.medians(dist, n.sim = 1000)
```

Arguments

dist	An object of class Distribution or a subclass
n.sim	If the values are to be calculated by random sample generation, how many random samples to generate

Details

Note to developers: This function should not be overridden directly. Insteady override do.get.medians

Value

A numeric vector with a median for each variable in the distribution

---

get.quantiles *Get quantiles for the variables in a distribution*

Description

Get quantiles for the variables in a distribution

Usage

```
get.quantiles(dist, p, lower.tail = T, log.p = F, n.sim = 1000)
```

Arguments

dist	An object of class Distribution or a subclass
p	A vector of probabilities for which to get the quantiles
lower.tail	If true, the returned values are the greatest values x for each variable such that $P(X \leq x) \geq p$. If fale, the returned values are the smallest values x for each variable such that $P(X > x) \geq p$. This can be either a scalar (in which case the value is applied to all quantiles), or a vector with one logical value for each value of p
n.sim	If the values are to be calculated by random sample generation, how many random samples to generate

Details

Note to developers: This function should not be overridden directly. Insteady override do.get.quantiles

Value

If there is only one variable in the distribution, a vector of quantiles for each value of p. If p is length one, a vector of quantiles for each variable in the distribution. If there is more than one variable in the distribution and p>1, a matrix with one row for each value of p, where each column represents a variable in the distribution

get.random.generator.function

Get a function that generates random samples form the distribution

Description

A convenience function that packages up a call to `generate.random.samples` on this distribution into its own stand-alone function (which could be passed along as an argument to other functions)

Usage

```
get.random.generator.function(dist)
```

Arguments

`dist`	An object of class Distribution or a subclass

Value

A function that takes one argument, 'n' - the number of random samples to generate

get.sds

Get the standard deviations of the variables in a distribution

Description

Get the standard deviations of the variables in a distribution

Usage

```
get.sds(dist, n.sim = 1000)
```

Arguments

`dist`	An object of class Distribution or a subclass
`n.sim`	If the values are to be calculated by random sample generation, how many random samples to generate

Details

Note to developers: this function should not be overridden. Override `do.get.sds` instead

Value

A numeric vector with a standard deviation for each variable in the distribution

get.support.bounds — *Get the maximum and minimm values within a support over real numbers*

Description

Get the maximum and minimm values within a support over real numbers

Usage

```
get.support.bounds(support)
```

Arguments

support	An object of class Support

Value

If called on an object of class Support, returns a numeric vector of length 2, with lower and upper bounds. If called on a list containing only Support objects, returns a matrix with two rows (lower and upper bounds) and a column for each element in the list

Integer.Support — *Create an object which represents support over a range of integers*

Description

Create an object which represents support over a range of integers

Usage

```
Integer.Support(lower, upper, lower.inclusive = T, upper.inclusive = T)
```

Arguments

lower	The lower and upper bounds of support. For a single range of support, these should both be of length 1. If support exists over multiple disjoint ranges, then lower and upper should be numeric vectors of the same length, with the lower and upper bounds of each of the disjoint ranges
upper	The lower and upper bounds of support. For a single range of support, these should both be of length 1. If support exists over multiple disjoint ranges, then lower and upper should be numeric vectors of the same length, with the lower and upper bounds of each of the disjoint ranges

`lower.inclusive`
: Logical vectors indicating whether the lower and upper bounds are inclusive (vs. exclusive). If length(lower)/length(upper) are >1, these can be either logical vectors of the same length, or logical scalars (in which case the value applies to each element of lower and upper)

`upper.inclusive`
: Logical vectors indicating whether the lower and upper bounds are inclusive (vs. exclusive). If length(lower)/length(upper) are >1, these can be either logical vectors of the same length, or logical scalars (in which case the value applies to each element of lower and upper)

Value

An object of class Range_Support, such that is.supported is true if x falls within any of the ranges lower[i] to upper[i]

`is.supported` *Test if numeric values fall within a defined support*

Description

Test if numeric values fall within a defined support

Usage

```
is.supported(support, x, by.variable = F)
```

Arguments

`support`	An object of class Support
`x`	If the support is univariate, a numeric vector. If the support is multivariate, a matrix with one column for each variable and one row for each point to test or a numeric vector with one value for each variable in the support
`by.variable`	If this is a Multivariate_Support, if by.variable is TRUE, calculates support for each variable independently

Value

If the support is univariate or if by.variable==FALSE or if only a vector is passed to x, returns a logical vector, with one value for each value of x, corresponding to whether that value of x falls within the defined support. If the support is multivariate and by.variable==TRUE and a matrix with more than one row was passed to x, returns a matrix with one column for each variable and one row for each point

`join.distributions` *Join Multiple Distributions*

Description

Joins multiple Distribution objects into a single Distribution object, on the premise that the variables in each sub-distribution are independent of the variables in any other sub-distribution

Usage

```
join.distributions(..., var.names = NULL)
```

Arguments

`...`	One or more Distribution objects or lists of Distribution objects to be joined. If var.names are set on the component distributions, those names will be preserved. If not, and if the elements of ... are named (or elements in lists contained in ... are named), those names will be used as the var.names for the corresponding sub-distributions

Value

An object of class Joint_Independent_Distributions

See Also

Other Distribution Constructors: `Autoregressive.Multivariate.Normal.Distribution()`, `Bernoulli.Distribution()`, `Beta.Distribution()`, `Binomial.Distribution()`, `Canonical.Mixture.Distribution()`, `Compound.Symmetry.Multi`, `Constant.Distribution()`, `Discrete.Set.Distribution()`, `Empiric.Distribution()`, `Logitnormal.Distribution()`, `Logitnormal.Mixture()`, `Logituniform.Distribution()`, `Lognormal.Distribution()`, `Lognormal.Mixture()`, `Loguniform.Distribution()`, `Multivariate.Correlated.Uniform.Distribution()`, `Multivariate.Logitnormal.D`, `Multivariate.Lognormal.Distribution()`, `Multivariate.Normal.Distribution()`, `Normal.Distribution()`, `Normal.Mixture()`, `Smoothed.Empiric.Distribution()`, `Transformed.Multivariate.Normal.Distribution()`, `Transformed.Normal.Distribution()`, `Transformed.Normal.Mixture()`, `Uniform.Distribution()`, `Univariate.Canonical.Distribution()`

`Joint_Independent_Distributions`
The Joint_Independent_Distributions class

Description

A class that joins multiple independent sub-distributions into a joint distribution

Slots

`subdistributions` A list of the Distribution objects representing the component distributions of this joint distribution

`indices.for.subdistributions` A list whose elements are integer vectors, such that indices.for.subdistributions[[i]] is the indices into the overall distribution of the variables represented by subdistributions[[i]]

`n.subdistributions` The number of component distributions

See Also

join.distributions

`Logitnormal.Distribution` *Create a Univariate Logit-Normal Distribution Object*

Description

Create a Univariate Logit-Normal Distribution Object

Usage

