# Package 'algebraic.dist'

March 18, 2026

**Title** Algebra over Probability Distributions

**Version** 1.0.0

**Description** Provides an algebra over probability distributions enabling composition, sampling, and automatic simplification to closed forms. Supports normal, exponential, gamma, Weibull, chi-squared, uniform, beta, log-normal, Poisson, multivariate normal, empirical, and mixture distributions with algebraic operators (addition, subtraction, multiplication, division, power, exp, log, min, max) that automatically simplify when mathematical identities apply. Includes closed-form MVN conditioning (Schur complement), affine transformations, mixture marginals/conditionals (Bayes rule), and limiting distribution builders (CLT, LLN, delta method). Uses S3 classes for distributions and R6 for support objects.

**License** GPL (>= 3)

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**Suggests** knitr, rmarkdown, testthat (>= 3.0.0)

**Config/testthat/edition** 3

**Imports** stats, mvtnorm, R6

**Depends** R (>= 3.5.0)

**VignetteBuilder** knitr

**URL** https://github.com/queelius/algebraic.dist, https://queelius.github.io/algebraic.dist/

**BugReports** https://github.com/queelius/algebraic.dist/issues

**NeedsCompilation** no

**Author** Alexander Towell [aut, cre] (ORCID: <https://orcid.org/0000-0001-6443-9897>)

**Maintainer** Alexander Towell <lex@metafunctor.com>

**Repository** CRAN

**Date/Publication** 2026-03-18 06:13:20 UTC

# Contents

*.dist . . . 7
+.dist . . . 8
-.dist . . . 9
/.dist . . . 9
affine_transform . . . 10
as_dist . . . 11
beta_dist . . . 12
cdf . . . 12
cdf.beta_dist . . . 13
cdf.chi_squared . . . 14
cdf.edist . . . 14
cdf.empirical_dist . . . 15
cdf.exponential . . . 16
cdf.gamma_dist . . . 16
cdf.lognormal . . . 17
cdf.mixture . . . 18
cdf.mvn . . . 18
cdf.normal . . . 19
cdf.poisson_dist . . . 20
cdf.uniform_dist . . . 20
cdf.weibull_dist . . . 21
chi_squared . . . 22
clt . . . 22
conditional . . . 23
conditional.dist . . . 24
conditional.edist . . . 24
conditional.empirical_dist . . . 25
conditional.mixture . . . 26
conditional.mvn . . . 27
countable_set . . . 28
delta_clt . . . 29
density.beta_dist . . . 29
density.chi_squared . . . 30
density.edist . . . 31
density.empirical_dist . . . 31
density.exponential . . . 32
density.gamma_dist . . . 33
density.lognormal . . . 33
density.mixture . . . 34
density.mvn . . . 35
density.normal . . . 35
density.poisson_dist . . . 36
density.uniform_dist . . . 37
density.weibull_dist . . . 37
dim.beta_dist . . . 38
dim.chi_squared . . . 38

dim.countable_set . . . 39
dim.edist . . . 40
dim.empirical_dist . . . 40
dim.exponential . . . 41
dim.finite_set . . . 41
dim.gamma_dist . . . 42
dim.interval . . . 42
dim.lognormal . . . 43
dim.mixture . . . 43
dim.mvn . . . 44
dim.normal . . . 45
dim.poisson_dist . . . 45
dim.uniform_dist . . . 46
dim.weibull_dist . . . 46
edist . . . 47
empirical_dist . . . 47
expectation . . . 48
expectation.dist . . . 49
expectation.empirical_dist . . . 50
expectation.poisson_dist . . . 50
expectation.univariate_dist . . . 51
expectation_data . . . 52
exponential . . . 53
finite_set . . . 54
format.beta_dist . . . 55
format.chi_squared . . . 56
format.edist . . . 56
format.empirical_dist . . . 57
format.exponential . . . 57
format.gamma_dist . . . 58
format.lognormal . . . 58
format.mixture . . . 59
format.mvn . . . 60
format.normal . . . 60
format.poisson_dist . . . 61
format.realized_dist . . . 61
format.uniform_dist . . . 62
format.weibull_dist . . . 62
gamma_dist . . . 63
has . . . 64
has.countable_set . . . 65
has.finite_set . . . 65
has.interval . . . 66
hazard.chi_squared . . . 67
hazard.continuous_dist . . . 67
hazard.exponential . . . 68
hazard.gamma_dist . . . 69
hazard.lognormal . . . 69

hazard.weibull_dist . . . 70
infimum . . . 71
infimum.countable_set . . . 71
infimum.finite_set . . . 72
infimum.interval . . . 72
interval . . . 73
inv_cdf . . . 75
inv_cdf.beta_dist . . . 75
inv_cdf.chi_squared . . . 76
inv_cdf.edist . . . 77
inv_cdf.empirical_dist . . . 77
inv_cdf.exponential . . . 78
inv_cdf.gamma_dist . . . 79
inv_cdf.lognormal . . . 79
inv_cdf.normal . . . 80
inv_cdf.poisson_dist . . . 81
inv_cdf.uniform_dist . . . 81
inv_cdf.weibull_dist . . . 82
is_beta_dist . . . 83
is_chi_squared . . . 83
is_dist . . . 84
is_edist . . . 84
is_empirical_dist . . . 85
is_exponential . . . 85
is_gamma_dist . . . 86
is_lognormal . . . 86
is_mixture . . . 87
is_mvn . . . 87
is_normal . . . 88
is_poisson_dist . . . 88
is_realized_dist . . . 89
is_uniform_dist . . . 89
is_weibull_dist . . . 90
lln . . . 90
lognormal . . . 91
marginal . . . 92
marginal.empirical_dist . . . 92
marginal.mixture . . . 93
marginal.mvn . . . 94
Math.dist . . . 94
mean.beta_dist . . . 95
mean.chi_squared . . . 96
mean.edist . . . 96
mean.empirical_dist . . . 97
mean.exponential . . . 97
mean.gamma_dist . . . 98
mean.lognormal . . . 99
mean.mixture . . . 99

mean.mvn . . . 100
mean.normal . . . 100
mean.poisson_dist . . . 101
mean.uniform_dist . . . 101
mean.univariate_dist . . . 102
mean.weibull_dist . . . 103
mixture . . . 103
mvn . . . 104
nobs.empirical_dist . . . 105
normal . . . 105
normal_approx . . . 106
nparams . . . 106
nparams.empirical_dist . . . 107
nparams.mixture . . . 108
obs . . . 108
obs.empirical_dist . . . 109
params . . . 109
params.beta_dist . . . 110
params.chi_squared . . . 111
params.edist . . . 111
params.empirical_dist . . . 112
params.exponential . . . 112
params.gamma_dist . . . 113
params.lognormal . . . 113
params.mixture . . . 114
params.mvn . . . 114
params.normal . . . 115
params.poisson_dist . . . 115
params.uniform_dist . . . 116
params.weibull_dist . . . 117
poisson_dist . . . 117
print.beta_dist . . . 118
print.chi_squared . . . 118
print.edist . . . 119
print.empirical_dist . . . 119
print.exponential . . . 120
print.gamma_dist . . . 120
print.interval . . . 121
print.lognormal . . . 122
print.mixture . . . 122
print.mvn . . . 123
print.normal . . . 123
print.poisson_dist . . . 124
print.realized_dist . . . 124
print.summary_dist . . . 125
print.uniform_dist . . . 126
print.weibull_dist . . . 126
realize . . . 127

rmap . . . 128
rmap.dist . . . 129
rmap.edist . . . 129
rmap.empirical_dist . . . 130
rmap.mvn . . . 131
sampler . . . 131
sampler.beta_dist . . . 132
sampler.chi_squared . . . 133
sampler.default . . . 133
sampler.edist . . . 134
sampler.empirical_dist . . . 135
sampler.exponential . . . 135
sampler.gamma_dist . . . 136
sampler.lognormal . . . 137
sampler.mixture . . . 137
sampler.mvn . . . 138
sampler.normal . . . 138
sampler.poisson_dist . . . 139
sampler.uniform_dist . . . 140
sampler.weibull_dist . . . 140
sample_mvn_region . . . 141
simplify . . . 142
simplify.dist . . . 142
simplify.edist . . . 143
Summary.dist . . . 144
summary.dist . . . 145
summary_dist . . . 146
sup . . . 146
sup.beta_dist . . . 147
sup.chi_squared . . . 148
sup.edist . . . 148
sup.empirical_dist . . . 149
sup.exponential . . . 149
sup.gamma_dist . . . 150
sup.lognormal . . . 151
sup.mixture . . . 151
sup.mvn . . . 152
sup.normal . . . 152
sup.poisson_dist . . . 153
sup.uniform_dist . . . 153
sup.weibull_dist . . . 154
supremum . . . 154
supremum.countable_set . . . 155
supremum.finite_set . . . 156
supremum.interval . . . 156
surv.chi_squared . . . 157
surv.continuous_dist . . . 157
surv.exponential . . . 158

surv.gamma_dist . . . 159
surv.lognormal . . . 159
surv.weibull_dist . . . 160
uniform_dist . . . 161
vcov.beta_dist . . . 161
vcov.chi_squared . . . 162
vcov.default . . . 162
vcov.edist . . . 163
vcov.empirical_dist . . . 164
vcov.exponential . . . 164
vcov.gamma_dist . . . 165
vcov.lognormal . . . 165
vcov.mixture . . . 166
vcov.mvn . . . 167
vcov.normal . . . 167
vcov.poisson_dist . . . 168
vcov.uniform_dist . . . 168
vcov.univariate_dist . . . 169
vcov.weibull_dist . . . 170
weibull_dist . . . 170
^.dist . . . 171

**Index** **172**

---

`*.dist` *Multiplication of distribution objects.*

---

## Description

Handles scalar * dist, dist * scalar, and dist * dist.

## Usage

```
## S3 method for class 'dist'
x * y
```

## Arguments

| | |
|---|---|
| `x` | first operand |
| `y` | second operand |

## Value

A simplified distribution or edist

## Examples

```
# Scalar multiplication simplifies for normal
z <- 2 * normal(0, 1)
z  # Normal(mu = 0, var = 4)

# Product of two distributions yields an edist
w <- normal(0, 1) * exponential(1)
is_edist(w)  # TRUE
```

---

`+.dist` *Method for adding* `dist` *objects, or shifting a distribution by a scalar.*

---

## Description

Creates an expression distribution and automatically simplifies to closed form when possible (e.g., normal + normal = normal, normal + scalar = normal with shifted mean).

## Usage

```
## S3 method for class 'dist'
x + y
```

## Arguments

| | |
|---|---|
| `x` | A `dist` object or numeric scalar |
| `y` | A `dist` object or numeric scalar |

## Value

A simplified distribution or `edist` if no closed form exists

## Examples

```
# Sum of two normals simplifies to a normal
z <- normal(0, 1) + normal(2, 3)
z  # Normal(mu = 2, var = 4)

# Shift a distribution by a constant
normal(0, 1) + 5  # Normal(mu = 5, var = 1)
```

## -.dist *Method for negation or subtraction of* dist *objects.*

### Description

Unary: returns negated distribution (e.g., -N(mu, var) = N(-mu, var)) Binary: creates expression distribution and simplifies to closed form when possible (e.g., normal - normal = normal, normal - scalar = normal).

### Usage

```
## S3 method for class 'dist'
x - y
```

### Arguments

| | |
|---|---|
| x | A dist object or numeric scalar |
| y | A dist object or numeric scalar (optional for unary negation) |

### Value

A simplified distribution or edist if no closed form exists

### Examples

```
# Difference of normals simplifies to a normal
z <- normal(5, 2) - normal(1, 3)
z  # Normal(mu = 4, var = 5)

# Unary negation
-normal(3, 1)  # Normal(mu = -3, var = 1)
```

## /.dist *Division of distribution objects.*

### Description

Handles dist / scalar (delegates to dist * (1/scalar)), scalar / dist, and dist / dist.

### Usage

```
## S3 method for class 'dist'
x / y
```

### Arguments

| | |
|---|---|
| x | first operand |
| y | second operand |

### Value

A simplified distribution or edist

### Examples

```
# Division by scalar reuses multiplication rule
z <- normal(0, 4) / 2
z  # Normal(mu = 0, var = 1)
```

---

## affine_transform *Affine transformation of a normal or multivariate normal distribution.*

---

### Description

Computes the distribution of $AX + b$ where $X \sim MVN(\mu, \Sigma)$. The result is $MVN(A\mu + b, A\Sigma A^T)$.

### Usage

```
affine_transform(x, A, b = NULL)
```

### Arguments

| | |
|---|---|
| x | A normal or mvn distribution object. |
| A | A numeric matrix (or scalar for univariate). |
| b | An optional numeric vector (or scalar) for the offset. Default is a zero vector. |

### Details

For a univariate normal, scalars A and b are promoted to 1x1 matrices and scalar internally. Returns a normal if the result is 1-dimensional.

### Value

A normal or mvn distribution.

## Examples

```
X <- mvn(c(0, 0), diag(2))
# Project to first component via 1x2 matrix
Y <- affine_transform(X, A = matrix(c(1, 0), 1, 2), b = 5)
mean(Y)

# Scale a univariate normal
Z <- affine_transform(normal(0, 1), A = 3, b = 2)
mean(Z)
vcov(Z)
```

---

# as_dist *Convert an object to a probability distribution.*

---

## Description

Generic method for converting objects (such as fitted models) into distribution objects from the `algebraic.dist` package.