```
Logitnormal.Distribution(
  meanlogit,
  sdlogit,
  lower = 0,
  upper = 1,
  var.name = NULL
)
```

Arguments

`meanlogit`, `sdlogit`	The mean and standard deviation (on the logit scale) of the distribution
`lower`, `upper`	The lower and the upper bounds (on the original scale, NOT the logit scale), if this is a bounded distribution
`var.name`	The name of the single variable in the distribution, or NULL if no name is specified

See Also

Other Univariate Canonical Distribution Constructors: `Bernoulli.Distribution()`, `Beta.Distribution()`, `Binomial.Distribution()`, `Canonical.Mixture.Distribution()`, `Logituniform.Distribution()`, `Lognormal.Distribution()`, `Loguniform.Distribution()`, `Normal.Distribution()`, `Transformed.Normal.Distribu` `Uniform.Distribution()`, `Univariate.Canonical.Distribution()`

Other Distribution Constructors: `Autoregressive.Multivariate.Normal.Distribution()`, `Bernoulli.Distribution(` `Beta.Distribution()`, `Binomial.Distribution()`, `Canonical.Mixture.Distribution()`, `Compound.Symmetry.Multi`

Constant.Distribution(), Discrete.Set.Distribution(), Empiric.Distribution(), Logitnormal.Mixture(), Logituniform.Distribution(), Lognormal.Distribution(), Lognormal.Mixture(), Loguniform.Distribution(), Multivariate.Correlated.Uniform.Distribution(), Multivariate.Logitnormal.Distribution(), Multivariate.Lognormal.Distribution(), Multivariate.Normal.Distribution(), Normal.Distribution(), Normal.Mixture(), Smoothed.Empiric.Distribution(), Transformed.Multivariate.Normal.Distribution(), Transformed.Normal.Distribution(), Transformed.Normal.Mixture(), Uniform.Distribution(), Univariate.Canonical.Distribution(), join.distributions()

`Logitnormal.Mixture` *Create a Mixture Univariate Logit-Normal Distributions*

Description

Create a Mixture Univariate Logit-Normal Distributions

Usage

```
Logitnormal.Mixture(
  meanslogit,
  sdslogit,
  weights = 1,
  lower = 0,
  upper = 1,
  var.name = NULL
)
```

Arguments

`meanslogit, sdslogit`	The means and standard deviations (on the logit scale) of the components of the distribution
`weights`	The weights applied to each component of the mixture. Weights need not be normalized to sum to 1. If a scalar value is given, all components of the mixture will have the same weight
`lower, upper`	The lower and the upper bounds (on the original scale, NOT the logit scale), if this is a bounded distribution (the same bounds apply for all components of the mixture)

See Also

Other Canonical Mixture Distribution Constructors: Lognormal.Mixture(), Normal.Mixture(), Transformed.Normal.Mixture()

Other Distribution Constructors: Autoregressive.Multivariate.Normal.Distribution(), Bernoulli.Distribution(), Beta.Distribution(), Binomial.Distribution(), Canonical.Mixture.Distribution(), Compound.Symmetry.Multi, Constant.Distribution(), Discrete.Set.Distribution(), Empiric.Distribution(), Logitnormal.Distribution(), Logituniform.Distribution(), Lognormal.Distribution(), Lognormal.Mixture(), Loguniform.Distribution(), Multivariate.Correlated.Uniform.Distribution(), Multivariate.Logitnormal.Distribution(),

Multivariate.Lognormal.Distribution(), Multivariate.Normal.Distribution(), Normal.Distribution(), Normal.Mixture(), Smoothed.Empiric.Distribution(), Transformed.Multivariate.Normal.Distribution(), Transformed.Normal.Distribution(), Transformed.Normal.Mixture(), Uniform.Distribution(), Univariate.Canonical.Distribution(), join.distributions()

Logituniform.Distribution

Create a Logit-Uniform Distribution Object

Description

Create a Logit-Uniform Distribution Object

Usage

```
Logituniform.Distribution(min = 0, max = 1, var.name = NULL)
```

Arguments

min, max	The bounds of the distribution (on the original scale, NOT the logit scale). If min is 0 or max is 1, returns an improper logit-uniform distribution (which returns 1/x/(1-x) for all densities, but cannot evaluate cdfs, quantiles, or generate random samples)
var.name	The name of the single variable in the distribution, or NULL if no name is specified

See Also

Other Univariate Canonical Distribution Constructors: Bernoulli.Distribution(), Beta.Distribution(), Binomial.Distribution(), Canonical.Mixture.Distribution(), Logitnormal.Distribution(), Lognormal.Distribution(), Loguniform.Distribution(), Normal.Distribution(), Transformed.Normal.Distribu Uniform.Distribution(), Univariate.Canonical.Distribution()

Other Distribution Constructors: Autoregressive.Multivariate.Normal.Distribution(), Bernoulli.Distribution(Beta.Distribution(), Binomial.Distribution(), Canonical.Mixture.Distribution(), Compound.Symmetry.Multi Constant.Distribution(), Discrete.Set.Distribution(), Empiric.Distribution(), Logitnormal.Distribution(Logitnormal.Mixture(), Lognormal.Distribution(), Lognormal.Mixture(), Loguniform.Distribution(), Multivariate.Correlated.Uniform.Distribution(), Multivariate.Logitnormal.Distribution(), Multivariate.Lognormal.Distribution(), Multivariate.Normal.Distribution(), Normal.Distribution(), Normal.Mixture(), Smoothed.Empiric.Distribution(), Transformed.Multivariate.Normal.Distribution(), Transformed.Normal.Distribution(), Transformed.Normal.Mixture(), Uniform.Distribution(), Univariate.Canonical.Distribution(), join.distributions()

`Lognormal.Distribution` *Create a Univariate Log-Normal Distribution Object*

Description

Create a Univariate Log-Normal Distribution Object

Usage

```
Lognormal.Distribution(
  meanlog = 0,
  sdlog = 1,
  lower = 0,
  upper = Inf,
  var.name = NULL
)
```

Arguments

`meanlog, sdlog`	The mean and standard deviation (on the log scale) of the distribution
`lower, upper`	The lower and the upper bounds (on the original scale, NOT the log scale), if this is a bounded distribution
`var.name`	The name of the single variable in the distribution, or NULL if no name is specified

See Also

Other Univariate Canonical Distribution Constructors: `Bernoulli.Distribution()`, `Beta.Distribution()`, `Binomial.Distribution()`, `Canonical.Mixture.Distribution()`, `Logitnormal.Distribution()`, `Logituniform.Distribution()`, `Loguniform.Distribution()`, `Normal.Distribution()`, `Transformed.Normal.Distr` `Uniform.Distribution()`, `Univariate.Canonical.Distribution()`

Other Distribution Constructors: `Autoregressive.Multivariate.Normal.Distribution()`, `Bernoulli.Distribution(` `Beta.Distribution()`, `Binomial.Distribution()`, `Canonical.Mixture.Distribution()`, `Compound.Symmetry.Multi` `Constant.Distribution()`, `Discrete.Set.Distribution()`, `Empiric.Distribution()`, `Logitnormal.Distribution(` `Logitnormal.Mixture()`, `Logituniform.Distribution()`, `Lognormal.Mixture()`, `Loguniform.Distribution()`, `Multivariate.Correlated.Uniform.Distribution()`, `Multivariate.Logitnormal.Distribution()`, `Multivariate.Lognormal.Distribution()`, `Multivariate.Normal.Distribution()`, `Normal.Distribution()`, `Normal.Mixture()`, `Smoothed.Empiric.Distribution()`, `Transformed.Multivariate.Normal.Distribution()`, `Transformed.Normal.Distribution()`, `Transformed.Normal.Mixture()`, `Uniform.Distribution()`, `Univariate.Canonical.Distribution()`, `join.distributions()`

`Lognormal.Mixture` *Create a Mixture of Univariate Log-Normal Distributions*

Description

Create a Mixture of Univariate Log-Normal Distributions

Usage

```
Lognormal.Mixture(
  meanslog = 0,
  sdslog = 1,
  weights = 1,
  lower = 0,
  upper = Inf,
  var.name = NULL
)
```

Arguments

`meanslog`, `sdslog`	The means and standard deviations (on the log scale) of the components of distribution
`weights`	The weights applied to each component of the mixture. Weights need not be normalized to sum to 1. If a scalar value is given, all components of the mixture will have the same weight
`lower`, `upper`	The lower and the upper bounds (on the original scale, NOT the log scale), if this is a bounded distribution (the same bounds apply for all components of the mixture)

See Also

Other Canonical Mixture Distribution Constructors: `Logitnormal.Mixture()`, `Normal.Mixture()`, `Transformed.Normal.Mixture()`

Other Distribution Constructors: `Autoregressive.Multivariate.Normal.Distribution()`, `Bernoulli.Distribution()`, `Beta.Distribution()`, `Binomial.Distribution()`, `Canonical.Mixture.Distribution()`, `Compound.Symmetry.Multivariate.Correlated.Uniform.Distribution()`, `Constant.Distribution()`, `Discrete.Set.Distribution()`, `Empiric.Distribution()`, `Logitnormal.Distribution()`, `Logitnormal.Mixture()`, `Logituniform.Distribution()`, `Lognormal.Distribution()`, `Loguniform.Distribution()`, `Multivariate.Correlated.Uniform.Distribution()`, `Multivariate.Logitnormal.Distribution()`, `Multivariate.Lognormal.Distribution()`, `Multivariate.Normal.Distribution()`, `Normal.Distribution()`, `Normal.Mixture()`, `Smoothed.Empiric.Distribution()`, `Transformed.Multivariate.Normal.Distribution()`, `Transformed.Normal.Distribution()`, `Transformed.Normal.Mixture()`, `Uniform.Distribution()`, `Univariate.Canonical.Distribution()`, `join.distributions()`

Loguniform.Distribution

Create a Log-Uniform Distribution Object

Description

Create a Log-Uniform Distribution Object

Usage

```
Loguniform.Distribution(min = 0, max = 1, var.name = NULL)
```

Arguments

min, max	The bounds of the distribution (on the original scale, NOT the log scale). If min is 0 or max is Inf, returns an improper log-uniform distribution (which returns 1/x for all densities, but cannot evaluate cdfs, quantiles, or generate random samples)
var.name	The name of the single variable in the distribution, or NULL if no name is specified

See Also

Other Univariate Canonical Distribution Constructors: Bernoulli.Distribution(), Beta.Distribution(), Binomial.Distribution(), Canonical.Mixture.Distribution(), Logitnormal.Distribution(), Logituniform.Distribution(), Lognormal.Distribution(), Normal.Distribution(), Transformed.Normal.Distri Uniform.Distribution(), Univariate.Canonical.Distribution()

Other Distribution Constructors: Autoregressive.Multivariate.Normal.Distribution(), Bernoulli.Distribution(Beta.Distribution(), Binomial.Distribution(), Canonical.Mixture.Distribution(), Compound.Symmetry.Multi Constant.Distribution(), Discrete.Set.Distribution(), Empiric.Distribution(), Logitnormal.Distribution(Logitnormal.Mixture(), Logituniform.Distribution(), Lognormal.Distribution(), Lognormal.Mixture(), Multivariate.Correlated.Uniform.Distribution(), Multivariate.Logitnormal.Distribution(), Multivariate.Lognormal.Distribution(), Multivariate.Normal.Distribution(), Normal.Distribution(), Normal.Mixture(), Smoothed.Empiric.Distribution(), Transformed.Multivariate.Normal.Distribution(), Transformed.Normal.Distribution(), Transformed.Normal.Mixture(), Uniform.Distribution(), Univariate.Canonical.Distribution(), join.distributions()

Multivariate.Correlated.Uniform.Distribution

Create a Multivariate Uniform Distribution Object Where the Variables are Correlated

Description

Uses a rough approximation to generate a multivariate normal distribution such that the correlations between variables are approximately equal to the given correlation.matrix. Specifically, generates random samples from a multivariate normal distribution with mean=0 and covariance matrix = cov2cor(correlation.matrix + 0.04*I)

Usage

```
Multivariate.Correlated.Uniform.Distribution(
  correlation.matrix,
  min = rep(0, dim(correlation.matrix)[1]),
  max = rep(1, dim(correlation.matrix)[1]),
  var.names = NULL
)
```

Arguments

`correlation.matrix`	The matrix specifying how variables should be correlated
`min, max`	The vectors of minima and maxima for each variable in the distribution. If given a scalar value, assumes that value applies for all variables
`var.names`	The names of the variables in the distribution. If NULL, uses the names of mu, the row names of sigma, or the column names of sigma, in that order

See Also

Other Distribution Constructors: `Autoregressive.Multivariate.Normal.Distribution()`, `Bernoulli.Distribution()`, `Beta.Distribution()`, `Binomial.Distribution()`, `Canonical.Mixture.Distribution()`, `Compound.Symmetry.Multivariate.Normal.Distribution()`, `Constant.Distribution()`, `Discrete.Set.Distribution()`, `Empiric.Distribution()`, `Logitnormal.Distribution()`, `Logitnormal.Mixture()`, `Logituniform.Distribution()`, `Lognormal.Distribution()`, `Lognormal.Mixture()`, `Loguniform.Distribution()`, `Multivariate.Logitnormal.Distribution()`, `Multivariate.Lognormal.Distribution()`, `Multivariate.Normal.Distribution()`, `Normal.Distribution()`, `Normal.Mixture()`, `Smoothed.Empiric.Distribution()`, `Transformed.Multivariate.Normal.Distribution()`, `Transformed.Normal.Distribution()`, `Transformed.Normal.Mixture()`, `Uniform.Distribution()`, `Univariate.Canonical.Distribution()`, `join.distributions()`

Multivariate.Logitnormal.Distribution

Create a Multivariate Logit-Normal Distribution Object

Description

Create a Multivariate Logit-Normal Distribution Object

Usage