## Usage

```
as_dist(x, ...)

## S3 method for class 'dist'
as_dist(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | The object to convert to a distribution. |
| `...` | Additional arguments to pass to methods. |

## Value

A `dist` object.

## Examples

```
# Identity for existing distributions
d <- normal(0, 1)
identical(as_dist(d), d)
```

## beta_dist *Construct a beta distribution object.*

### Description

Creates an S3 object representing a beta distribution with shape parameters `shape1` and `shape2`. The PDF on $(0, 1)$ is

$$f(x) = \frac{x^{a-1}(1-x)^{b-1}}{B(a,b)}$$

where $a$ = `shape1`, $b$ = `shape2`, and $B(a, b)$ is the beta function.

### Usage

```
beta_dist(shape1, shape2)
```

### Arguments

| | |
|---|---|
| `shape1` | First shape parameter, must be a positive scalar. |
| `shape2` | Second shape parameter, must be a positive scalar. |

### Value

A `beta_dist` object with classes `c("beta_dist", "univariate_dist", "continuous_dist", "dist")`.

### Examples

```
x <- beta_dist(shape1 = 2, shape2 = 5)
mean(x)
vcov(x)
format(x)
```

## cdf *Generic method for obtaining the cdf of an object.*

### Description

Generic method for obtaining the cdf of an object.

### Usage

```
cdf(x, ...)
```

### Arguments

| | |
|---|---|
| x | The object to obtain the cdf of. |
| ... | Additional arguments to pass. |

### Value

A function computing the cumulative distribution function.

### Examples

```
x <- normal(0, 1)
F <- cdf(x)
F(0)    # 0.5 (median of standard normal)
F(1.96) # approximately 0.975
```

---

## cdf.beta_dist *Cumulative distribution function for a beta distribution.*

---

### Description

Returns a function that evaluates the beta CDF at given points.

### Usage

```
## S3 method for class 'beta_dist'
cdf(x, ...)
```

### Arguments

| | |
|---|---|
| x | A beta_dist object. |
| ... | Additional arguments (not used). |

### Value

A function function(q, log.p = FALSE, ...) returning the CDF (or log-CDF) at q.

### Examples

```
x <- beta_dist(2, 5)
F <- cdf(x)
F(0.3)
F(0.5)
```

## cdf.chi_squared *Method for obtaining the cdf of a* chi_squared *object.*

### Description

Method for obtaining the cdf of a chi_squared object.

### Usage

```
## S3 method for class 'chi_squared'
cdf(x, ...)
```

### Arguments

| | |
|---|---|
| x | The chi_squared object |
| ... | Additional arguments (not used) |

### Value

A function that computes the cdf at point(s) t

### Examples

```
x <- chi_squared(5)
F <- cdf(x)
F(5)
F(10)
```

## cdf.edist *CDF for expression distributions.*

### Description

Falls back to realize to materialize the distribution as an empirical_dist, then delegates to cdf.empirical_dist.

### Usage

```
## S3 method for class 'edist'
cdf(x, ...)
```

### Arguments

| | |
|---|---|
| x | An edist object. |
| ... | Additional arguments forwarded to cdf.empirical_dist. |

## Value

A function computing the empirical CDF.

## Examples

```
set.seed(1)
z <- normal(0, 1) * exponential(1)
Fz <- cdf(z)
Fz(0)
```

---

`cdf.empirical_dist` *Method for obtaining the cdf of* `empirical_dist` *object* `x`.

---

## Description

If `x` is a multivariate empirical distribution, this function will throw an error. It's only defined for univariate empirical distributions.

## Usage

```
## S3 method for class 'empirical_dist'
cdf(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | The empirical distribution object. |
| `...` | Additional arguments to pass (not used)) |

## Value

A function that takes a numeric vector `t` and returns the empirical cdf of `x` evaluated at `t`.

## Examples

```
ed <- empirical_dist(c(1, 2, 3, 4, 5))
Fx <- cdf(ed)
Fx(3) # 0.6
Fx(c(1, 5)) # 0.2, 1.0
```

## cdf.exponential *Method to obtain the cdf of an* exponential *object.*

### Description

Method to obtain the cdf of an exponential object.

### Usage

```
## S3 method for class 'exponential'
cdf(x, ...)
```

### Arguments

| | |
|---|---|
| x | The object to obtain the cdf of |
| ... | Additional arguments (not used) |

### Value

A function function(q, lower.tail = TRUE, log.p = FALSE, ...) that computes the cdf (or log-cdf) of the exponential distribution.

### Examples

```
x <- exponential(rate = 1)
F <- cdf(x)
F(1)
F(2)
```

## cdf.gamma_dist *Method for obtaining the cdf of a* gamma_dist *object.*

### Description

Method for obtaining the cdf of a gamma_dist object.

### Usage

```
## S3 method for class 'gamma_dist'
cdf(x, ...)
```

### Arguments

| | |
|---|---|
| x | The gamma_dist object |
| ... | Additional arguments (not used) |

## Value

A function that computes the cdf at point(s) `t`

## Examples

```
x <- gamma_dist(shape = 2, rate = 1)
F <- cdf(x)
F(1)
F(2)
```

---

`cdf.lognormal` *Cumulative distribution function for a log-normal distribution.*

---

## Description

Returns a function that evaluates the log-normal CDF at given points.

## Usage

```
## S3 method for class 'lognormal'
cdf(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A `lognormal` object. |
| `...` | Additional arguments (not used). |

## Value

A function `function(q, log.p = FALSE, ...)` returning the CDF (or log-CDF) at `q`.

## Examples

```
x <- lognormal(0, 1)
F <- cdf(x)
F(1)
F(2)
```

## cdf.mixture *Cumulative distribution function for a mixture distribution.*

### Description

Returns a function that evaluates the mixture CDF at given points. The mixture CDF is $F(x) = \sum_k w_k F_k(x)$.

### Usage

```
## S3 method for class 'mixture'
cdf(x, ...)
```

### Arguments

| | |
|---|---|
| x | A mixture object. |
| ... | Additional arguments (not used). |

### Value

A function function(q, ...) returning the CDF at q.

### Examples

```
m <- mixture(list(normal(0, 1), normal(5, 1)), c(0.5, 0.5))
F <- cdf(m)
F(0)
F(5)
```

## cdf.mvn *Method for obtaining the CDF of a* mvn *object.*

### Description

Method for obtaining the CDF of a mvn object.

### Usage

```
## S3 method for class 'mvn'
cdf(x, ...)
```

### Arguments

| | |
|---|---|
| x | The object to obtain the CDF of |
| ... | Additional arguments to pass (not used) |

## Value

A function computing the multivariate normal CDF.

## Examples

```
X <- mvn(c(0, 0), diag(2))
F <- cdf(X)
F(c(0, 0))
```

---

`cdf.normal` *Method for obtaining the cdf of an* `normal` *object.*

---

## Description

Method for obtaining the cdf of an `normal` object.

## Usage

```
## S3 method for class 'normal'
cdf(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | The object to obtain the cdf of |
| `...` | Additional arguments to pass (not used) |

## Value

A function `function(q, lower.tail = TRUE, log.p = FALSE, ...)` that computes the cdf (or log-cdf) of the normal distribution at `q`.

## Examples

```
x <- normal(0, 1)
F <- cdf(x)
F(0)
F(1.96)
```

## cdf.poisson_dist *Cumulative distribution function for a Poisson distribution.*

### Description

Returns a function that evaluates the Poisson CDF at given points.

### Usage

```
## S3 method for class 'poisson_dist'
cdf(x, ...)
```

### Arguments

| | |
|---|---|
| x | A poisson_dist object. |
| ... | Additional arguments (not used). |

### Value

A function function(q, log.p = FALSE, ...) returning the CDF (or log-CDF) at q.

### Examples

```
x <- poisson_dist(5)
F <- cdf(x)
F(5)
F(10)
```

## cdf.uniform_dist *Cumulative distribution function for a uniform distribution.*

### Description

Returns a function that evaluates the uniform CDF at given points.

### Usage

```
## S3 method for class 'uniform_dist'
cdf(x, ...)
```

### Arguments

| | |
|---|---|
| x | A uniform_dist object. |
| ... | Additional arguments (not used). |

### Value

A function `function(q, log.p = FALSE, ...)` returning the CDF (or log-CDF) at q.

### Examples

```
x <- uniform_dist(0, 10)
F <- cdf(x)
F(5)
F(10)
```

---

## cdf.weibull_dist *Cumulative distribution function for a Weibull distribution.*

---

### Description

Returns a function that evaluates the Weibull CDF at given points.

### Usage

```
## S3 method for class 'weibull_dist'
cdf(x, ...)
```

### Arguments

| | |
|---|---|
| x | A `weibull_dist` object. |
| ... | Additional arguments (not used). |

### Value

A function `function(q, log.p = FALSE, ...)` returning the CDF (or log-CDF) at q.

### Examples

```
x <- weibull_dist(shape = 2, scale = 3)
F <- cdf(x)
F(1)
F(3)
```

## chi_squared *Construct a chi-squared distribution object.*

### Description

Construct a chi-squared distribution object.

### Usage

```
chi_squared(df)
```

### Arguments

| | |
|---|---|
| df | Degrees of freedom (positive scalar) |

### Value

A chi_squared object

### Examples

```
x <- chi_squared(df = 5)
mean(x)
vcov(x)
format(x)
```

## clt *Central Limit Theorem Limiting Distribution*

### Description

Returns the limiting distribution of the standardized sample mean $\sqrt{n}(\bar{X}_n - \mu)$ under the Central Limit Theorem. For a univariate distribution with variance $\sigma^2$, this is $N(0, \sigma^2)$. For a multivariate distribution with covariance matrix $\Sigma$, this is $MVN(0, \Sigma)$.

### Usage

```
clt(base_dist)
```

### Arguments

| | |
|---|---|
| base_dist | A dist object representing the base distribution. |

### Value

A normal or mvn distribution representing the CLT limiting distribution.

## Examples

```
# CLT for Exp(2): sqrt(n)(Xbar - 1/2) -> N(0, 1/4)
x <- exponential(rate = 2)
z <- clt(x)
mean(z)
vcov(z)
```

---

`conditional` *Generic method for obtaining the conditional distribution of a distribution object* `x` *given condition* `P`.

---

## Description

Generic method for obtaining the conditional distribution of a distribution object `x` given condition `P`.

## Usage

```
conditional(x, P, ...)
```

## Arguments

| | |
|---|---|
| `x` | The empirical distribution object. |
| `P` | The predicate function to condition `x` on |
| `...` | additional arguments to pass into `P` |

## Value

A distribution object for the conditional distribution.

## Examples

```
d <- empirical_dist(1:100)
# condition on values greater than 50
d_gt50 <- conditional(d, function(x) x > 50)
mean(d_gt50)
```

## conditional.dist *Method for obtaining the condition distribution,* `x | P(x)`*, of* `dist` *object* `x`.

### Description

Falls back to MC: materializes `x` via `ensure_realized()` and then conditions on the resulting empirical distribution.

### Usage

```
## S3 method for class 'dist'
conditional(x, P, n = 10000L, ...)
```

### Arguments

| | |
|---|---|
| `x` | The distribution object. |
| `P` | The predicate function to condition the distribution on |
| `n` | The number of samples to generate for the MC estimate of the conditional distribution x \| P. Defaults to 10000. |
| `...` | additional arguments to pass into `P`. |

### Value

An `empirical_dist` approximating the conditional distribution.

### Examples

```
set.seed(1)
x <- exponential(1)
# Condition on X > 2
x_gt2 <- conditional(x, function(t) t > 2)
mean(x_gt2)
```

## conditional.edist *Conditional distribution for expression distributions.*

### Description

Falls back to `realize` and delegates to `conditional.empirical_dist`.

### Usage

```
## S3 method for class 'edist'
conditional(x, P, ...)
```

### Arguments

| | |
|---|---|
| x | An edist object. |
| P | Predicate function to condition on. |
| ... | Additional arguments forwarded to the predicate P. |

### Value

A conditional empirical_dist.

### Examples

```
set.seed(1)
z <- normal(0, 1) + exponential(1)
z_pos <- conditional(z, function(t) t > 2)
mean(z_pos)
```

---

## conditional.empirical_dist

*Method for obtaining the condition distribution,* x | P(x)*, of* empirical_dist *object* x.

---

### Description

In other words, we condition the data on the predicate function. In order to do so, we simply remove all rows from the data that do not satisfy the predicate P. For instance, if we have a 2-dimensional distribution, and we want to condition on the first dimension being greater than the second dimension, we would do the following:

### Usage

```
## S3 method for class 'empirical_dist'
conditional(x, P, ...)
```

### Arguments

| | |
|---|---|
| x | The empirical distribution object. |
| P | The predicate function to condition the data on. |
| ... | additional arguments to pass into P. |

### Details

```
x_cond <- conditional(x, function(d) d[1] > d[2])
```

This would return a new empirical distribution object with the same dimensionality as x, but with all rows where the first dimension is less than or equal to the second dimension removed.

### Value

An `empirical_dist` containing only rows satisfying `P`.

### Examples

```
mat <- matrix(c(1, 5, 2, 3, 4, 1, 6, 2), ncol = 2)
ed <- empirical_dist(mat)
# Condition on first column being greater than second
ed_cond <- conditional(ed, function(d) d[1] > d[2])
nobs(ed_cond)
```

---

## `conditional.mixture` *Conditional distribution of a mixture.*

---

### Description

For a mixture of distributions that support closed-form conditioning (e.g. MVN), uses Bayes' rule to update the mixing weights:

$$w_k' \propto w_k f_k(x_{given})$$

where $f_k$ is the marginal density of component $k$ at the observed values. The component conditionals are computed via `conditional(component_k, given_indices = ..., given_values = ...)`.

### Usage

```
## S3 method for class 'mixture'
conditional(x, P = NULL, ..., given_indices = NULL, given_values = NULL)
```

### Arguments

| | |
|---|---|
| `x` | A `mixture` object. |
| `P` | Optional predicate function for MC fallback. |
| `...` | Additional arguments. |
| `given_indices` | Integer vector of observed variable indices. |
| `given_values` | Numeric vector of observed values. |

### Details

Falls back to MC realization if `P` is provided or if any component does not support `given_indices`/`given_values`.

### Value

A `mixture` or `empirical_dist` object.

### Examples

```
# Closed-form conditioning on MVN mixture
m <- mixture(
  list(mvn(c(0, 0), diag(2)), mvn(c(3, 3), diag(2))),
  c(0.5, 0.5)
)
# Condition on X2 = 1
mc <- conditional(m, given_indices = 2, given_values = 1)
mean(mc)
```

---

`conditional.mvn` *Conditional distribution for multivariate normal.*

---

### Description

Supports two calling patterns:

1. **Closed-form** (via `given_indices` and `given_values`): Uses the exact Schur complement formula. Returns a `normal` (1D result) or `mvn`.
2. **Predicate-based** (via `P`): Falls back to MC realization via `ensure_realized`.

### Usage

```
## S3 method for class 'mvn'
conditional(x, P = NULL, ..., given_indices = NULL, given_values = NULL)
```

### Arguments

| | |
|---|---|
| `x` | An `mvn` object. |
| `P` | Optional predicate function for MC fallback. |
| `...` | Additional arguments forwarded to the predicate `P`. |
| `given_indices` | Integer vector of observed variable indices. |
| `given_values` | Numeric vector of observed values (same length as `given_indices`). |

### Value

A `normal`, `mvn`, or `empirical_dist` object.

### Examples

```
# Closed-form conditioning: X2 | X1 = 1
sigma <- matrix(c(1, 0.5, 0.5, 1), 2, 2)
X <- mvn(c(0, 0), sigma)
X2_given <- conditional(X, given_indices = 1, given_values = 1)
mean(X2_given)
vcov(X2_given)
```

```
# Predicate-based MC fallback (slower)

set.seed(42)
X2_mc <- conditional(X, P = function(x) x[1] > 0)
```

---

`countable_set` *Countable Set*

---

## Description

A countably infinite support set, such as the non-negative integers. It satisfies the concept of a support (see `has`, `infimum`, `supremum`, `dim`).

## Public fields

`lower_bound` Integer lower bound of the set.

## Methods

### Public methods:

- `countable_set$new()`
- `countable_set$clone()`

**Method** `new()`**:** Initialize a countable set.

*Usage:*

```
countable_set$new(lower = 0L)
```

*Arguments:*

`lower` Integer lower bound (default 0).