```
Multivariate.Logitnormal.Distribution(
  mu = rep(0, nrow(sigma)),
  sigma = diag(rep(1, length(mu))),
  lower = 0,
  upper = Inf,
  var.names = NULL
)
```

Arguments

mu, sigma	The mean vector and covariance matrix of the distribution (after logit transformation)
lower, upper	The upper and lower bounds for each variable (on the original scale, NOT the logit scale). If only one value is given, assumed to apply to all variables in the distribution
var.names	The names of the variables in the distribution. If NULL, uses the names of mu, the row names of sigma, or the column names of sigma, in that order

See Also

Other Distribution Constructors: Autoregressive.Multivariate.Normal.Distribution(), Bernoulli.Distribution(
Beta.Distribution(), Binomial.Distribution(), Canonical.Mixture.Distribution(), Compound.Symmetry.Multi
Constant.Distribution(), Discrete.Set.Distribution(), Empiric.Distribution(), Logitnormal.Distribution(
Logitnormal.Mixture(), Logituniform.Distribution(), Lognormal.Distribution(), Lognormal.Mixture(),
Loguniform.Distribution(), Multivariate.Correlated.Uniform.Distribution(), Multivariate.Lognormal.Dis
Multivariate.Normal.Distribution(), Normal.Distribution(), Normal.Mixture(), Smoothed.Empiric.Distribut
Transformed.Multivariate.Normal.Distribution(), Transformed.Normal.Distribution(),
Transformed.Normal.Mixture(), Uniform.Distribution(), Univariate.Canonical.Distribution(),
join.distributions()

Multivariate.Lognormal.Distribution

Create a Multivariate Log-Normal Distribution Object

Description

Create a Multivariate Log-Normal Distribution Object

Usage

```
Multivariate.Lognormal.Distribution(
  mu = rep(0, nrow(sigma)),
  sigma = diag(rep(1, length(mu))),
  lower = 0,
  upper = Inf,
  var.names = NULL
)
```

Arguments

mu, sigma	The mean vector and covariance matrix of the distribution (after log transformation)
lower, upper	The upper and lower bounds for each variable (on the original scale, NOT the log scale). If only one value is given, assumed to apply to all variables in the distribution
var.names	The names of the variables in the distribution. If NULL, uses the names of mu, the row names of sigma, or the column names of sigma, in that order

See Also

Other Distribution Constructors: Autoregressive.Multivariate.Normal.Distribution(), Bernoulli.Distribution(), Beta.Distribution(), Binomial.Distribution(), Canonical.Mixture.Distribution(), Compound.Symmetry.Multi, Constant.Distribution(), Discrete.Set.Distribution(), Empiric.Distribution(), Logitnormal.Distribution(), Logitnormal.Mixture(), Logituniform.Distribution(), Lognormal.Distribution(), Lognormal.Mixture(), Loguniform.Distribution(), Multivariate.Correlated.Uniform.Distribution(), Multivariate.Logitnormal.D, Multivariate.Normal.Distribution(), Normal.Distribution(), Normal.Mixture(), Smoothed.Empiric.Distribut, Transformed.Multivariate.Normal.Distribution(), Transformed.Normal.Distribution(), Transformed.Normal.Mixture(), Uniform.Distribution(), Univariate.Canonical.Distribution(), join.distributions()

Multivariate.Normal.Distribution

Create a Multivariate Normal Distribution Object

Description

Create a Multivariate Normal Distribution Object

Usage

```
Multivariate.Normal.Distribution(
  mu = rep(0, nrow(sigma)),
  sigma = diag(rep(1, length(mu))),
  lower = -Inf,
  upper = Inf,
  var.names = NULL
)
```

Arguments

mu, sigma	The mean vector and covariance matrix of the distribution
lower, upper	The upper and lower bounds for each variable, if bounded. If only one value is given, assumed to apply to all variables in the distribution
var.names	The names of the variables in the distribution. If NULL, uses the names of mu, the row names of sigma, or the column names of sigma, in that order

See Also

Other Distribution Constructors: Autoregressive.Multivariate.Normal.Distribution(), Bernoulli.Distribution(), Beta.Distribution(), Binomial.Distribution(), Canonical.Mixture.Distribution(), Compound.Symmetry.Multi, Constant.Distribution(), Discrete.Set.Distribution(), Empiric.Distribution(), Logitnormal.Distribution(, Logitnormal.Mixture(), Logituniform.Distribution(), Lognormal.Distribution(), Lognormal.Mixture(), Loguniform.Distribution(), Multivariate.Correlated.Uniform.Distribution(), Multivariate.Logitnormal.D, Multivariate.Lognormal.Distribution(), Normal.Distribution(), Normal.Mixture(), Smoothed.Empiric.Distri, Transformed.Multivariate.Normal.Distribution(), Transformed.Normal.Distribution(), Transformed.Normal.Mixture(), Uniform.Distribution(), Univariate.Canonical.Distribution(), join.distributions()

Multivariate.Support *Join multiple Support objects into a multivariate Support*

Description

Join multiple Support objects into a multivariate Support

Usage

```
Multivariate.Support(...)
```

Arguments

...	Objects of class Support, or lists that contain only objects of class Support

Value

A Multivariate_Support object

Multivariate_Normal_Distribution *The Multivariate_Normal_Distribution Class*

Description

Defines a multivariate normal distribution

Slots

mu,sigma	The mean vector and covariance matrix of the distribution

Multivariate_Support *An object representing support (on the real numbers) over across multiple variables*

Description

An object representing support (on the real numbers) over across multiple variables

Normal.Distribution *Create a Univariate Normal Distribution Object*

Description

Create a Univariate Normal Distribution Object

Usage

```
Normal.Distribution(
  mean = 0,
  sd = 1,
  lower = -Inf,
  upper = Inf,
  var.name = NULL
)
```

Arguments

mean, sd	The mean and standard deviation of the distribution
lower, upper	The lower and the upper bounds, if this is a bounded distribution
var.name	The name of the single variable in the distribution, or NULL if no name is specified

See Also

Other Univariate Canonical Distribution Constructors: Bernoulli.Distribution(), Beta.Distribution(), Binomial.Distribution(), Canonical.Mixture.Distribution(), Logitnormal.Distribution(), Logituniform.Distribution(), Lognormal.Distribution(), Loguniform.Distribution(), Transformed.Normal.Distribution(), Uniform.Distribution(), Univariate.Canonical.Distribution()

Other Distribution Constructors: Autoregressive.Multivariate.Normal.Distribution(), Bernoulli.Distribution(, Beta.Distribution(), Binomial.Distribution(), Canonical.Mixture.Distribution(), Compound.Symmetry.Multi, Constant.Distribution(), Discrete.Set.Distribution(), Empiric.Distribution(), Logitnormal.Distribution(, Logitnormal.Mixture(), Logituniform.Distribution(), Lognormal.Distribution(), Lognormal.Mixture(), Loguniform.Distribution(), Multivariate.Correlated.Uniform.Distribution(), Multivariate.Logitnormal.D, Multivariate.Lognormal.Distribution(), Multivariate.Normal.Distribution(), Normal.Mixture(),

Smoothed.Empiric.Distribution(), Transformed.Multivariate.Normal.Distribution(), Transformed.Normal.Di
Transformed.Normal.Mixture(), Uniform.Distribution(), Univariate.Canonical.Distribution(),
join.distributions()

`Normal.Mixture` *Create a Mixture of Univariate Normal Distributions*

Description

Create a Mixture of Univariate Normal Distributions

Usage