**Method** `clone()`**:** The objects of this class are cloneable with this method.

*Usage:*

```
countable_set$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## delta_clt *Delta Method CLT Limiting Distribution*

### Description

Returns the limiting distribution of $\sqrt{n}(g(\bar{X}_n) - g(\mu))$ under the Delta Method. For a univariate distribution, this is $N(0, g'(\mu)^2\sigma^2)$. For a multivariate distribution with Jacobian $J = Dg(\mu)$, this is $MVN(0, J\Sigma J^T)$.

### Usage

```
delta_clt(base_dist, g, dg)
```

### Arguments

| | |
|---|---|
| base_dist | A dist object representing the base distribution. |
| g | The function to apply to the sample mean. |
| dg | The derivative (univariate) or Jacobian function (multivariate) of g. For univariate distributions, dg(x) should return a scalar. For multivariate distributions, dg(x) should return a matrix (the Jacobian). |

### Value

A normal or mvn distribution representing the Delta Method limiting distribution.

### Examples

```
# Delta method: g = exp, dg = exp
x <- exponential(rate = 1)
z <- delta_clt(x, g = exp, dg = exp)
mean(z)
vcov(z)
```

## density.beta_dist *Probability density function for a beta distribution.*

### Description

Returns a function that evaluates the beta PDF at given points.

### Usage

```
## S3 method for class 'beta_dist'
density(x, ...)
```

### Arguments

| | |
|---|---|
| x | A beta_dist object. |
| ... | Additional arguments (not used). |

### Value

A function function(t, log = FALSE, ...) returning the density (or log-density) at t.

### Examples

```
x <- beta_dist(2, 5)
f <- density(x)
f(0.3)
f(0.5)
```

---

## density.chi_squared *Method for obtaining the density (pdf) of a* chi_squared *object.*

---

### Description

Method for obtaining the density (pdf) of a chi_squared object.

### Usage

```
## S3 method for class 'chi_squared'
density(x, ...)
```

### Arguments

| | |
|---|---|
| x | The chi_squared object |
| ... | Additional arguments (not used) |

### Value

A function that computes the pdf at point(s) t

### Examples

```
x <- chi_squared(5)
f <- density(x)
f(5)
f(10)
```

## density.edist *Density for expression distributions.*

### Description

Falls back to `realize` and delegates to `density.empirical_dist`.

### Usage

```
## S3 method for class 'edist'
density(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | An `edist` object. |
| `...` | Additional arguments forwarded to `density.empirical_dist`. |

### Value

A function computing the empirical density (PMF).

### Examples

```
set.seed(1)
z <- normal(0, 1) * exponential(1)
fz <- density(z)
```

## density.empirical_dist *Method for obtaining the pdf of a* `empirical_dist` *object.*

### Description

Method for obtaining the pdf of a `empirical_dist` object.

### Usage

```
## S3 method for class 'empirical_dist'
density(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | The object to obtain the pdf of. |
| `...` | Additional arguments to pass into the pdf function. |

## Value

A function computing the empirical PMF at given points.

## Note

sort tibble lexicographically and do a binary search to find upper and lower bound in `log(nobs(x))` time.

## Examples

```
ed <- empirical_dist(c(1, 2, 2, 3, 3, 3))
f <- density(ed)
f(2)          # 2/6
f(3, log = TRUE) # log(3/6)
```

---

`density.exponential` *Method to obtain the pdf of an* `exponential` *object.*

---

## Description

Method to obtain the pdf of an `exponential` object.

## Usage

```
## S3 method for class 'exponential'
density(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | The object to obtain the pdf of |
| `...` | Additional arguments (not used) |

## Value

A function `function(t, log = FALSE, ...)` that computes the pdf (or log-pdf) of the exponential distribution at `t`.

## Examples

```
x <- exponential(rate = 2)
f <- density(x)
f(0)
f(1)
```

## density.gamma_dist *Method for obtaining the density (pdf) of a* gamma_dist *object.*

### Description

Method for obtaining the density (pdf) of a gamma_dist object.

### Usage

```
## S3 method for class 'gamma_dist'
density(x, ...)
```

### Arguments

| | |
|---|---|
| x | The gamma_dist object |
| ... | Additional arguments (not used) |

### Value

A function that computes the pdf at point(s) t

### Examples

```
x <- gamma_dist(shape = 2, rate = 1)
f <- density(x)
f(1)
f(2)
```

## density.lognormal *Probability density function for a log-normal distribution.*

### Description

Returns a function that evaluates the log-normal PDF at given points.

### Usage

```
## S3 method for class 'lognormal'
density(x, ...)
```

### Arguments

| | |
|---|---|
| x | A lognormal object. |
| ... | Additional arguments (not used). |

## Value

A function `function(t, log = FALSE, ...)` returning the density (or log-density) at `t`.

## Examples

```
x <- lognormal(0, 1)
f <- density(x)
f(1)
f(2)
```

---

`density.mixture` *Probability density function for a mixture distribution.*

---

## Description

Returns a function that evaluates the mixture density at given points. The mixture density is $f(x) = \sum_k w_k f_k(x)$.

## Usage

```
## S3 method for class 'mixture'
density(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A `mixture` object. |
| `...` | Additional arguments (not used). |

## Value

A function `function(t, log = FALSE, ...)` returning the density (or log-density) at `t`.

## Examples

```
m <- mixture(list(normal(0, 1), normal(5, 1)), c(0.5, 0.5))
f <- density(m)
f(0)
f(2.5)
```

`density.mvn` *Function generator for obtaining the pdf of an* `mvn` *object (multivariate normal).*

## Description

Function generator for obtaining the pdf of an `mvn` object (multivariate normal).

## Usage

```
## S3 method for class 'mvn'
density(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | The `mvn` (S3) object to obtain the pdf (density) of |
| `...` | Additional arguments passed to `dmvnorm` on every call. |

## Value

A function `function(obs, log = FALSE, ...)` that computes the pdf (or log-pdf) of the multivariate normal distribution.

## Examples

```
X <- mvn(c(0, 0), diag(2))
f <- density(X)
f(c(0, 0))
f(c(1, 1))
```

`density.normal` *Method for obtaining the pdf of an* `normal` *object.*

## Description

Method for obtaining the pdf of an `normal` object.

## Usage

```
## S3 method for class 'normal'
density(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | The object to obtain the pdf of |
| `...` | Additional arguments to pass (not used) |

## Value

A function `function(t, log = FALSE, ...)` that computes the pdf (or log-pdf) of the normal distribution at `t`.

## Examples

```
x <- normal(0, 1)
f <- density(x)
f(0)
f(1)
```

---

`density.poisson_dist` *Probability mass function for a Poisson distribution.*

---

## Description

Returns a function that evaluates the Poisson PMF at given points.

## Usage

```
## S3 method for class 'poisson_dist'
density(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A `poisson_dist` object. |
| `...` | Additional arguments (not used). |

## Value

A function `function(k, log = FALSE, ...)` returning the probability mass (or log-probability) at `k`.

## Examples

```
x <- poisson_dist(5)
f <- density(x)
f(5)
f(0)
```

## density.uniform_dist *Probability density function for a uniform distribution.*

### Description

Returns a function that evaluates the uniform PDF at given points.

### Usage

```
## S3 method for class 'uniform_dist'
density(x, ...)
```

### Arguments

| | |
|---|---|
| x | A uniform_dist object. |
| ... | Additional arguments (not used). |

### Value

A function function(t, log = FALSE, ...) returning the density (or log-density) at t.

### Examples

```
x <- uniform_dist(0, 10)
f <- density(x)
f(5)
f(15)
```

## density.weibull_dist *Probability density function for a Weibull distribution.*

### Description

Returns a function that evaluates the Weibull PDF at given points.

### Usage

```
## S3 method for class 'weibull_dist'
density(x, ...)
```

### Arguments

| | |
|---|---|
| x | A weibull_dist object. |
| ... | Additional arguments (not used). |

## Value

A function `function(t, log = FALSE, ...)` returning the density (or log-density) at `t`.

## Examples

```
x <- weibull_dist(shape = 2, scale = 3)
f <- density(x)
f(1)
f(3)
```

---

`dim.beta_dist` *Dimension of a beta distribution (always 1).*

---

## Description

Dimension of a beta distribution (always 1).

## Usage

```
## S3 method for class 'beta_dist'
dim(x)
```

## Arguments

| | |
|---|---|
| `x` | A `beta_dist` object. |

## Value

1.

## Examples

```
dim(beta_dist(2, 5))
```

---

`dim.chi_squared` *Retrieve the dimension of a* `chi_squared` *object.*

---

## Description

Retrieve the dimension of a `chi_squared` object.

## Usage

```
## S3 method for class 'chi_squared'
dim(x)
```

### Arguments

| | |
|---|---|
| x | The chi_squared object |

### Value

1 (univariate)

### Examples

```
dim(chi_squared(5))
```

---

## dim.countable_set *Get the dimension of a countable set.*

---

### Description

Get the dimension of a countable set.

### Usage

```
## S3 method for class 'countable_set'
dim(x)
```

### Arguments

| | |
|---|---|
| x | A countable_set object. |

### Value

1 (always univariate).

### Examples

```
cs <- countable_set$new(0L)
dim(cs)  # 1
```

## dim.edist *Method for obtaining the dimension of an* edist *object.*

### Description

Determines the dimension by drawing a single sample and checking whether it is a matrix (multivariate) or scalar (univariate).

### Usage

```
## S3 method for class 'edist'
dim(x)
```

### Arguments

| | |
|---|---|
| x | The edist object. |

### Value

Integer; the number of dimensions.

### Examples

```
z <- normal(0, 1) * exponential(1)
dim(z)
```

## dim.empirical_dist *Method for obtaining the dimension of a* empirical_dist *object.*

### Description

Method for obtaining the dimension of a empirical_dist object.

### Usage

```
## S3 method for class 'empirical_dist'
dim(x)
```

### Arguments

| | |
|---|---|
| x | The object to obtain the dimension of. |

### Value

Integer; the number of dimensions.

## Examples

```
ed1 <- empirical_dist(c(1, 2, 3))
dim(ed1) # 1

ed2 <- empirical_dist(matrix(1:6, ncol = 2))
dim(ed2) # 2
```

---

`dim.exponential` *Method to obtain the dimension of an* `exponential` *object.*

---

## Description

Method to obtain the dimension of an `exponential` object.

## Usage

```
## S3 method for class 'exponential'
dim(x)
```

## Arguments

| | |
|---|---|
| `x` | The `exponential` object to obtain the dimension of |

## Value

The dimension of the `exponential` object

## Examples

```
dim(exponential(rate = 1))
```

---

`dim.finite_set` *Return the dimension of the finite set.*

---

## Description

Return the dimension of the finite set.

## Usage

```
## S3 method for class 'finite_set'
dim(x)
```

## Arguments

| | |
|---|---|
| `x` | A finite set. |

## Value

Integer; the dimension of the set.

## Examples

```
fs <- finite_set$new(c(1, 3, 5, 7))
dim(fs) # 1
```

---

`dim.gamma_dist` *Retrieve the dimension of a* `gamma_dist` *object.*

---

## Description

Retrieve the dimension of a `gamma_dist` object.

## Usage

```
## S3 method for class 'gamma_dist'
dim(x)
```

## Arguments

| | |
|---|---|
| `x` | The `gamma_dist` object |

## Value

1 (univariate)

## Examples

```
dim(gamma_dist(2, 1))
```

---

`dim.interval` *Return the dimension of the interval.*

---

## Description

Return the dimension of the interval.

## Usage

```
## S3 method for class 'interval'
dim(x)
```

## Arguments

| | |
|---|---|
| `x` | An interval object. |

### Value

Integer; the number of interval components.

### Examples

```
iv <- interval$new(lower = 0, upper = 1)
dim(iv) # 1
```

---

## dim.lognormal *Dimension of a log-normal distribution (always 1).*

---

### Description

Dimension of a log-normal distribution (always 1).

### Usage

```
## S3 method for class 'lognormal'
dim(x)
```

### Arguments

| | |
|---|---|
| x | A lognormal object. |

### Value

1.

### Examples

```
dim(lognormal(0, 1))
```

---

## dim.mixture *Dimension of a mixture distribution.*

---

### Description

Returns the dimension of the first component (all components are assumed to have the same dimension).

### Usage

```
## S3 method for class 'mixture'
dim(x)
```

## Arguments

| | |
|---|---|
| x | A mixture object. |

## Value

The dimension of the distribution.

## Examples

```
m <- mixture(list(normal(0, 1), normal(5, 1)), c(0.5, 0.5))
dim(m)
```

---

# dim.mvn *Method for obtaining the dimension of an* mvn *object.*

---

## Description

Method for obtaining the dimension of an mvn object.

## Usage

```
## S3 method for class 'mvn'
dim(x)
```

## Arguments

| | |
|---|---|
| x | The object to obtain the dimension of |

## Value

The dimension of the mvn object

## Examples

```
dim(mvn(c(0, 0, 0)))
```

## dim.normal *Method for obtaining the dimension of a* normal *object.*

### Description

Method for obtaining the dimension of a normal object.

### Usage

```
## S3 method for class 'normal'
dim(x)
```

### Arguments

| | |
|---|---|
| x | The normal object to obtain the dimension of |

### Value

The dimension of the normal object

### Examples

```
dim(normal(0, 1))
```

## dim.poisson_dist *Dimension of a Poisson distribution (always 1).*

### Description

Dimension of a Poisson distribution (always 1).

### Usage

```
## S3 method for class 'poisson_dist'
dim(x)
```

### Arguments

| | |
|---|---|
| x | A poisson_dist object. |

### Value

1.

### Examples

```
dim(poisson_dist(5))
```

## dim.uniform_dist *Dimension of a uniform distribution (always 1).*

### Description

Dimension of a uniform distribution (always 1).

### Usage

```
## S3 method for class 'uniform_dist'
dim(x)
```

### Arguments

| | |
|---|---|
| x | A uniform_dist object. |

### Value

1.

### Examples

```
dim(uniform_dist(0, 1))
```

## dim.weibull_dist *Dimension of a Weibull distribution (always 1).*

### Description

Dimension of a Weibull distribution (always 1).

### Usage

```
## S3 method for class 'weibull_dist'
dim(x)
```

### Arguments

| | |
|---|---|
| x | A weibull_dist object. |

### Value

1.

### Examples

```
dim(weibull_dist(2, 3))
```

edist *Takes an expression* e *and a list* vars *and returns a lazy* edist *(expression distribution object), that is a subclass of* dist *that can be used in place of a* dist *object.*

## Description

Takes an expression e and a list vars and returns a lazy edist (expression distribution object), that is a subclass of dist that can be used in place of a dist object.

## Usage

```
edist(e, vars)
```

## Arguments

| | |
|---|---|
| e | the expression to evaluate against the arguments. |
| vars | the list of distributions (with variable names) to evaluate the expression e against. |

## Value

An edist object.

## Examples

```
x <- normal(0, 1)
y <- normal(2, 3)
e <- edist(quote(x + y), list(x = x, y = y))
e
```

empirical_dist *Construct empirical distribution object.*

## Description

Construct empirical distribution object.

## Usage

```
empirical_dist(data)
```

## Arguments

| | |
|---|---|
| data | data to construct empirical distribution from. if matrix or data frame, each row is a joint observation, if a vector, each element is an observation. whatever data is, it must be convertible to a tibble. |

## Value

An `empirical_dist` object.

## Examples

```
# Univariate empirical distribution from a vector
ed <- empirical_dist(c(1, 2, 3, 4, 5))
mean(ed)

# Multivariate empirical distribution from a matrix
mat <- matrix(c(1, 2, 3, 4, 5, 6), ncol = 2)
ed_mv <- empirical_dist(mat)
dim(ed_mv)
```

---

# expectation *Generic method for obtaining the expectation of* `f` *with respect to* `x`.

---

## Description

Generic method for obtaining the expectation of `f` with respect to `x`.

## Usage

```
expectation(x, g, ...)
```

## Arguments

| | |
|---|---|
| `x` | The distribution object. |
| `g` | The function to take the expectation of. |
| `...` | Additional arguments to pass into g. |

## Value

The expected value of `g(x)`.

## Examples

```
x <- exponential(1)
# E[X] for Exp(1) is 1
expectation(x, function(t) t)
```

`expectation.dist` *Expectation of a Function Applied to a* `dist` *Object*

## Description

Expectation operator applied to `x` of type `dist` with respect to a function `g`. Optionally, constructs a confidence interval for the expectation estimate using the Central Limit Theorem.

## Usage

```
## S3 method for class 'dist'
expectation(x, g = function(t) t, ..., control = list())
```

## Arguments

| | |
|---|---|
| `x` | A `dist` object. |
| `g` | Characteristic function of interest, defaults to identity. |
| `...` | Additional arguments to pass to `g`. |
| `control` | A list of control parameters: compute_stats - Logical, whether to compute CIs for the expectations, defaults to FALSE n - Integer, the number of samples to use for the MC estimate, defaults to 10000L alpha - Real, the significance level for the confidence interval, defaults to 0.05 |

## Value

If `compute_stats` is FALSE, then the estimate of the expectation, otherwise a list with the following components: value - The estimate of the expectation ci - The confidence intervals for each component of the expectation n - The number of samples

## Examples

```
# MC expectation of X^2 where X ~ Exp(1)
set.seed(1)
ex <- exponential(1)
expectation(ex, g = function(t) t^2)
```

`expectation.empirical_dist` *Method for obtaining the expectation of* `empirical_dist` *object* `x` *under function* `g`.

## Description

Method for obtaining the expectation of `empirical_dist` object `x` under function `g`.

## Usage

```
## S3 method for class 'empirical_dist'
expectation(x, g = function(t) t, ..., control = list())
```

## Arguments

| | |
|---|---|
| `x` | The distribution object. |
| `g` | The function to take the expectation of. |
| `...` | Additional arguments to pass into function `g`. |
| `control` | a list of control parameters: compute_stats - Whether to compute CIs for the expectations, defaults to FALSE n - The number of samples to use for the MC estimate, defaults to 10000 alpha - The significance level for the confidence interval, defaults to 0.05 |

## Value

If `compute_stats` is FALSE, then the estimate of the expectation, otherwise a list with the following components: value - The estimate of the expectation ci - The confidence intervals for each component of the expectation n - The number of samples

## Examples

```
ed <- empirical_dist(c(1, 2, 3, 4, 5))
expectation(ed)                     # E[X] = 3
expectation(ed, function(x) x^2)    # E[X^2] = 11
```

`expectation.poisson_dist` *Exact expectation for a Poisson distribution.*

## Description

Computes $E[g(X)]$ using truncated summation over the support. The summation is truncated at the $1 - 10^{-12}$ quantile to ensure negligible truncation error.

### Usage

```
## S3 method for class 'poisson_dist'
expectation(x, g, ...)
```

### Arguments

| | |
|---|---|
| x | A poisson_dist object. |
| g | A function to take the expectation of. |
| ... | Additional arguments passed to g. |

### Value

The expected value $E[g(X)]$.

### Examples

```
x <- poisson_dist(5)
expectation(x, identity)
expectation(x, function(k) k^2)
```

---

## expectation.univariate_dist

*Method for obtaining the expectation of* f *with respect to a* univariate_dist *object* x.

---

### Description

Assumes the support is a contiguous interval that has operations for retrieving the lower and upper bounds.

### Usage

```
## S3 method for class 'univariate_dist'
expectation(x, g, ..., control = list())
```

### Arguments

| | |
|---|---|
| x | The distribution object. |
| g | The function to take the expectation of. |
| ... | Additional arguments to pass into g. |
| control | An (optional) list of control parameters for integrate or expectation_data (if x is not continuous) |

### Value

The expected value (numeric scalar), or the full integrate() result if compute_stats = TRUE.

## Examples

```
x <- normal(3, 4)
# E[X] for Normal(3, 4) is 3
expectation(x, function(t) t)

# E[X^2] for Exp(1) is 2
expectation(exponential(1), function(t) t^2)
```

---

`expectation_data` *Function used for computing expectations given data (e.g., from an MC simulation or bootstrap). it expects a matrix, or something that can be coerced to a matrix (e.g., a data frame). it also expects a function* `g` *to apply to each row of the data, and returns the expectation of* `g` *under the empirical distribution of the data. it also returns a confidence interval for the expectation, and the number of samples used to compute the expectation.*

---

## Description

example: expectation_data(D, function(x) (x-colMeans(D)) %*% t(x-colMeans(D))) computes the covariance of the data D, except the matrix structure is lost (it's just a vector, which can be coerced back to a matrix if needed).

## Usage

```
expectation_data(
  data,
  g = function(x) x,
  ...,
  compute_stats = TRUE,
  alpha = 0.05
)
```

## Arguments

| | |
|---|---|
| `data` | a matrix of data |
| `g` | a function to apply to each row of the data |
| `...` | additional arguments to pass to g |
| `compute_stats` | whether to compute CIs for the expectations |
| `alpha` | the confidence level for the confidence interval for each component of the expectation (if compute_stats is TRUE) |

## Value

if compute_stats is TRUE, then a list with the following components: value - The estimate of the expectation ci - The confidence intervals for each component of the expectation n - The number of samples otherwise, just the value of the expectation.

## Examples

```
set.seed(42)
data <- matrix(rnorm(200), ncol = 2)
# sample mean with confidence interval
expectation_data(data)

# just the point estimate, no CI
expectation_data(data, compute_stats = FALSE)

# expectation of a function of the data (row-wise)
expectation_data(data, g = function(x) sum(x^2))
```

---

`exponential` *Construct exponential distribution object.*

---

## Description

Construct exponential distribution object.

## Usage

```
exponential(rate)
```

## Arguments

| | |
|---|---|
| `rate` | failure rate |

## Value

An `exponential` distribution object.

## Examples

```
x <- exponential(rate = 2)
mean(x)
vcov(x)
format(x)
```

`finite_set` *Finite set*

## Description

A finite set. It also satisfies the concept of a support.

## Public fields

`values` A vector of values.

## Methods

### Public methods:

- `finite_set$new()`
- `finite_set$has()`
- `finite_set$infimum()`
- `finite_set$supremum()`
- `finite_set$dim()`
- `finite_set$clone()`

**Method** `new()`**:** Initialize a finite set.

*Usage:*

```
finite_set$new(values)
```

*Arguments:*

`values` A vector of values.

**Method** `has()`**:** Determine if a value is contained in the finite set.

*Usage:*

```
finite_set$has(x)
```

*Arguments:*

`x` A vector of values.

**Method** `infimum()`**:** Get the infimum of the finite set.

*Usage:*

```
finite_set$infimum()
```

*Returns:* A numeric vector of infimums.

**Method** `supremum()`**:** Get the supremum of the finite set.

*Usage:*

```
finite_set$supremum()
```

*Returns:* A numeric vector of supremums.

**Method** `dim()`**:** Get the dimension of the finite set.

*Usage:*

```
finite_set$dim()
```

*Returns:* The dimension of the finite set.

**Method** `clone()`**:** The objects of this class are cloneable with this method.

*Usage:*

```
finite_set$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

---

`format.beta_dist` *Format a* `beta_dist` *object as a character string.*

---

## Description

Format a `beta_dist` object as a character string.

## Usage

```
## S3 method for class 'beta_dist'
format(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A `beta_dist` object. |
| `...` | Additional arguments (not used). |

## Value

A character string describing the distribution.

## Examples

```
format(beta_dist(2, 5))
```

## format.chi_squared *Format a* chi_squared *object as a character string.*

### Description

Format a chi_squared object as a character string.

### Usage

```
## S3 method for class 'chi_squared'
format(x, ...)
```

### Arguments

| | |
|---|---|
| x | The chi_squared object |
| ... | Additional arguments (not used) |

### Value

A character string describing the distribution

### Examples

```
format(chi_squared(5))
```

## format.edist *Format method for* edist *objects.*

### Description

Format method for edist objects.

### Usage

```
## S3 method for class 'edist'
format(x, ...)
```

### Arguments

| | |
|---|---|
| x | The object to format |
| ... | Additional arguments (not used) |

### Value

A character string

### Examples

```
z <- normal(0, 1) * exponential(2)
format(z)
```

---

## format.empirical_dist *Format method for* empirical_dist *objects.*

---

### Description

Format method for empirical_dist objects.

### Usage

```
## S3 method for class 'empirical_dist'
format(x, ...)
```

### Arguments

| | |
|---|---|
| x | The object to format |
| ... | Additional arguments (not used) |

### Value

A character string

### Examples

```
ed <- empirical_dist(c(1, 2, 3, 4, 5))
format(ed)
```

---

## format.exponential *Format method for* exponential *objects.*

---

### Description

Format method for exponential objects.

### Usage

```
## S3 method for class 'exponential'
format(x, ...)
```

### Arguments

| | |
|---|---|
| x | The exponential object to format |
| ... | Additional arguments (not used) |

## Value

A character string

## Examples

```
format(exponential(rate = 2))
```

---

`format.gamma_dist` *Format a* `gamma_dist` *object as a character string.*

---

## Description

Format a `gamma_dist` object as a character string.

## Usage

```
## S3 method for class 'gamma_dist'
format(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | The `gamma_dist` object |
| `...` | Additional arguments (not used) |

## Value

A character string describing the distribution

## Examples

```
format(gamma_dist(2, 1))
```

---

`format.lognormal` *Format a* `lognormal` *object as a character string.*

---

## Description

Format a `lognormal` object as a character string.

## Usage

```
## S3 method for class 'lognormal'
format(x, ...)
```

### Arguments

| | |
|---|---|
| x | A lognormal object. |
| ... | Additional arguments (not used). |

### Value

A character string describing the distribution.

### Examples

```
format(lognormal(0, 1))
```

---

## format.mixture *Format a* mixture *object as a character string.*

---

### Description

Format a mixture object as a character string.

### Usage

```
## S3 method for class 'mixture'
format(x, ...)
```

### Arguments

| | |
|---|---|
| x | A mixture object. |
| ... | Additional arguments (not used). |

### Value

A character string describing the mixture.

### Examples

```
m <- mixture(list(normal(0, 1), normal(5, 1)), c(0.5, 0.5))
format(m)
```

## format.mvn *Format method for* mvn *objects.*

### Description

Format method for mvn objects.

### Usage

```
## S3 method for class 'mvn'
format(x, ...)
```

### Arguments

| | |
|---|---|
| x | The object to format |
| ... | Additional arguments (not used) |

### Value

A character string

### Examples

```
format(mvn(c(0, 0)))
```

## format.normal *Format method for* normal *objects.*

### Description

Format method for normal objects.

### Usage

```
## S3 method for class 'normal'
format(x, ...)
```

### Arguments

| | |
|---|---|
| x | The object to format |
| ... | Additional arguments (not used) |

### Value

A character string

### Examples

```
x <- normal(2, 3)
format(x)
```

---

## format.poisson_dist *Format a* poisson_dist *object as a character string.*

---

### Description

Format a poisson_dist object as a character string.

### Usage

```
## S3 method for class 'poisson_dist'
format(x, ...)
```

### Arguments

| | |
|---|---|
| x | A poisson_dist object. |
| ... | Additional arguments (not used). |

### Value

A character string describing the distribution.

### Examples

```
format(poisson_dist(5))
```

---

## format.realized_dist *Format a* realized_dist *object as a character string.*

---

### Description

Shows the number of samples and a summary of the source distribution.

### Usage

```
## S3 method for class 'realized_dist'
format(x, ...)
```

### Arguments

| | |
|---|---|
| x | A realized_dist object. |
| ... | Additional arguments (not used). |

## Value

A character string.

## Examples

```
rd <- realize(normal(0, 1), n = 100)
format(rd)
```

---

`format.uniform_dist` *Format a* `uniform_dist` *object as a character string.*

---

## Description

Format a `uniform_dist` object as a character string.

## Usage

```
## S3 method for class 'uniform_dist'
format(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A `uniform_dist` object. |
| `...` | Additional arguments (not used). |

## Value

A character string describing the distribution.

## Examples

```
format(uniform_dist(0, 10))
```

---

`format.weibull_dist` *Format a* `weibull_dist` *object as a character string.*

---

## Description

Format a `weibull_dist` object as a character string.

## Usage

```
## S3 method for class 'weibull_dist'
format(x, ...)
```

### Arguments

| | |
|---|---|
| x | A weibull_dist object. |
| ... | Additional arguments (not used). |

### Value

A character string describing the distribution.

### Examples

```
format(weibull_dist(2, 3))
```

---

## gamma_dist *Construct a gamma distribution object.*

---

### Description

Construct a gamma distribution object.

### Usage

```
gamma_dist(shape, rate)
```

### Arguments

| | |
|---|---|
| shape | Shape parameter (positive scalar) |
| rate | Rate parameter (positive scalar) |

### Value

A gamma_dist object

### Examples

```
x <- gamma_dist(shape = 2, rate = 1)
mean(x)
vcov(x)
format(x)
```

has *Support*

## Description

support is a class that represents the support of a random element or distribution, i.e. the set of values that it realize.

It's a conceptual class. To satisfy the concept of a support, the following methods must be implemented:

1. has: a function that returns a logical vector indicating whether each value in a vector is contained in the support
2. infimum: a function that returns the infimum of the support
3. supremum: a function that returns the supremum of the support
4. dim: a function that returns the dimension of the support

We provide two implementations that satisfy the concept:

- `interval`: a support that is an infiite set of contiguous numeric values
- `finite_set`: a support that is a finite set of values Determine if a value is contained in the support.

## Usage

```
has(object, x)
```

## Arguments

| | |
|---|---|
| `object` | A support object. |
| `x` | A vector of values. |

## Value

Logical vector indicating membership.

## Examples

```
I <- interval$new(0, 1, lower_closed = TRUE, upper_closed = TRUE)
has(I, 0.5)  # TRUE
has(I, 2)    # FALSE

S <- finite_set$new(c(1, 2, 3))
has(S, 2)    # TRUE
has(S, 4)    # FALSE
```

## has.countable_set *Check membership in a countable set.*

### Description

Returns TRUE if all values are integers (within floating-point tolerance) that are at least as large as the lower bound.

### Usage

```
## S3 method for class 'countable_set'
has(object, x)
```

### Arguments

| | |
|---|---|
| object | A countable_set object. |
| x | Value(s) to check. |

### Value

Logical; TRUE if all values are valid members of the set.