```
Normal.Mixture(
  means = 0,
  sds = 1,
  weights = 1,
  lower = -Inf,
  upper = Inf,
  var.name = NULL
)
```

Arguments

means, sds	The means and standard deviations of each component of the mixture
weights	The weights applied to each component of the mixture. Weights need not be normalized to sum to 1. If a scalar value is given, all components of the mixture will have the same weight
lower, upper	The lower and the upper bounds, if this is a bounded distribution (the same bounds apply for all components of the mixture)

See Also

Other Canonical Mixture Distribution Constructors: Logitnormal.Mixture(), Lognormal.Mixture(),
Transformed.Normal.Mixture()

Other Distribution Constructors: Autoregressive.Multivariate.Normal.Distribution(), Bernoulli.Distribution(
Beta.Distribution(), Binomial.Distribution(), Canonical.Mixture.Distribution(), Compound.Symmetry.Multi
Constant.Distribution(), Discrete.Set.Distribution(), Empiric.Distribution(), Logitnormal.Distribution(
Logitnormal.Mixture(), Logituniform.Distribution(), Lognormal.Distribution(), Lognormal.Mixture(),
Loguniform.Distribution(), Multivariate.Correlated.Uniform.Distribution(), Multivariate.Logitnormal.Di
Multivariate.Lognormal.Distribution(), Multivariate.Normal.Distribution(), Normal.Distribution(),
Smoothed.Empiric.Distribution(), Transformed.Multivariate.Normal.Distribution(), Transformed.Normal.Di
Transformed.Normal.Mixture(), Uniform.Distribution(), Univariate.Canonical.Distribution(),
join.distributions()

Range_Support *An object representing support over a range of real numbers*

Description

An object representing support over a range of real numbers

separate.independent.distributions *Separate a distribution object into a list of distributions that are independent from each other*

Description

Separate a distribution object into a list of distributions that are independent from each other

Usage

```
separate.independent.distributions(object, var.names)
```

Arguments

object	A distribution object

Value

A list of Distribution objects, where each distribution either (a) is univariate or (b) contains only correlated variables

set.var.names *Sets the variable names for a Distribution Object*

Description

Sets the variable names for a Distribution Object

Usage

```
set.var.names(object, var.names)
```

Arguments

var.names	The character vector of variable names

Value

A Distribution object with the new var.names set

`Smoothed.Empiric.Distribution` *Create an smoothed empiric distribution from a set of samples*

Description

Create an smoothed empiric distribution from a set of samples

Usage

```
Smoothed.Empiric.Distribution(
  points,
  weights = 1,
  var.names = NULL,
  bandwidth.multipliers = 1
)
```

Arguments

`points`	A matrix of values, where each row represents one sample and each column represents one variable. If a numeric vector is passed, assumed to be samples from a univariate distribution
`weights`	A numeric vector of weights, one for each row in points
`var.names`	The names of the variables in the distribution. If NULL, uses the column names from points

See Also

Other Distribution Constructors: `Autoregressive.Multivariate.Normal.Distribution()`, `Bernoulli.Distribution()`, `Beta.Distribution()`, `Binomial.Distribution()`, `Canonical.Mixture.Distribution()`, `Compound.Symmetry.Multi`, `Constant.Distribution()`, `Discrete.Set.Distribution()`, `Empiric.Distribution()`, `Logitnormal.Distribution()`, `Logitnormal.Mixture()`, `Logituniform.Distribution()`, `Lognormal.Distribution()`, `Lognormal.Mixture()`, `Loguniform.Distribution()`, `Multivariate.Correlated.Uniform.Distribution()`, `Multivariate.Logitnormal.D`, `Multivariate.Lognormal.Distribution()`, `Multivariate.Normal.Distribution()`, `Normal.Distribution()`, `Normal.Mixture()`, `Transformed.Multivariate.Normal.Distribution()`, `Transformed.Normal.Distribution()`, `Transformed.Normal.Mixture()`, `Uniform.Distribution()`, `Univariate.Canonical.Distribution()`, `join.distributions()`

`Smoothed_Empiric_Distribution` *The Smoothed_Empiric_Distribution Class*

Description

A class that defines a distribution based on a set of samples from that distribution, but applies Gaussian kernel smoothers. For an unsmoothed representation, see Empiric_Distribution

Slots

`points` A matrix containing the samples

`weights` A numeric vector with the weight given to each row in points. Sum to 1

`n.points` The number of rows in points

See Also

Empiric_Distribution, Smoothed.Empiric.Distribution

`subset.distribution` *Get a distribution that describes a subset of the variables in a given distribution*

Description

Get a distribution that describes a subset of the variables in a given distribution

Usage

```
subset.distribution(dist, vars.to.keep)
```

Arguments

`dist`	An object of class Distribution or a subclass
`vars.to.keep`	Either a character vector with the names of the variables to keep, an integer/numeric vector with the indices of the variables to keep, or a logical vector of length n.var indicating which variables to keep

Details

Note to developers: this function should not be overridden. Override `do.subset.distribution` instead

Value

An object of class `Distribution-class`

subset.support *Subset a Support object*

Description

Subset a Support object

Usage

```
subset.support(support, keep)
```

Arguments

support	An object of class Support
keep	Which dimensions of the support to keep. Either a set of integer indices or a logical vector

Value

A Support object

Support *An object representing support over real numbers*

Description

An object representing support over real numbers

Support.Union *Join multiple univariate Support objects into one univariate Support*

Description

Join multiple univariate Support objects into one univariate Support

Usage

```
Support.Union(...)
```

Arguments

...	Objects of class Support, or lists that contain only objects of class Support

Value

A Union_Support object, such that is.supported is true if is.supported is true for any of its component supports

transformation *The transformation class*

Description

A class that represents a functional transformation, along with how to reverse the transformation and take its derivative

Slots

`transform.function,reverse.transform.function,transformation.derivative,log.abs.transformation.deri`
: Functions that compute the transformation, inverse transformation, derivative of the transformation function, and log of the absolute value of the derivative of the transformation function, directly. All should take one argument, 'x', which can be either a numeric vector or numeric scalar. log.abs.transformation.derivative may be supplied as NULL; the other three functions must be defined

`name` A character value with a descriptive name of the transformation

Transformed.Multivariate.Normal.Distribution *Create a transformed multivariate normal distribution object*

Description

Creates a distribution where all the variables, after arbitrary transformations, follow a normal distribution

Usage

```
Transformed.Multivariate.Normal.Distribution(
  mu = rep(0, nrow(sigma)),
  sigma = diag(rep(1, length(mu))),
  lower = -Inf,
  upper = Inf,
  var.names = NULL,
  transformations = NULL
)
```

Arguments

`mu, sigma`	The mean vector and covariance matrix of the distribution (after transformations)
`lower, upper`	The upper and lower bounds for each variable (on the original scale, NOT the transformed scale). If only one value is given, assumed to apply to all variables in the distribution

`var.names` The names of the variables in the distribution. If NULL, uses the names of mu, the row names of sigma, or the column names of sigma, in that order

`transformations`
Objects of class transformation to be applied to each variable. Can be either an object of class transformation, the character name of a defined transformation, or a list of transformation objects. If only one transformation is supplied, assumed to apply to all variables. If this is a named list, any variables missing transformation are assmed to not be transformed

See Also

Other Distribution Constructors: `Autoregressive.Multivariate.Normal.Distribution()`, `Bernoulli.Distribution()`, `Beta.Distribution()`, `Binomial.Distribution()`, `Canonical.Mixture.Distribution()`, `Compound.Symmetry.Multivariate.Normal.Distribution()`, `Constant.Distribution()`, `Discrete.Set.Distribution()`, `Empiric.Distribution()`, `Logitnormal.Distribution()`, `Logitnormal.Mixture()`, `Logituniform.Distribution()`, `Lognormal.Distribution()`, `Lognormal.Mixture()`, `Loguniform.Distribution()`, `Multivariate.Correlated.Uniform.Distribution()`, `Multivariate.Logitnormal.Distribution()`, `Multivariate.Lognormal.Distribution()`, `Multivariate.Normal.Distribution()`, `Normal.Distribution()`, `Normal.Mixture()`, `Smoothed.Empiric.Distribution()`, `Transformed.Normal.Distribution()`, `Transformed.Normal.Mixture()`, `Uniform.Distribution()`, `Univariate.Canonical.Distribution()`, `join.distributions()`

Transformed.Normal.Distribution

Create a Distribution of a Random Variable that is Normal after an arbitrary transformation

---

Description

Create a Distribution of a Random Variable that is Normal after an arbitrary transformation

Usage