### Examples

```
cs <- countable_set$new(0L)
has(cs, c(0, 3, 5))   # TRUE
has(cs, c(-1, 2))     # FALSE (negative integer)
has(cs, 1.5)          # FALSE (not integer)
```

## has.finite_set *Determine if a value is contained in the finite set.*

### Description

Determine if a value is contained in the finite set.

### Usage

```
## S3 method for class 'finite_set'
has(object, x)
```

### Arguments

| | |
|---|---|
| object | A finite set. |
| x | A vector of values. |

## Value

Logical indicating membership.

## Examples

```
fs <- finite_set$new(c(1, 3, 5, 7))
has(fs, 3) # TRUE
has(fs, 4) # FALSE
```

---

`has.interval` *Determine if a value is contained in the interval.*

---

## Description

Determine if a value is contained in the interval.

## Usage

```
## S3 method for class 'interval'
has(object, x)
```

## Arguments

| | |
|---|---|
| `object` | An interval object. |
| `x` | A vector of values. |

## Value

Logical vector indicating containment.

## Examples

```
iv <- interval$new(lower = 0, upper = 1)
has(iv, 0.5) # TRUE
has(iv, 2.0) # FALSE
```

## hazard.chi_squared *Method for obtaining the hazard function of a* chi_squared *object.*

### Description

Method for obtaining the hazard function of a chi_squared object.

### Usage

```
## S3 method for class 'chi_squared'
hazard(x, ...)
```

### Arguments

| | |
|---|---|
| x | The chi_squared object |
| ... | Additional arguments (not used) |

### Value

A function that computes h(t) = f(t) / S(t)

### Examples

```
x <- chi_squared(5)
h <- hazard(x)
h(5)
```

## hazard.continuous_dist *Default hazard function for continuous distributions.*

### Description

Computes $h(t) = f(t)/S(t)$ from the density and survival function.

### Usage

```
## S3 method for class 'continuous_dist'
hazard(x, ...)

hazard(x, ...)
```

### Arguments

| | |
|---|---|
| x | The object to obtain the hazard function of. |
| ... | Additional arguments to pass. |

## Value

A function `function(t, ...)` returning the hazard rate.

A function computing the hazard rate at given points.

## Examples

```
x <- normal(0, 1)
h <- hazard(x)
h(0)
x <- exponential(2)
h <- hazard(x)
h(1)  # hazard rate at t = 1 (constant for exponential)
```

---

`hazard.exponential` *Method to obtain the hazard function of an* `exponential` *object.*

---

## Description

Method to obtain the hazard function of an `exponential` object.

## Usage

```
## S3 method for class 'exponential'
hazard(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | The `exponential` object to obtain the hazard function of |
| `...` | Additional arguments (not used) |

## Value

A function `function(t, log = FALSE, ...)` that computes the hazard rate (or log-hazard) of the exponential distribution.

## Examples

```
x <- exponential(rate = 2)
h <- hazard(x)
h(1)
h(5)
```

## hazard.gamma_dist *Method for obtaining the hazard function of a* gamma_dist *object.*

### Description

Method for obtaining the hazard function of a gamma_dist object.

### Usage

```
## S3 method for class 'gamma_dist'
hazard(x, ...)
```

### Arguments

| | |
|---|---|
| x | The gamma_dist object |
| ... | Additional arguments (not used) |

### Value

A function that computes h(t) = f(t) / S(t)

### Examples

```
x <- gamma_dist(shape = 2, rate = 1)
h <- hazard(x)
h(1)
```

## hazard.lognormal *Hazard function for a log-normal distribution.*

### Description

Returns a function that evaluates the log-normal hazard rate $h(t) = f(t)/S(t)$ for $t > 0$.

### Usage

```
## S3 method for class 'lognormal'
hazard(x, ...)
```

### Arguments

| | |
|---|---|
| x | A lognormal object. |
| ... | Additional arguments (not used). |

## Value

A function `function(t, log = FALSE)` returning the hazard (or log-hazard) at `t`.

## Examples

```
x <- lognormal(0, 1)
h <- hazard(x)
h(1)
h(2)
```

---

`hazard.weibull_dist` *Hazard function for a Weibull distribution.*

---

## Description

Returns a function that evaluates the Weibull hazard rate $h(t) = (shape/scale)(t/scale)^{shape-1}$ for $t > 0$.

## Usage

```
## S3 method for class 'weibull_dist'
hazard(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A `weibull_dist` object. |
| `...` | Additional arguments (not used). |

## Value

A function `function(t, log = FALSE)` returning the hazard (or log-hazard) at `t`.

## Examples

```
x <- weibull_dist(shape = 2, scale = 3)
h <- hazard(x)
h(1)
h(3)
```

## infimum *Get the infimum of the support.*

### Description

Get the infimum of the support.

### Usage

```
infimum(object)
```

### Arguments

| | |
|---|---|
| object | A support object. |

### Value

The infimum (greatest lower bound) of the support.

### Examples

```
I <- interval$new(0, 10)
infimum(I)  # 0

S <- finite_set$new(c(3, 7, 11))
infimum(S)  # 3
```

## infimum.countable_set *Get the infimum of a countable set.*

### Description

Get the infimum of a countable set.

### Usage

```
## S3 method for class 'countable_set'
infimum(object)
```

### Arguments

| | |
|---|---|
| object | A countable_set object. |

### Value

The lower bound (integer).

## Examples

```
cs <- countable_set$new(0L)
infimum(cs)  # 0
```

---

`infimum.finite_set` *Return the infimum of the finite set.*

---

## Description

Return the infimum of the finite set.

## Usage

```
## S3 method for class 'finite_set'
infimum(object)
```

## Arguments

| | |
|---|---|
| `object` | A finite set. |

## Value

Numeric; the minimum value(s).

## Examples

```
fs <- finite_set$new(c(1, 3, 5, 7))
infimum(fs) # 1
```

---

`infimum.interval` *Return the (vector of) infimum of the interval.*

---

## Description

Return the (vector of) infimum of the interval.

## Usage

```
## S3 method for class 'interval'
infimum(object)
```

## Arguments

| | |
|---|---|
| `object` | An interval object. |

### Value

Numeric vector of lower bounds.

### Examples

```
iv <- interval$new(lower = 0, upper = 1)
infimum(iv) # 0
```

---

## interval *Interval*

---

### Description

An interval is a support that is a finite union of intervals.

### Public fields

`lower` A numeric vector of lower bounds.

`upper` A numeric vector of upper bounds.

`lower_closed` A logical vector indicating whether the lower bound is closed.

`upper_closed` A logical vector indicating whether the upper bound is closed.

### Methods

#### Public methods:

- `interval$new()`
- `interval$is_empty()`
- `interval$has()`
- `interval$infimum()`
- `interval$supremum()`
- `interval$dim()`
- `interval$clone()`

**Method** `new()`**:** Initialize an interval.

*Usage:*

```
interval$new(
  lower = -Inf,
  upper = Inf,
  lower_closed = FALSE,
  upper_closed = FALSE
)
```

*Arguments:*

`lower` A numeric vector of lower bounds.

`upper` A numeric vector of upper bounds.

`lower_closed` A logical vector indicating whether the lower bound is closed.

`upper_closed` A logical vector indicating whether the upper bound is closed.

**Method** `is_empty()`**:** Determine if the interval is empty

*Usage:*

```
interval$is_empty()
```

*Returns:* A logical vector indicating whether the interval is empty.

**Method** `has()`**:** Determine if a value is contained in the interval.

*Usage:*

```
interval$has(x)
```

*Arguments:*

`x` A numeric vector of values.

*Returns:* A logical vector indicating whether each value is contained

**Method** `infimum()`**:** Get the infimum of the interval.

*Usage:*

```
interval$infimum()
```

*Returns:* A numeric vector of infimums.

**Method** `supremum()`**:** Get the supremum of the interval.

*Usage:*

```
interval$supremum()
```

*Returns:* A numeric vector of supremums.

**Method** `dim()`**:** Get the dimension of the interval.

*Usage:*

```
interval$dim()
```

*Returns:* The dimension of the interval.

**Method** `clone()`**:** The objects of this class are cloneable with this method.

*Usage:*

```
interval$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## inv_cdf *Generic method for obtaining the quantile (inverse cdf) of an object.*

### Description

Generic method for obtaining the quantile (inverse cdf) of an object.

### Usage

```
inv_cdf(x, ...)
```

### Arguments

| | |
|---|---|
| x | The object to obtain the quantile of. |
| ... | Additional arguments to pass. |

### Value

A function computing the quantile (inverse CDF).

### Examples

```
x <- normal(0, 1)
Q <- inv_cdf(x)
Q(0.5)   # 0 (median of standard normal)
Q(0.975) # approximately 1.96
```

## inv_cdf.beta_dist *Inverse CDF (quantile function) for a beta distribution.*

### Description

Returns a function that computes quantiles of the beta distribution.

### Usage

```
## S3 method for class 'beta_dist'
inv_cdf(x, ...)
```

### Arguments

| | |
|---|---|
| x | A beta_dist object. |
| ... | Additional arguments (not used). |

## Value

A function `function(p, lower.tail = TRUE, log.p = FALSE, ...)` returning the quantile at probability `p`.

## Examples

```
x <- beta_dist(2, 5)
q <- inv_cdf(x)
q(0.5)
q(0.95)
```

---

`inv_cdf.chi_squared` *Method for obtaining the inverse cdf (quantile function) of a* `chi_squared` *object.*

---

## Description

Method for obtaining the inverse cdf (quantile function) of a `chi_squared` object.

## Usage

```
## S3 method for class 'chi_squared'
inv_cdf(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | The `chi_squared` object |
| `...` | Additional arguments (not used) |

## Value

A function that computes the quantile at probability `p`

## Examples

```
x <- chi_squared(5)
q <- inv_cdf(x)
q(0.5)
q(0.95)
```

## inv_cdf.edist *Inverse CDF (quantile function) for expression distributions.*

### Description

Falls back to `realize` and delegates to `inv_cdf.empirical_dist`.

### Usage

```
## S3 method for class 'edist'
inv_cdf(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | An `edist` object. |
| `...` | Additional arguments forwarded to `inv_cdf.empirical_dist`. |

### Value

A function computing the empirical quantile function.

### Examples

```
set.seed(1)
z <- normal(0, 1) * exponential(1)
qz <- inv_cdf(z)
qz(0.5)
```

## inv_cdf.empirical_dist *Method for obtaining the inverse CDF (quantile function) of a univariate* `empirical_dist` *object.*

### Description

Uses the empirical quantile function from the observed data.

### Usage

```
## S3 method for class 'empirical_dist'
inv_cdf(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | The empirical distribution object. |
| `...` | Additional arguments (not used). |

## Value

A function that accepts a vector of probabilities p and returns the corresponding quantiles.

## Examples

```
ed <- empirical_dist(c(1, 2, 3, 4, 5))
qf <- inv_cdf(ed)
qf(0.5)       # median
qf(c(0.25, 0.75)) # quartiles
```

---

# inv_cdf.exponential *Method to obtain the inverse cdf of an* exponential *object.*

---

## Description

Method to obtain the inverse cdf of an exponential object.

## Usage

```
## S3 method for class 'exponential'
inv_cdf(x, ...)
```

## Arguments

| | |
|---|---|
| x | The object to obtain the inverse cdf of |
| ... | Additional arguments (not used) |

## Value

A function function(p, lower.tail = TRUE, log.p = FALSE, ...) that computes the inverse cdf of the exponential distribution.

## Examples

```
x <- exponential(rate = 1)
q <- inv_cdf(x)
q(0.5)
q(0.95)
```

## inv_cdf.gamma_dist *Method for obtaining the inverse cdf (quantile function) of a* gamma_dist *object.*

### Description

Method for obtaining the inverse cdf (quantile function) of a gamma_dist object.

### Usage

```
## S3 method for class 'gamma_dist'
inv_cdf(x, ...)
```

### Arguments

| | |
|---|---|
| x | The gamma_dist object |
| ... | Additional arguments (not used) |

### Value

A function that computes the quantile at probability p

### Examples

```
x <- gamma_dist(shape = 2, rate = 1)
q <- inv_cdf(x)
q(0.5)
q(0.95)
```

## inv_cdf.lognormal *Inverse CDF (quantile function) for a log-normal distribution.*

### Description

Returns a function that computes quantiles of the log-normal distribution.

### Usage

```
## S3 method for class 'lognormal'
inv_cdf(x, ...)
```

### Arguments

| | |
|---|---|
| x | A lognormal object. |
| ... | Additional arguments (not used). |

## Value

A function `function(p, lower.tail = TRUE, log.p = FALSE, ...)` returning the quantile at probability p.

## Examples

```
x <- lognormal(0, 1)
q <- inv_cdf(x)
q(0.5)
q(0.95)
```

---

`inv_cdf.normal` *Method for obtaining the inverse cdf of an* `normal` *object.*

---

## Description

Method for obtaining the inverse cdf of an `normal` object.

## Usage

```
## S3 method for class 'normal'
inv_cdf(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | The object to obtain the inverse cdf of |
| `...` | Additional arguments to pass (not used) |

## Value

A function `function(p, lower.tail = TRUE, log.p = FALSE, ...)` that computes the inverse cdf of the normal distribution.

## Examples

```
x <- normal(0, 1)
q <- inv_cdf(x)
q(0.5)
q(0.975)
```

## inv_cdf.poisson_dist *Inverse CDF (quantile function) for a Poisson distribution.*

### Description

Returns a function that computes quantiles of the Poisson distribution.

### Usage

```
## S3 method for class 'poisson_dist'
inv_cdf(x, ...)
```

### Arguments

| | |
|---|---|
| x | A poisson_dist object. |
| ... | Additional arguments (not used). |

### Value

A function function(p, lower.tail = TRUE, log.p = FALSE, ...) returning the quantile at probability p.

### Examples

```
x <- poisson_dist(5)
q <- inv_cdf(x)
q(0.5)
q(0.95)
```

## inv_cdf.uniform_dist *Inverse CDF (quantile function) for a uniform distribution.*

### Description

Returns a function that computes quantiles of the uniform distribution.

### Usage

```
## S3 method for class 'uniform_dist'
inv_cdf(x, ...)
```

### Arguments

| | |
|---|---|
| x | A uniform_dist object. |
| ... | Additional arguments (not used). |

## Value

A function `function(p, lower.tail = TRUE, log.p = FALSE, ...)` returning the quantile at probability p.

## Examples

```
x <- uniform_dist(0, 10)
q <- inv_cdf(x)
q(0.5)
q(0.9)
```

---

`inv_cdf.weibull_dist` *Inverse CDF (quantile function) for a Weibull distribution.*

---

## Description

Returns a function that computes quantiles of the Weibull distribution.

## Usage

```
## S3 method for class 'weibull_dist'
inv_cdf(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A `weibull_dist` object. |
| `...` | Additional arguments (not used). |

## Value

A function `function(p, lower.tail = TRUE, log.p = FALSE, ...)` returning the quantile at probability p.