```
Transformed.Normal.Distribution(
  mean = 0,
  sd = 1,
  transformation = NULL,
  lower = -Inf,
  upper = Inf,
  var.name = NULL
)
```

Arguments

`mean, sd` The mean and standard deviation of the random variable (after transformation has been applied)

`lower, upper` The lower and the upper bounds, if this is a bounded distribution. The bounds are on the ORIGINAL scale (before transformation).

`var.name` The name of the single variable in the distribution, or NULL if no name is specified

See Also

Other Univariate Canonical Distribution Constructors: `Bernoulli.Distribution()`, `Beta.Distribution()`, `Binomial.Distribution()`, `Canonical.Mixture.Distribution()`, `Logitnormal.Distribution()`, `Logituniform.Distribution()`, `Lognormal.Distribution()`, `Loguniform.Distribution()`, `Normal.Distribution()`, `Uniform.Distribution()`, `Univariate.Canonical.Distribution()`

Other Distribution Constructors: `Autoregressive.Multivariate.Normal.Distribution()`, `Bernoulli.Distribution()`, `Beta.Distribution()`, `Binomial.Distribution()`, `Canonical.Mixture.Distribution()`, `Compound.Symmetry.Multivariate.Normal.Distribution()`, `Constant.Distribution()`, `Discrete.Set.Distribution()`, `Empiric.Distribution()`, `Logitnormal.Distribution()`, `Logitnormal.Mixture()`, `Logituniform.Distribution()`, `Lognormal.Distribution()`, `Lognormal.Mixture()`, `Loguniform.Distribution()`, `Multivariate.Correlated.Uniform.Distribution()`, `Multivariate.Logitnormal.Distribution()`, `Multivariate.Lognormal.Distribution()`, `Multivariate.Normal.Distribution()`, `Normal.Distribution()`, `Normal.Mixture()`, `Smoothed.Empiric.Distribution()`, `Transformed.Multivariate.Normal.Distribution()`, `Transformed.Normal.Mixture()`, `Uniform.Distribution()`, `Univariate.Canonical.Distribution()`, `join.distributions()`

Transformed.Normal.Mixture

Create a Mixture of Transformed Normal Distributions

Description

Creates a mixture of distributions representing random variables that are normal after transformation (eg, mixture of normals, log-normals, logit-normals, etc.)

Usage

```
Transformed.Normal.Mixture(
  means = 0,
  sds = 1,
  weights = 1,
  transformation = NULL,
  lower = -Inf,
  upper = Inf,
  var.name = NULL
)
```

Arguments

`means, sds` The means and standard deviations of each component of the mixture (after transformation has been applied)

`weights` The weights applied to each component of the mixture. Weights need not be normalized to sum to 1. If a scalar value is given, all components of the mixture will have the same weight

`lower, upper` The lower and the upper bounds, if this is a bounded distribution. The bounds are on the ORIGINAL scale (before transformation). The same bounds apply for all components of the mixture

See Also

Other Canonical Mixture Distribution Constructors: `Logitnormal.Mixture()`, `Lognormal.Mixture()`, `Normal.Mixture()`

Other Distribution Constructors: `Autoregressive.Multivariate.Normal.Distribution()`, `Bernoulli.Distribution()`, `Beta.Distribution()`, `Binomial.Distribution()`, `Canonical.Mixture.Distribution()`, `Compound.Symmetry.Multivariate.Normal.Distribution()`, `Constant.Distribution()`, `Discrete.Set.Distribution()`, `Empiric.Distribution()`, `Logitnormal.Distribution()`, `Logitnormal.Mixture()`, `Logituniform.Distribution()`, `Lognormal.Distribution()`, `Lognormal.Mixture()`, `Loguniform.Distribution()`, `Multivariate.Correlated.Uniform.Distribution()`, `Multivariate.Logitnormal.Distribution()`, `Multivariate.Lognormal.Distribution()`, `Multivariate.Normal.Distribution()`, `Normal.Distribution()`, `Normal.Mixture()`, `Smoothed.Empiric.Distribution()`, `Transformed.Multivariate.Normal.Distribution()`, `Transformed.Normal.Distribution()`, `Uniform.Distribution()`, `Univariate.Canonical.Distribution()`, `join.distributions()`

`Uniform.Distribution` *Create a Uniform Distribution Object*

Description

Create a Uniform Distribution Object

Usage

```
Uniform.Distribution(min = 0, max = 1, var.name = NULL)
```

Arguments

`min, max` The bounds of the distribution. If either is infinite, will create an improper uniform distribution (which returns 1 for all densities, but cannot evaluate cdfs, quantiles, or generate random samples)

`var.name` The name of the single variable in the distribution, or NULL if no name is specified

See Also

Other Univariate Canonical Distribution Constructors: `Bernoulli.Distribution()`, `Beta.Distribution()`, `Binomial.Distribution()`, `Canonical.Mixture.Distribution()`, `Logitnormal.Distribution()`, `Logituniform.Distribution()`, `Lognormal.Distribution()`, `Loguniform.Distribution()`, `Normal.Distribution()`, `Transformed.Normal.Distribution()`, `Univariate.Canonical.Distribution()`

Other Distribution Constructors: `Autoregressive.Multivariate.Normal.Distribution()`, `Bernoulli.Distribution()`, `Beta.Distribution()`, `Binomial.Distribution()`, `Canonical.Mixture.Distribution()`, `Compound.Symmetry.Multivariate.Normal.Distribution()`, `Constant.Distribution()`, `Discrete.Set.Distribution()`, `Empiric.Distribution()`, `Logitnormal.Distribution()`, `Logitnormal.Mixture()`, `Logituniform.Distribution()`, `Lognormal.Distribution()`, `Lognormal.Mixture()`,

Loguniform.Distribution(), Multivariate.Correlated.Uniform.Distribution(), Multivariate.Logitnormal.Di
Multivariate.Lognormal.Distribution(), Multivariate.Normal.Distribution(), Normal.Distribution(),
Normal.Mixture(), Smoothed.Empiric.Distribution(), Transformed.Multivariate.Normal.Distribution(),
Transformed.Normal.Distribution(), Transformed.Normal.Mixture(), Univariate.Canonical.Distribution(),
join.distributions()

`Union_Support` *An object representing the union of two or more supports over a range of real numbers*

Description

An object representing the union of two or more supports over a range of real numbers

---

`Univariate.Canonical.Distribution` *Create a distribution object based off of a canonical, univariate distribution*

Usage

```
Univariate.Canonical.Distribution(
  name,
  dist.name,
  parameters,
  var.name = NULL,
  support = Continuous.Support(),
  density.function.name = paste0("d", dist.name),
  cdf.function.name = paste0("p", dist.name),
  quantile.function.name = paste0("q", dist.name),
  transformation = NULL,
  is.improper = F,
  mean.value = as.numeric(NA),
  variance.value = as.numeric(NA)
)
```

Arguments

`name`	A short, descriptive name for the distribution
`dist.name`	The root name of the distribution as used by d/r/p functions in R. For example, to create a normal distribution, dist.name='norm' (like dnorm, rnorm, pnorm). To create a beta distribution, dist.name='beta' (dbeta, rbeta, qbeta). This argument is only necessary if density.function, cdf.function, and quantile.function are not passed explicitly

`parameters` A list of parameters that would be passed to d/r/p functions in R. For example, for a normal(0,1), parameters=list(mean=0, sd=1)

`var.name` The name of the variable in this distribution

`support` A support object that

`transformation` A transformation object, if the named distribution operates on a transformation of the random variable (can also pass NULL for no transformation or the name of a predefined transformation - see `get.defined.transformation`). For example, to create a log-normal distribution, use dist.name='norm' with transformation='log

`is.improper` Logicals indicating whether the distribution is improper/discrete

`mean.value, variance`
If known, these values will be returned for calls to get.means, get.sds, get.covariance.matrix. If passed NA, random sampling will be used to estimate distribution parameters if requested

`density.function, cdf.function, quantile.function`
The functions that calculate the density, cdf, and quantile respectively. These should take the standard parameters of such functions in R. density.function should take 'x', distribution-specific parameters, and 'log'. cdf.function should take 'q', distribution-specific parameters, 'lower.tail' and 'log.p'. quantile.function should take 'p', distribution-specific parameters, and 'lower.tail' and 'log.p'

See Also

`get.defined.transformation`, `create.transformation`

Other Univariate Canonical Distribution Constructors: `Bernoulli.Distribution()`, `Beta.Distribution()`, `Binomial.Distribution()`, `Canonical.Mixture.Distribution()`, `Logitnormal.Distribution()`, `Logituniform.Distribution()`, `Lognormal.Distribution()`, `Loguniform.Distribution()`, `Normal.Distribution()`, `Transformed.Normal.Distribution()`, `Uniform.Distribution()`

Other Distribution Constructors: `Autoregressive.Multivariate.Normal.Distribution()`, `Bernoulli.Distribution()`, `Beta.Distribution()`, `Binomial.Distribution()`, `Canonical.Mixture.Distribution()`, `Compound.Symmetry.Multivariate.Normal.Distribution()`, `Constant.Distribution()`, `Discrete.Set.Distribution()`, `Empiric.Distribution()`, `Logitnormal.Distribution()`, `Logitnormal.Mixture()`, `Logituniform.Distribution()`, `Lognormal.Distribution()`, `Lognormal.Mixture()`, `Loguniform.Distribution()`, `Multivariate.Correlated.Uniform.Distribution()`, `Multivariate.Logitnormal.Distribution()`, `Multivariate.Lognormal.Distribution()`, `Multivariate.Normal.Distribution()`, `Normal.Distribution()`, `Normal.Mixture()`, `Smoothed.Empiric.Distribution()`, `Transformed.Multivariate.Normal.Distribution()`, `Transformed.Normal.Distribution()`, `Transformed.Normal.Mixture()`, `Uniform.Distribution()`, `join.distributions()`

`Univariate_Canonical_Distribution`
The Univariate_Canonical_Distribution class

Description

A class that represents a univariate canonical distribution, such as a normal or beta distribution

Slots

`name` A short, descriptive name for the distribution

`density.function.name,cdf.function.name,quantile.function.name` Names of the (elsewhere defined) functions that calculate density, cdf, and quantiles (eg 'dnorm', 'pnorm', and 'qnorm)

`parameters` A list of parameters that would be passed to density.function, cdf.function, and quantile.function. For example, for a normal(0,1), parameters=list(mean=0, sd=1)

`p.in.bounds,log.p.in.bounds` The cumulative probability and of the log cumulative probability between the lower and upper bounds

`p.lower.bound` The value of the cdf evaluated at the lower bound

`transformation` An object of class transformation, defining the transformation applied to the variable before applying the distribution (eg,)

`mean.value,variance` If known, the mean and variance of the distribution. May be NA

See Also

The family of functions that create specific distributions, such as Normal.Distribution

Index

∗ **Canonical Mixture Distribution Constructors**
Logitnormal.Mixture, 33
Lognormal.Mixture, 36
Normal.Mixture, 43
Transformed.Normal.Mixture, 50
∗ **Discrete Distribution Constructors**
Constant.Distribution, 12
Discrete.Set.Distribution, 15
∗ **Distribution Constructors**
Autoregressive.Multivariate.Normal.Distribution, 3
Bernoulli.Distribution, 4
Beta.Distribution, 5
Binomial.Distribution, 6
Canonical.Mixture.Distribution, 9
Compound.Symmetry.Multivariate.Normal.Distribution, 11
Constant.Distribution, 12
Discrete.Set.Distribution, 15
Empiric.Distribution, 18
join.distributions, 31
Logitnormal.Distribution, 32
Logitnormal.Mixture, 33
Logituniform.Distribution, 34
Lognormal.Distribution, 35
Lognormal.Mixture, 36
Loguniform.Distribution, 37
Multivariate.Correlated.Uniform.Distribution, 37
Multivariate.Logitnormal.Distribution, 38
Multivariate.Lognormal.Distribution, 39
Multivariate.Normal.Distribution, 40
Normal.Distribution, 42
Normal.Mixture, 43
Smoothed.Empiric.Distribution, 45
Transformed.Multivariate.Normal.Distribution, 48
Transformed.Normal.Distribution, 49
Transformed.Normal.Mixture, 50
Uniform.Distribution, 51
Univariate.Canonical.Distribution, 52
∗ **Univariate Canonical Distribution Constructors**
Bernoulli.Distribution, 4
Beta.Distribution, 5