## Examples

```
x <- weibull_dist(shape = 2, scale = 3)
q <- inv_cdf(x)
q(0.5)
q(0.95)
```

## is_beta_dist *Test whether an object is a* beta_dist.

### Description

Test whether an object is a beta_dist.

### Usage

```
is_beta_dist(x)
```

### Arguments

| | |
|---|---|
| x | The object to test. |

### Value

TRUE if x inherits from "beta_dist", FALSE otherwise.

### Examples

```
is_beta_dist(beta_dist(2, 5))
is_beta_dist(normal(0, 1))
```

## is_chi_squared *Test whether an object is a* chi_squared.

### Description

Test whether an object is a chi_squared.

### Usage

```
is_chi_squared(x)
```

### Arguments

| | |
|---|---|
| x | The object to test |

### Value

Logical; TRUE if x inherits from chi_squared

### Examples

```
is_chi_squared(chi_squared(3))
is_chi_squared(normal(0, 1))
```

## is_dist *Function to determine whether an object* x *is a* dist *object.*

### Description

Function to determine whether an object x is a dist object.

### Usage

```
is_dist(x)
```

### Arguments

| | |
|---|---|
| x | The object to test |

### Value

Logical indicating whether x is a dist object.

### Examples

```
is_dist(normal(0, 1))   # TRUE
is_dist(42)             # FALSE
```

## is_edist *Function to determine whether an object* x *is an* edist *object.*

### Description

Function to determine whether an object x is an edist object.

### Usage

```
is_edist(x)
```

### Arguments

| | |
|---|---|
| x | The object to test |

### Value

Logical; TRUE if x is an edist.

### Examples

```
is_edist(normal(0, 1) * exponential(1))  # TRUE
is_edist(normal(0, 1))                   # FALSE
```

## is_empirical_dist *Function to determine whether an object* x *is an* empirical_dist *object.*

### Description

Function to determine whether an object x is an empirical_dist object.

### Usage

```
is_empirical_dist(x)
```

### Arguments

| | |
|---|---|
| x | The object to test |

### Value

Logical; TRUE if x is an empirical_dist.

### Examples

```
ed <- empirical_dist(c(1, 2, 3))
is_empirical_dist(ed)    # TRUE
is_empirical_dist("abc") # FALSE
```

## is_exponential *Function to determine whether an object* x *is an* exponential *object.*

### Description

Function to determine whether an object x is an exponential object.

### Usage

```
is_exponential(x)
```

### Arguments

| | |
|---|---|
| x | The object to test |

### Value

Logical; TRUE if x is an exponential.

### Examples

```
is_exponential(exponential(1))
is_exponential(normal(0, 1))
```

## is_gamma_dist *Test whether an object is a* gamma_dist.

### Description

Test whether an object is a gamma_dist.

### Usage

```
is_gamma_dist(x)
```

### Arguments

| | |
|---|---|
| x | The object to test |

### Value

Logical; TRUE if x inherits from gamma_dist

### Examples

```
is_gamma_dist(gamma_dist(2, 1))
is_gamma_dist(normal(0, 1))
```

## is_lognormal *Test whether an object is a* lognormal.

### Description

Test whether an object is a lognormal.

### Usage

```
is_lognormal(x)
```

### Arguments

| | |
|---|---|
| x | The object to test. |

### Value

TRUE if x inherits from "lognormal", FALSE otherwise.

### Examples

```
is_lognormal(lognormal(0, 1))
is_lognormal(normal(0, 1))
```

## is_mixture *Test whether an object is a* mixture *distribution.*

### Description

Test whether an object is a mixture distribution.

### Usage

```
is_mixture(x)
```

### Arguments

| | |
|---|---|
| x | The object to test. |

### Value

TRUE if x inherits from "mixture", FALSE otherwise.

### Examples

```
m <- mixture(list(normal(0, 1), normal(5, 2)), c(0.5, 0.5))
is_mixture(m)
is_mixture(normal(0, 1))
```

## is_mvn *Function to determine whether an object* x *is an* mvn *object.*

### Description

Function to determine whether an object x is an mvn object.

### Usage

```
is_mvn(x)
```

### Arguments

| | |
|---|---|
| x | The object to test |

### Value

Logical; TRUE if x is an mvn.

### Examples

```
is_mvn(mvn(c(0, 0)))
is_mvn(normal(0, 1))
```

## is_normal *Function to determine whether an object* x *is an* normal *object.*

### Description

Function to determine whether an object x is an normal object.

### Usage

```
is_normal(x)
```

### Arguments

| | |
|---|---|
| x | The object to test |

### Value

Logical; TRUE if x is a normal.

### Examples

```
is_normal(normal(0, 1))
is_normal(exponential(1))
```

## is_poisson_dist *Test whether an object is a* poisson_dist.

### Description

Test whether an object is a poisson_dist.

### Usage

```
is_poisson_dist(x)
```

### Arguments

| | |
|---|---|
| x | The object to test. |

### Value

TRUE if x inherits from "poisson_dist", FALSE otherwise.

### Examples

```
is_poisson_dist(poisson_dist(5))
is_poisson_dist(normal(0, 1))
```

## is_realized_dist *Test whether an object is a* realized_dist.

### Description

Test whether an object is a realized_dist.

### Usage

```
is_realized_dist(x)
```

### Arguments

| | |
|---|---|
| x | The object to test. |

### Value

TRUE if x inherits from "realized_dist", FALSE otherwise.

### Examples

```
rd <- realize(normal(0, 1), n = 100)
is_realized_dist(rd)  # TRUE

is_realized_dist(normal(0, 1))  # FALSE
```

## is_uniform_dist *Test whether an object is a* uniform_dist.

### Description

Test whether an object is a uniform_dist.

### Usage

```
is_uniform_dist(x)
```

### Arguments

| | |
|---|---|
| x | The object to test. |

### Value

TRUE if x inherits from "uniform_dist", FALSE otherwise.

### Examples

```
is_uniform_dist(uniform_dist(0, 1))
is_uniform_dist(normal(0, 1))
```

## is_weibull_dist *Test whether an object is a* weibull_dist.

### Description

Test whether an object is a weibull_dist.

### Usage

```
is_weibull_dist(x)
```

### Arguments

| | |
|---|---|
| x | The object to test. |

### Value

TRUE if x inherits from "weibull_dist", FALSE otherwise.

### Examples

```
is_weibull_dist(weibull_dist(2, 3))
is_weibull_dist(normal(0, 1))
```

## lln *Law of Large Numbers Limiting Distribution*

### Description

Returns the degenerate limiting distribution of the sample mean $\bar{X}_n$ under the Law of Large Numbers. The limit is a point mass at the population mean (represented as a normal or mvn with zero variance).

### Usage

```
lln(base_dist)
```

### Arguments

| | |
|---|---|
| base_dist | A dist object representing the base distribution. |

### Value

A normal or mvn distribution with zero variance, representing the degenerate distribution at the mean.

## Examples

```
# LLN for Exp(2): Xbar -> 1/2 (degenerate)
x <- exponential(rate = 2)
d <- lln(x)
mean(d)
vcov(d)
```

---

`lognormal` *Construct a log-normal distribution object.*

---

## Description

Creates an S3 object representing a log-normal distribution with the given `meanlog` and `sdlog` parameters. The log-normal PDF is

$$f(t) = \frac{1}{t \cdot sdlog\sqrt{2\pi}} \exp\left(-\frac{(\log t - meanlog)^2}{2 \cdot sdlog^2}\right)$$

for $t > 0$.

## Usage

```
lognormal(meanlog = 0, sdlog = 1)
```

## Arguments

| | |
|---|---|
| `meanlog` | Mean of the distribution on the log scale (default 0). |
| `sdlog` | Standard deviation on the log scale (default 1), must be positive. |

## Value

A `lognormal` object with classes `c("lognormal", "univariate_dist", "continuous_dist", "dist")`.

## Examples

```
x <- lognormal(meanlog = 0, sdlog = 1)
mean(x)
vcov(x)
format(x)
```

`marginal` *Generic method for obtaining the marginal distribution of a distribution object* `x` *over components* `indices`.

## Description

Generic method for obtaining the marginal distribution of a distribution object `x` over components `indices`.

## Usage

```
marginal(x, indices)
```

## Arguments

| | |
|---|---|
| `x` | The distribution object. |
| `indices` | The indices of the marginal distribution to obtain. |

## Value

A distribution object for the marginal over `indices`.

## Examples

```
x <- mvn(c(0, 0), diag(2))
m <- marginal(x, 1)  # marginal over first component
mean(m)              # 0
```

`marginal.empirical_dist` *Method for obtaining the marginal distribution of* `empirical_dist` *object* `x`.

## Description

Method for obtaining the marginal distribution of `empirical_dist` object `x`.

## Usage

```
## S3 method for class 'empirical_dist'
marginal(x, indices)
```

## Arguments

| | |
|---|---|
| `x` | The empirical distribution object. |
| `indices` | The indices of the marginal distribution to obtain. |

### Value

An `empirical_dist` over the selected columns.

### Examples

```
mat <- matrix(1:12, ncol = 3)
ed <- empirical_dist(mat)
ed_marginal <- marginal(ed, c(1, 3))
dim(ed_marginal) # 2
```

---

## `marginal.mixture` *Marginal distribution of a mixture.*

---

### Description

The marginal of a mixture is itself a mixture of the component marginals with the same mixing weights: $p(x_I) = \sum_k w_k p_k(x_I)$.

### Usage

```
## S3 method for class 'mixture'
marginal(x, indices)
```

### Arguments

| | |
|---|---|
| `x` | A `mixture` object. |
| `indices` | Integer vector of variable indices to keep. |

### Details

Requires all components to support `marginal`.

### Value

A `mixture` object with marginalized components.

### Examples

```
# Mixture of bivariate normals, extract marginal over first variable
m <- mixture(
  list(mvn(c(0, 0), diag(2)), mvn(c(3, 3), diag(2))),
  c(0.5, 0.5)
)
m1 <- marginal(m, 1)
mean(m1)
```

`marginal.mvn` *Generic method for obtaining the marginal distribution of an* `mvn` *object* `x` *over components* `indices`.

## Description

Generic method for obtaining the marginal distribution of an `mvn` object `x` over components `indices`.

## Usage

```
## S3 method for class 'mvn'
marginal(x, indices)
```

## Arguments

| | |
|---|---|
| `x` | The `mvn` object. |
| `indices` | The indices of the marginal distribution to obtain. |

## Value

A `normal` (for a single index) or `mvn` marginal distribution.

## Examples

```
X <- mvn(c(1, 2, 3))
# Univariate marginal
marginal(X, 1)
# Bivariate marginal
marginal(X, c(1, 3))
```

`Math.dist` *Math group generic for distribution objects.*

## Description

Handles exp(), log(), sqrt(), abs(), cos(), sin(), etc.

## Usage

```
## S3 method for class 'dist'
Math(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | a dist object |
| `...` | additional arguments |

## Value

A simplified distribution or edist

## Examples

```
# exp(Normal) simplifies to LogNormal
z <- exp(normal(0, 1))
z

# sqrt of a distribution (no closed-form rule, remains edist)
w <- sqrt(exponential(1))
is_edist(w)  # TRUE
```

---

`mean.beta_dist` *Mean of a beta distribution.*

---

## Description

Computes $\alpha/(\alpha+\beta)$ where $\alpha$ = shape1 and $\beta$ = shape2.

## Usage

```
## S3 method for class 'beta_dist'
mean(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A `beta_dist` object. |
| `...` | Additional arguments (not used). |

## Value

The mean of the distribution.

## Examples

```
mean(beta_dist(2, 5))
```

## mean.chi_squared *Retrieve the mean of a* chi_squared *object.*

### Description

Retrieve the mean of a chi_squared object.

### Usage

```
## S3 method for class 'chi_squared'
mean(x, ...)
```

### Arguments

| | |
|---|---|
| x | The chi_squared object |
| ... | Additional arguments (not used) |

### Value

The mean, equal to df

### Examples

```
mean(chi_squared(10))
```

## mean.edist *Method for obtaining the mean of an* edist *object.*

### Description

Method for obtaining the mean of an edist object.

### Usage

```
## S3 method for class 'edist'
mean(x, n = 10000, ...)
```

### Arguments

| | |
|---|---|
| x | The edist object to retrieve the mean from |
| n | The number of samples to take (default: 10000) |
| ... | Additional arguments to pass (not used) |

### Value

The mean of the edist object

### Examples

```
set.seed(1)
z <- normal(0, 1) * exponential(2)
mean(z)
```

---

## mean.empirical_dist *Method for obtaining the mean of* empirical_dist *object* x.

---

### Description

Method for obtaining the mean of empirical_dist object x.

### Usage

```
## S3 method for class 'empirical_dist'
mean(x, ...)
```

### Arguments

| | |
|---|---|
| x | The distribution object. |
| ... | Additional arguments to pass (not used). |

### Value

Numeric vector of column means.

### Examples

```
ed <- empirical_dist(c(1, 2, 3, 4, 5))
mean(ed) # 3
```

---

## mean.exponential *Method to obtain the mean of an* exponential *object.*

---

### Description

Method to obtain the mean of an exponential object.

### Usage

```
## S3 method for class 'exponential'
mean(x, ...)
```

### Arguments

| | |
|---|---|
| x | The exponential object to obtain the mean of |
| ... | Additional arguments (not used) |

### Value

The mean of the exponential distribution (1 / rate).

### Examples

```
x <- exponential(rate = 0.5)
mean(x)
```

---

## mean.gamma_dist *Retrieve the mean of a* gamma_dist *object.*

---

### Description

Retrieve the mean of a gamma_dist object.

### Usage

```
## S3 method for class 'gamma_dist'
mean(x, ...)
```

### Arguments

| | |
|---|---|
| x | The gamma_dist object |
| ... | Additional arguments (not used) |

### Value

The mean, shape / rate

### Examples

```
mean(gamma_dist(shape = 3, rate = 2))
```

## mean.lognormal *Mean of a log-normal distribution.*

### Description

Computes $\exp(meanlog + sdlog^2/2)$.

### Usage

```
## S3 method for class 'lognormal'
mean(x, ...)
```

### Arguments

| | |
|---|---|
| x | A lognormal object. |
| ... | Additional arguments (not used). |

### Value

The mean of the distribution.

### Examples

```
mean(lognormal(0, 1))
```

## mean.mixture *Mean of a mixture distribution.*

### Description

The mean of a mixture is the weighted sum of the component means: $E[X] = \sum_k w_k \mu_k$.

### Usage

```
## S3 method for class 'mixture'
mean(x, ...)
```

### Arguments

| | |
|---|---|
| x | A mixture object. |
| ... | Additional arguments (not used). |

### Value

The mean of the mixture distribution.

## Examples

```
m <- mixture(list(normal(0, 1), normal(10, 1)), c(0.5, 0.5))
mean(m)
```

---

`mean.mvn` *Retrieve the mean of a* `mvn` *object.*

---

## Description

Retrieve the mean of a `mvn` object.

## Usage

```
## S3 method for class 'mvn'
mean(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | The `mvn` object to retrieve the mean from |
| `...` | Additional arguments to pass (not used) |

## Value

The mean of the `mvn` object

## Examples

```
X <- mvn(c(1, 2, 3))
mean(X)
```

---

`mean.normal` *Retrieve the mean of a* `normal` *object.*

---

## Description

Retrieve the mean of a `normal` object.

## Usage

```
## S3 method for class 'normal'
mean(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | The `normal` object to retrieve the mean from |
| `...` | Additional arguments to pass (not used) |

### Value

The mean of the `normal` object

### Examples

```
x <- normal(5, 2)
mean(x)
```

---

## `mean.poisson_dist` *Mean of a Poisson distribution.*

---

### Description

Mean of a Poisson distribution.

### Usage

```
## S3 method for class 'poisson_dist'
mean(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | A `poisson_dist` object. |
| `...` | Additional arguments (not used). |

### Value

The mean, equal to `lambda`.

### Examples

```
mean(poisson_dist(5))
```

---

## `mean.uniform_dist` *Mean of a uniform distribution.*

---

### Description

Computes $(min + max)/2$.

### Usage

```
## S3 method for class 'uniform_dist'
mean(x, ...)
```

### Arguments

| | |
|---|---|
| x | A uniform_dist object. |
| ... | Additional arguments (not used). |

### Value

The mean of the distribution.

### Examples

```
mean(uniform_dist(0, 10))
```

---

## mean.univariate_dist *Method for obtaining the mean of* univariate_dist *object* x.

---

### Description

Method for obtaining the mean of univariate_dist object x.

### Usage

```
## S3 method for class 'univariate_dist'
mean(x, ...)
```

### Arguments

| | |
|---|---|
| x | The distribution object. |
| ... | Additional arguments to pass into expectation. |

### Value

Numeric scalar; the mean of the distribution.

### Examples

```
mean(normal(5, 2))    # 5
mean(exponential(2))  # 0.5
```

## mean.weibull_dist *Mean of a Weibull distribution.*

### Description

Computes $scale \cdot \Gamma(1 + 1/shape)$.

### Usage

```
## S3 method for class 'weibull_dist'
mean(x, ...)
```

### Arguments

| | |
|---|---|
| x | A weibull_dist object. |
| ... | Additional arguments (not used). |

### Value

The mean of the distribution.

### Examples

```
mean(weibull_dist(shape = 2, scale = 3))
```

## mixture *Construct a mixture distribution.*

### Description

Creates an S3 object representing a finite mixture distribution. The density is $f(x) = \sum_{k=1}^{K} w_k f_k(x)$ where $f_k$ are the component densities and $w_k$ are the mixing weights.

### Usage

```
mixture(components, weights)
```

### Arguments

| | |
|---|---|
| components | A non-empty list of dist objects. |
| weights | A numeric vector of non-negative mixing weights that sum to 1 (within tolerance 1e-10). Must have the same length as components. |

### Details

The class hierarchy is determined by the components: if all components are univariate (or multivariate, continuous, discrete), the mixture inherits those classes as well.

## Value

A `mixture` object with appropriate class hierarchy.

## Examples

```
m <- mixture(
  components = list(normal(0, 1), normal(5, 2)),
  weights = c(0.3, 0.7)
)
mean(m)
vcov(m)
format(m)
```

---

`mvn` *Construct a multivariate or univariate normal distribution object.*

---

## Description

This function constructs an object representing a normal distribution. If the length of the mean vector `mu` is 1, it creates a univariate normal distribution. Otherwise, it creates a multivariate normal distribution.

## Usage

```
mvn(mu, sigma = diag(length(mu)))
```

## Arguments

| | |
|---|---|
| `mu` | A numeric vector specifying the means of the distribution. If `mu` has length 1, a univariate normal distribution is created. If `mu` has length > 1, a multivariate normal distribution is created. |
| `sigma` | A numeric matrix specifying the variance-covariance matrix of the distribution. It must be a square matrix with the same number of rows and columns as the length of `mu`. Default is the identity matrix of size equal to the length of `mu`. |

## Value

If `mu` has length 1, it returns a `normal` object. If `mu` has length > 1, it returns an `mvn` object. Both types of objects contain `mu` and `sigma` as their properties.

## Examples

```
# Bivariate normal with identity covariance
X <- mvn(mu = c(0, 0))
mean(X)
vcov(X)

# 1D case returns a normal object
is_normal(mvn(mu = 1, sigma = matrix(4)))
```

## nobs.empirical_dist *Method for obtaining the number of observations used to construct a* empirical_dist *object.*

### Description

Method for obtaining the number of observations used to construct a empirical_dist object.

### Usage

```
## S3 method for class 'empirical_dist'
nobs(object, ...)
```

### Arguments

| | |
|---|---|
| object | The empirical distribution object. |
| ... | Additional arguments to pass (not used). |

### Value

Integer; number of observations.

### Examples

```
ed <- empirical_dist(c(10, 20, 30, 40))
nobs(ed) # 4
```

## normal *Construct univariate normal distribution object.*

### Description

Construct univariate normal distribution object.

### Usage

```
normal(mu = 0, var = 1)
```

### Arguments

| | |
|---|---|
| mu | mean |
| var | variance |

### Value

A normal distribution object.

## Examples

```
x <- normal(mu = 0, var = 1)
mean(x)
vcov(x)
format(x)
```

---

`normal_approx` *Moment-Matching Normal Approximation*

---

## Description

Constructs a normal (or multivariate normal) distribution that matches the mean and variance-covariance of the input distribution. This is useful as a quick Gaussian approximation for any distribution whose first two moments are available.

## Usage

```
normal_approx(x)
```

## Arguments

| | |
|---|---|
| `x` | A `dist` object to approximate. |

## Value

A `normal` distribution (for univariate inputs) or an `mvn` distribution (for multivariate inputs) with the same mean and variance-covariance as `x`.

## Examples

```
# Approximate a Gamma(5, 2) with a normal
g <- gamma_dist(shape = 5, rate = 2)
n <- normal_approx(g)
mean(n)
vcov(n)
```

---

`nparams` *Generic method for obtaining the number of parameters of distribution-like object* `x`.

---

## Description

Generic method for obtaining the number of parameters of distribution-like object `x`.

### Usage

```
nparams(x)

## S3 method for class 'dist'
nparams(x)
```

### Arguments

| | |
|---|---|
| x | the object to obtain the number of parameters for |

### Value

Integer; the number of parameters.

### Examples

```
d <- empirical_dist(matrix(rnorm(30), ncol = 3))
nparams(d)  # 0 (non-parametric)

nparams(normal(0, 1))  # 2
```

---

## nparams.empirical_dist

*Method for obtaining the name of a* `empirical_dist` *object. Since the empirical distribution is parameter-free, this function returns 0.*

---

### Description

Method for obtaining the name of a `empirical_dist` object. Since the empirical distribution is parameter-free, this function returns 0.

### Usage

```
## S3 method for class 'empirical_dist'
nparams(x)
```

### Arguments

| | |
|---|---|
| x | The empirical distribution object. |

### Value

0 (empirical distributions are non-parametric).

### Examples

```
ed <- empirical_dist(c(1, 2, 3))
nparams(ed) # 0
```

## nparams.mixture *Number of parameters for a* mixture *distribution.*

### Description

The total number of parameters is the sum of component parameters plus the number of mixing weights.

### Usage

```
## S3 method for class 'mixture'
nparams(x)
```

### Arguments

| | |
|---|---|
| x | A mixture object. |

### Value

An integer count of parameters.

### Examples

```
m <- mixture(list(normal(0, 1), normal(5, 2)), c(0.3, 0.7))
nparams(m)
```

## obs *Retrieve the observations used to construct a distribution-like object. This is useful for obtaining the data used to construct an empirical distribution, but it is also useful for, say, retrieving the sample that was used by a fitted object, like an maximum likelihood estimate.*

### Description

Retrieve the observations used to construct a distribution-like object. This is useful for obtaining the data used to construct an empirical distribution, but it is also useful for, say, retrieving the sample that was used by a fitted object, like an maximum likelihood estimate.

### Usage

```
obs(x)
```

### Arguments

| | |
|---|---|
| x | the object to retrieve the observations from |

### Value

The data (matrix or vector) used to construct `x`.

### Examples

```
d <- empirical_dist(1:10)
obs(d)  # returns the vector 1:10
```

---

## obs.empirical_dist *Method for obtaining the observations used to construct a* `empirical_dist` *object.*

---

### Description

Method for obtaining the observations used to construct a `empirical_dist` object.

### Usage

```
## S3 method for class 'empirical_dist'
obs(x)
```

### Arguments

| | |
|---|---|
| `x` | The empirical distribution object. |

### Value

A matrix of observations (rows = observations, columns = dimensions).

### Examples

```
ed <- empirical_dist(c(5, 10, 15))
obs(ed)
```

---

## params *Generic method for obtaining the parameters of an object.*

---

### Description

Generic method for obtaining the parameters of an object.

### Usage

```
params(x)
```

### Arguments

| | |
|---|---|
| x | The object to obtain the parameters of. |

### Value

A named vector (or list) of distribution parameters.

### Examples

```
x <- normal(5, 2)
params(x)  # mu = 5, var = 2

y <- exponential(3)
params(y)  # rate = 3
```

---

## params.beta_dist *Retrieve the parameters of a* beta_dist *object.*

---

### Description

Retrieve the parameters of a beta_dist object.

### Usage

```
## S3 method for class 'beta_dist'
params(x)
```

### Arguments

| | |
|---|---|
| x | A beta_dist object. |

### Value

A named numeric vector with elements shape1 and shape2.

### Examples

```
params(beta_dist(2, 5))
```

## params.chi_squared *Method for obtaining the parameters of a* chi_squared *object.*

### Description

Method for obtaining the parameters of a chi_squared object.

### Usage

```
## S3 method for class 'chi_squared'
params(x)
```

### Arguments

| | |
|---|---|
| x | The chi_squared object |

### Value

A named numeric vector of parameters

### Examples

```
params(chi_squared(5))
```

## params.edist *Method for obtaining the parameters of an* edist *object.*

### Description

Method for obtaining the parameters of an edist object.

### Usage

```
## S3 method for class 'edist'
params(x)
```

### Arguments

| | |
|---|---|
| x | The object to obtain the parameters of |

### Value

A named vector of parameters

### Examples

```
z <- normal(0, 1) * exponential(2)
params(z)
```

## params.empirical_dist *empirical_dist objects have no parameters, so this function returns NULL.*

### Description

empirical_dist objects have no parameters, so this function returns NULL.

### Usage

```
## S3 method for class 'empirical_dist'
params(x)
```

### Arguments

| | |
|---|---|
| x | The empirical distribution object. |

### Value

NULL (empirical distributions have no parameters).

### Examples

```
ed <- empirical_dist(c(1, 2, 3))
params(ed) # NULL
```

## params.exponential *Method for obtaining the parameters of an exponential object.*

### Description

Method for obtaining the parameters of an exponential object.

### Usage

```
## S3 method for class 'exponential'
params(x)
```

### Arguments

| | |
|---|---|
| x | The object to obtain the parameters of |

### Value

A named vector of parameters

### Examples

```
x <- exponential(rate = 0.5)
params(x)
```

---

## params.gamma_dist *Method for obtaining the parameters of a* gamma_dist *object.*

---

### Description

Method for obtaining the parameters of a gamma_dist object.

### Usage

```
## S3 method for class 'gamma_dist'
params(x)
```

### Arguments

| | |
|---|---|
| x | The gamma_dist object |

### Value

A named numeric vector of parameters

### Examples

```
params(gamma_dist(2, 1))
```

---

## params.lognormal *Retrieve the parameters of a* lognormal *object.*

---

### Description

Retrieve the parameters of a lognormal object.

### Usage

```
## S3 method for class 'lognormal'
params(x)
```

### Arguments

| | |
|---|---|
| x | A lognormal object. |

### Value

A named numeric vector with elements meanlog and sdlog.

## Examples

```
params(lognormal(0, 1))
```

---

`params.mixture` *Retrieve the parameters of a* `mixture` *object.*

---

## Description

Returns a named numeric vector containing all component parameters (flattened) followed by the mixing weights.

## Usage

```
## S3 method for class 'mixture'
params(x)
```

## Arguments

| | |
|---|---|
| `x` | A `mixture` object. |

## Value

A named numeric vector.

## Examples

```
m <- mixture(list(normal(0, 1), normal(5, 2)), c(0.3, 0.7))
params(m)
```

---

`params.mvn` *Method for obtaining the parameters of a* `mvn` *object.*

---

## Description

Method for obtaining the parameters of a `mvn` object.

## Usage

```
## S3 method for class 'mvn'
params(x)
```

## Arguments

| | |
|---|---|
| `x` | The object to obtain the parameters of |

### Value

A named vector of parameters

### Examples

```
X <- mvn(c(0, 0), diag(2))
params(X)
```

---

## params.normal *Method for obtaining the parameters of a* normal *object.*

---

### Description

Method for obtaining the parameters of a normal object.

### Usage

```
## S3 method for class 'normal'
params(x)
```

### Arguments

| | |
|---|---|
| x | The object to obtain the parameters of |

### Value

A named vector of parameters

### Examples

```
x <- normal(3, 2)
params(x)
```

---

## params.poisson_dist *Retrieve the parameters of a* poisson_dist *object.*

---

### Description

Retrieve the parameters of a poisson_dist object.

### Usage

```
## S3 method for class 'poisson_dist'
params(x)
```

### Arguments

| | |
|---|---|
| x | A poisson_dist object. |

### Value

A named numeric vector with element lambda.

### Examples

```
params(poisson_dist(5))
```

---

## params.uniform_dist *Retrieve the parameters of a* uniform_dist *object.*

---

### Description

Retrieve the parameters of a uniform_dist object.

### Usage

```
## S3 method for class 'uniform_dist'
params(x)
```

### Arguments

| | |
|---|---|
| x | A uniform_dist object. |

### Value

A named numeric vector with elements min and max.

### Examples

```
params(uniform_dist(0, 10))
```

## params.weibull_dist *Retrieve the parameters of a* weibull_dist *object.*

### Description

Retrieve the parameters of a weibull_dist object.

### Usage

```
## S3 method for class 'weibull_dist'
params(x)
```

### Arguments

| | |
|---|---|
| x | A weibull_dist object. |

### Value

A named numeric vector with elements shape and scale.

### Examples

```
params(weibull_dist(2, 3))
```

## poisson_dist *Construct a Poisson distribution object.*

### Description

Creates an S3 object representing a Poisson distribution with rate parameter $\lambda$. The PMF is $P(X = k) = \lambda^k e^{-\lambda}/k!$ for $k = 0, 1, 2, \ldots$.

### Usage

```
poisson_dist(lambda)
```

### Arguments

| | |
|---|---|
| lambda | Rate parameter (mean), must be a positive scalar. |

### Value

A poisson_dist object with classes c("poisson_dist", "univariate_dist", "discrete_dist", "dist").

## Examples

```
x <- poisson_dist(lambda = 5)
mean(x)
vcov(x)
format(x)
```

---

`print.beta_dist` *Print a* `beta_dist` *object.*

---

## Description

Print a `beta_dist` object.

## Usage

```
## S3 method for class 'beta_dist'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A `beta_dist` object. |
| `...` | Additional arguments (not used). |

## Value

`x`, invisibly.