Binomial.Distribution, 6
Canonical.Mixture.Distribution, 9
Logitnormal.Distribution, 32
Logituniform.Distribution, 34
Lognormal.Distribution, 35
Loguniform.Distribution, 37
Normal.Distribution, 42
Transformed.Normal.Distribution, 49
Uniform.Distribution, 51
Univariate.Canonical.Distribution, 52

Autoregressive.Multivariate.Normal.Distribution, 3, *5*, *6*, *10*, *12*, *13*, *15*, *19*, *31–43*, *45*, *49–51*, *53*

Bernoulli.Distribution, 4, *4*, *6*, *10*, *12*, *13*, *15*, *19*, *31–43*, *45*, *49–51*, *53*
Beta.Distribution, *4*, 5, *5*, *6*, *10*, *12*, *13*, *15*, *19*, *31–43*, *45*, *49–51*, *53*
Binomial.Distribution, *4*, *5*, 6, *6*, *10*, *12*, *13*, *15*, *19*, *31–43*, *45*, *49–51*, *53*

calculate.cdf, 7
calculate.density, *18*, *22*
calculate.marginal.cdfs, 7
calculate.marginal.densities, 8

Canonical.Mixture.Distribution, *4–6*, 9, *12, 13*, *15*, *19*, *31–43*, *45*, *49–51*, *53*
Canonical_Mixture_Distribution, 10
Canonical_Mixture_Distribution-class *(*Canonical_Mixture_Distribution*)*, 10
Compound.Symmetry.Multivariate.Normal.Distribution, *4–6*, *10*, 11, *13*, *15*, *19*, *31–43*, *45*, *49–51*, *53*
Constant.Distribution, *4–6*, *10*, 12, *12*, *15, 16*, *19*, *31*, *33–43*, *45*, *49–51*, *53*
Constant_Distribution, 13
Constant_Distribution-class *(*Constant_Distribution*)*, 13
Continuous.Support, 13
create.empiric.distribution, *19*
create.transformation, *10*, 14, *53*

Discrete.Set.Distribution, *4–6*, *10*, *12, 13*, 15, *19*, *31*, *33–43*, *45*, *49–51*, *53*
Discrete.Set.Support, 15
Discrete_Set_Distribution, 16
Discrete_Set_Distribution-class *(*Discrete_Set_Distribution*)*, 16
Discrete_Set_Support, 16, *16*
Discrete_Set_Support-class *(*Discrete_Set_Support*)*, 16
Distribution, 17
Distribution-class *(*Distribution*)*, 17
distributions.equal, 17
do.calculate.cdf, *7*
do.calculate.density, 18
do.calculate.marginal.cdfs, *8*
do.calculate.marginal.densities, *8*
do.generate.random.marginal.samples, *20*
do.generate.random.samples, *21*
do.get.covariance.matrix, *21*
do.get.equal.tailed.intervals, *23*, *25*
do.get.highest.density.intervals, *24, 25*
do.get.means, *26*
do.get.medians, *27*
do.get.quantiles, *27*
do.get.sds, *28*
do.subset.distribution, *46*

Empiric.Distribution, *4–6*, *10*, *12, 13*, *15*, 18, *31*, *33–43*, *45*, *49–51*, *53*
Empiric_Distribution, *17*, 19, *46*
Empiric_Distribution-class *(*Empiric_Distribution*)*, 19

generate.random.marginal.samples, 20
generate.random.samples, 20, *28*
get.covariance.matrix, 21
get.defined.transformation, *9, 10*, 22, *53*
get.density.function, 22
get.description, 23
get.equal.tailed.intervals, 23, *24–26*
get.highest.density.intervals, 24, *24–26*
get.intervals, 25
get.means, 26
get.medians, 26
get.quantiles, 27
get.random.generator.function, 28
get.sds, 28
get.support.bounds, 29

Integer.Support, 29
is.supported, 30

join.distributions, *4–7*, *10*, *12, 13*, *15*, *19*, 31, *32–41*, *43*, *45*, *49–53*
Joint_Independent_Distributions, *17*, 31
Joint_Independent_Distributions-class *(*Joint_Independent_Distributions*)*, 31

Logitnormal.Distribution, *4–6*, *10*, *12, 13*, *15*, *19*, *31*, 32, *33–43*, *45*, *49–51*, *53*
Logitnormal.Mixture, *4–6*, *10*, *12, 13*, *15*, *19*, *31*, 33, *33–43*, *45*, *49–51*, *53*
Logituniform.Distribution, *4–6*, *10*, *12, 13*, *15*, *19*, *31–33*, 34, *35–43*, *45*, *49–51*, *53*
Lognormal.Distribution, *4–6*, *10*, *12, 13*, *15*, *19*, *31–34*, 35, *36–43*, *45*, *49–51*, *53*
Lognormal.Mixture, *4–6*, *10*, *12, 13*, *15*, *19*, *31*, *33–35*, 36, *37–43*, *45*, *49–51*, *53*
Loguniform.Distribution, *4–6*, *10*, *12, 13*, *15*, *19*, *31–36*, 37, *38–43*, *45*, *49–53*

Multivariate.Correlated.Uniform.Distribution, *4–6*, *10*, *12, 13*, *15*, *19*, *31*, *33–36*, 37, *37*, *39–43*, *45*, *49–53*

Multivariate.Logitnormal.Distribution, *4–6*, *10*, *12, 13*, *15*, *19*, *31*, *33–37*, 38, *38*, *40–43*, *45*, *49–53*
Multivariate.Lognormal.Distribution, *4–6*, *10*, *12, 13*, *15*, *19*, *31*, *33–38*, 39, *39*, *41–43*, *45*, *49–53*
Multivariate.Normal.Distribution, *4–6*, *10*, *12, 13*, *15*, *19*, *31*, *33–39*, 40, *40*, *42, 43*, *45*, *49–53*
Multivariate.Support, 41
Multivariate_Normal_Distribution, 41
Multivariate_Normal_Distribution-class *(*Multivariate_Normal_Distribution*)*, 41
Multivariate_Support, *41*, 42
Multivariate_Support-class *(*Multivariate_Support*)*, 42

Normal.Distribution, *4–6*, *10*, *12, 13*, *15*, *19*, *31–41*, 42, *43*, *45*, *49–54*
Normal.Mixture, *4–7*, *10*, *12, 13*, *15*, *19*, *31*, *33–42*, 43, *45*, *49–53*
normal.mixture, *11*

Range_Support, *14*, *30*, 44
Range_Support-class *(*Range_Support*)*, 44

separate.independent.distributions, 44
set.var.names, 44
Smoothed.Empiric.Distribution, *4–7*, *10*, *12, 13*, *15*, *19*, *31*, *33–41*, *43*, 45, *46*, *49–53*
Smoothed_Empiric_Distribution, *19*, 46
Smoothed_Empiric_Distribution-class *(*Smoothed_Empiric_Distribution*)*, 46
subset.distribution, 46
subset.support, 47
Support, *17*, *29, 30*, *41*, 47, *47*
support, *53*
Support-class *(*Support*)*, 47
Support.Union, 47

transformation, *4*, *11, 12*, 48, *49*, *54*
transformation-class *(*transformation*)*, 48
Transformed.Multivariate.Normal.Distribution, *4–7*, *10*, *12, 13*, *15*, *19*, *31*, *33–41*, *43*, *45*, 48, *50–53*
Transformed.Normal.Distribution, *4–7*, *10*, *12, 13*, *15*, *19*, *31–43*, *45*, 49, *49*, *51–53*
Transformed.Normal.Mixture, *4–7*, *10*, *12, 13*, *15*, *19*, *31*, *33–41*, *43*, *45*, *49*, 50, *50*, *52, 53*

Uniform.Distribution, *4–7*, *10*, *12, 13*, *15*, *19*, *31–43*, *45*, *49, 50*, 51, *51*, *53*
Union_Support, *47*, 52
Union_Support-class *(*Union_Support*)*, 52
Univariate.Canonical.Distribution, *4–7*, *10*, *12, 13*, *15*, *19*, *31–43*, *45*, *49–51*, 52, *52*
Univariate_Canonical_Distribution, *17*, 53
Univariate_Canonical_Distribution-class *(*Univariate_Canonical_Distribution*)*, 53