## Examples

```
print(beta_dist(2, 5))
```

---

`print.chi_squared` *Print method for* `chi_squared` *objects.*

---

## Description

Print method for `chi_squared` objects.

## Usage

```
## S3 method for class 'chi_squared'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | The `chi_squared` object to print |
| `...` | Additional arguments (not used) |

### Value

x, invisibly.

### Examples

```
print(chi_squared(5))
```

---

## print.edist *Print method for* edist *objects.*

---

### Description

Print method for edist objects.

### Usage

```
## S3 method for class 'edist'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | The object to print |
| ... | Additional arguments to pass (not used) |

### Examples

```
z <- normal(0, 1) * exponential(2)
print(z)
```

---

## print.empirical_dist *Print method for* empirical_dist *objects.*

---

### Description

Print method for empirical_dist objects.

### Usage

```
## S3 method for class 'empirical_dist'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | The object to print |
| ... | Additional arguments to pass |

### Value

x, invisibly.

### Examples

```
ed <- empirical_dist(c(1, 2, 3, 4, 5))
print(ed)
```

---

## print.exponential *Print method for* exponential *objects.*

---

### Description

Print method for exponential objects.

### Usage

```
## S3 method for class 'exponential'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | The exponential object to print. |
| ... | Additional arguments (not used) |

### Value

x, invisibly.

### Examples

```
print(exponential(rate = 2))
```

---

## print.gamma_dist *Print method for* gamma_dist *objects.*

---

### Description

Print method for gamma_dist objects.

### Usage

```
## S3 method for class 'gamma_dist'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | The gamma_dist object to print |
| ... | Additional arguments (not used) |

### Value

x, invisibly.

### Examples

```
print(gamma_dist(2, 1))
```

---

## print.interval *Print the interval.*

---

### Description

Print the interval.

### Usage

```
## S3 method for class 'interval'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | An interval object. |
| ... | Additional arguments. |

### Value

x, invisibly.

### Examples

```
iv <- interval$new(lower = 0, upper = 1, lower_closed = TRUE)
print(iv) # [0, 1)
```

## print.lognormal *Print a* lognormal *object.*

### Description

Print a lognormal object.

### Usage

```
## S3 method for class 'lognormal'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | A lognormal object. |
| ... | Additional arguments (not used). |

### Value

x, invisibly.

### Examples

```
print(lognormal(0, 1))
```

## print.mixture *Print a* mixture *object.*

### Description

Print a mixture object.

### Usage

```
## S3 method for class 'mixture'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | A mixture object. |
| ... | Additional arguments (not used). |

### Value

x, invisibly.

### Examples

```
m <- mixture(list(normal(0, 1), normal(5, 1)), c(0.5, 0.5))
print(m)
```

---

## print.mvn *Method for printing an* mvn *object.*

---

### Description

Method for printing an mvn object.

### Usage

```
## S3 method for class 'mvn'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | The object to print |
| ... | Additional arguments to pass to print |

### Value

x, invisibly.

### Examples

```
print(mvn(c(0, 0)))
```

---

## print.normal *Print method for* normal *objects.*

---

### Description

Print method for normal objects.

### Usage

```
## S3 method for class 'normal'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | The object to print |
| ... | Additional arguments to pass (not used) |

## Value

x, invisibly.

## Examples

```
x <- normal(2, 3)
print(x)
```

---

# print.poisson_dist *Print a* poisson_dist *object.*

---

## Description

Print a poisson_dist object.

## Usage

```
## S3 method for class 'poisson_dist'
print(x, ...)
```

## Arguments

| | |
|---|---|
| x | A poisson_dist object. |
| ... | Additional arguments (not used). |

## Value

x, invisibly.

## Examples

```
print(poisson_dist(5))
```

---

# print.realized_dist *Print a* realized_dist *object.*

---

## Description

Print a realized_dist object.

## Usage

```
## S3 method for class 'realized_dist'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | A realized_dist object. |
| ... | Additional arguments (not used). |

### Value

x, invisibly.

### Examples

```
rd <- realize(normal(0, 1), n = 100)
print(rd)
```

---

## print.summary_dist *Print method for* summary_dist *objects.*

---

### Description

Print method for summary_dist objects.

### Usage

```
## S3 method for class 'summary_dist'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | The object to print |
| ... | Additional arguments |

### Value

x, invisibly.

### Examples

```
s <- summary(normal(5, 2))
print(s)
```

## print.uniform_dist *Print a* uniform_dist *object.*

### Description

Print a uniform_dist object.

### Usage

```
## S3 method for class 'uniform_dist'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | A uniform_dist object. |
| ... | Additional arguments (not used). |

### Value

x, invisibly.

### Examples

```
print(uniform_dist(0, 10))
```

## print.weibull_dist *Print a* weibull_dist *object.*

### Description

Print a weibull_dist object.

### Usage

```
## S3 method for class 'weibull_dist'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | A weibull_dist object. |
| ... | Additional arguments (not used). |

### Value

x, invisibly.

### Examples

```
print(weibull_dist(2, 3))
```

---

`realize` *Materialize any distribution to empirical_dist by sampling.*

---

### Description

`realize` draws `n` samples from a distribution and wraps them in an `empirical_dist`. This is the universal fallback that lets any `dist` object be converted to a discrete approximation on which methods like `cdf`, `density`, and `conditional` are always available.

### Usage

```
realize(x, n = 10000, ...)

## S3 method for class 'dist'
realize(x, n = 10000, ...)

## S3 method for class 'empirical_dist'
realize(x, ...)

## S3 method for class 'realized_dist'
realize(x, n = 10000, ...)
```

### Arguments

| | |
|---|---|
| `x` | A distribution object (inheriting from `dist`). |
| `n` | Number of samples (default: 10000). |
| `...` | Additional arguments passed to methods. |

### Details

For non-empirical distributions, the result is a `realized_dist` that preserves the source distribution as provenance metadata. This enables re-sampling via `realize(x$source, n = ...)` and informative printing.

The `empirical_dist` method is a no-op: the distribution is already materialized.

The `realized_dist` method re-samples from the original source distribution, allowing cheap regeneration with a different sample size.

### Value

An `empirical_dist` (or `realized_dist`) object.

## Examples

```
set.seed(1)
x <- normal(0, 1)
rd <- realize(x, n = 1000)
mean(rd)
```

---

`rmap` *Generic method for applying a map* `f` *to distribution object* `x`.

---

## Description

Generic method for applying a map `f` to distribution object `x`.

## Usage

```
rmap(x, g, ...)
```

## Arguments

| | |
|---|---|
| `x` | The distribution object. |
| `g` | The function to apply. |
| `...` | Additional arguments to pass into `g`. |

## Value

A distribution representing the push-forward of `x` through `g`.

## Examples

```
d <- empirical_dist(1:20)
d_sq <- rmap(d, function(x) x^2)
mean(d_sq)  # E[X^2] for uniform 1..20
```

## rmap.dist *Method for obtaining g(x)) where x is a* dist *object.*

### Description

Falls back to MC: materializes x via ensure_realized() and then applies rmap with g to the resulting empirical distribution.

### Usage

```
## S3 method for class 'dist'
rmap(x, g, n = 10000L, ...)
```

### Arguments

| | |
|---|---|
| x | The distribution object. |
| g | The function to apply to the distribution. |
| n | The number of samples to generate for the MC estimate of the conditional distribution x \| P. Defaults to 10000. |
| ... | additional arguments to pass into g. |

### Value

An empirical_dist of the transformed samples.

### Examples

```
set.seed(1)
x <- exponential(1)
# Distribution of log(X) where X ~ Exp(1)
log_x <- rmap(x, log)
mean(log_x)
```

## rmap.edist *Map function over expression distribution.*

### Description

Falls back to realize and delegates to rmap.empirical_dist.

### Usage

```
## S3 method for class 'edist'
rmap(x, g, ...)
```

### Arguments

| | |
|---|---|
| x | An edist object. |
| g | Function to apply to each observation. |
| ... | Additional arguments forwarded to g. |

### Value

A transformed empirical_dist.

### Examples

```
set.seed(1)
z <- normal(0, 1) * exponential(1)
abs_z <- rmap(z, abs)
mean(abs_z)
```

---

## rmap.empirical_dist *Method for obtaining the empirical distribution of a function of the observations of* empirical_dist *object* x.

---

### Description

Method for obtaining the empirical distribution of a function of the observations of empirical_dist object x.

### Usage

```
## S3 method for class 'empirical_dist'
rmap(x, g, ...)
```

### Arguments

| | |
|---|---|
| x | The empirical distribution object. |
| g | The function to apply to each observation. |
| ... | Additional arguments to pass into function g. |

### Value

An empirical_dist of the transformed observations.

### Examples

```
ed <- empirical_dist(c(1, 2, 3, 4))
ed2 <- rmap(ed, function(x) x^2)
mean(ed2) # mean of 1, 4, 9, 16
```

## rmap.mvn *Computes the distribution of* g(x) *where* x *is an* mvn *object.*

### Description

By the invariance property, if x is an mvn object, then under the right conditions, asymptotically, g(x) is an MVN distributed, g(x) ~ normal(g(mean(x)), sigma) where sigma is the variance-covariance of g(x)

### Usage

```
## S3 method for class 'mvn'
rmap(x, g, n = 10000L, ...)
```

### Arguments

| | |
|---|---|
| x | The mvn object to apply g to |
| g | The function to apply to x |
| n | number of samples to take to estimate distribution of g(x) if method is mc or empirical. Defaults to 10000. |
| ... | additional arguments to pass into the g function. |

### Value

An mvn distribution fitted to the transformed samples.

### Examples

```
X <- mvn(c(1, 2), diag(2))
set.seed(42)
Y <- rmap(X, function(x) x^2)
mean(Y)
```

## sampler *Generic method for sampling from distribution-like objects.*

### Description

It creates a sampler for the x object. It returns a function that accepts a parameter n denoting the number of samples to draw from the x object and also any additional parameters ... are passed to the generated function.

### Usage

```
sampler(x, ...)
```

### Arguments

| | |
|---|---|
| x | the x object to create a sampler for |
| ... | additional arguments to pass |

### Value

A function that takes n and returns n samples.

### Examples

```
x <- normal(0, 1)
samp <- sampler(x)
set.seed(42)
samp(5)  # draw 5 samples from standard normal
```

---

## sampler.beta_dist *Sampler for a beta distribution.*

---

### Description

Returns a function that draws n independent samples from the beta distribution.

### Usage

```
## S3 method for class 'beta_dist'
sampler(x, ...)
```

### Arguments

| | |
|---|---|
| x | A beta_dist object. |
| ... | Additional arguments (not used). |

### Value

A function function(n = 1, ...) returning a numeric vector of length n.

### Examples

```
x <- beta_dist(2, 5)
s <- sampler(x)
set.seed(42)
s(5)
```

## sampler.chi_squared *Method for sampling from a* chi_squared *object.*

### Description

Method for sampling from a chi_squared object.

### Usage

```
## S3 method for class 'chi_squared'
sampler(x, ...)
```

### Arguments

| | |
|---|---|
| x | The chi_squared object to sample from |
| ... | Additional arguments (not used) |

### Value

A function that generates n samples from the chi-squared distribution

### Examples

```
x <- chi_squared(5)
s <- sampler(x)
set.seed(42)
s(5)
```

## sampler.default *Sampler for non-dist objects (degenerate distributions).*

### Description

Sampler for non-dist objects (degenerate distributions).

### Usage

```
## Default S3 method:
sampler(x, ...)
```

### Arguments

| | |
|---|---|
| x | The object to sample from |
| ... | Additional arguments to pass |

## Value

A function that takes n and returns n copies of x

## Examples

```
s <- sampler(5)
s(3)  # returns c(5, 5, 5)
```

---

`sampler.edist` *Method for obtaining the sampler of an* `edist` *object.*

---

## Description

Method for obtaining the sampler of an `edist` object.

## Usage

```
## S3 method for class 'edist'
sampler(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | The `edist` object to obtain the sampler of. |
| `...` | Additional arguments to pass into each of the `sampler` function generators. |

## Value

A function that takes a number of samples `n`, `...` which is passed into the expression `x$e` and returns the result of applying the expression `x$e` to the sampled values.

## Examples

```
set.seed(1)
z <- normal(0, 1) * exponential(2)
s <- sampler(z)
samples <- s(100)
head(samples)
```

## sampler.empirical_dist *Method for obtaining the sampler for a* empirical_dist *object.*

### Description

Method for obtaining the sampler for a empirical_dist object.

### Usage

```
## S3 method for class 'empirical_dist'
sampler(x, ...)
```

### Arguments

| | |
|---|---|
| x | The object to obtain the sampler of. |
| ... | Additional arguments to pass (not used). |

### Value

A function that takes n and returns n resampled observations.

### Examples

```
ed <- empirical_dist(c(10, 20, 30))
s <- sampler(ed)
set.seed(42)
s(5)
```

## sampler.exponential *Method to sample from an* exponential *object.*

### Description

Method to sample from an exponential object.

### Usage

```
## S3 method for class 'exponential'
sampler(x, ...)
```

### Arguments

| | |
|---|---|
| x | The exponential object to sample from. |
| ... | Additional arguments to pass (not used) |

### Value

A function `function(n = 1, ...)` that draws `n` samples from the exponential distribution.

### Examples

```
x <- exponential(rate = 2)
s <- sampler(x)
set.seed(42)
s(5)
```

---

## `sampler.gamma_dist` *Method for sampling from a* `gamma_dist` *object.*

---

### Description

Method for sampling from a `gamma_dist` object.

### Usage

```
## S3 method for class 'gamma_dist'
sampler(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | The `gamma_dist` object to sample from |
| `...` | Additional arguments (not used) |

### Value

A function that generates `n` samples from the gamma distribution

### Examples

```
x <- gamma_dist(shape = 2, rate = 1)
s <- sampler(x)
set.seed(42)
s(5)
```

## sampler.lognormal *Sampler for a log-normal distribution.*

### Description

Returns a function that draws n independent samples from the log-normal distribution.

### Usage

```
## S3 method for class 'lognormal'
sampler(x, ...)
```

### Arguments

| | |
|---|---|
| x | A lognormal object. |
| ... | Additional arguments (not used). |

### Value

A function function(n = 1, ...) returning a numeric vector of length n.

### Examples

```
x <- lognormal(0, 1)
s <- sampler(x)
set.seed(42)
s(5)
```

## sampler.mixture *Sampler for a mixture distribution.*

### Description

Returns a function that draws samples from the mixture by first selecting a component according to the mixing weights, then sampling from the selected component.

### Usage

```
## S3 method for class 'mixture'
sampler(x, ...)
```

### Arguments

| | |
|---|---|
| x | A mixture object. |
| ... | Additional arguments (not used). |

## Value

A function `function(n = 1, ...)` returning a numeric vector of length `n`.

## Examples

```
m <- mixture(list(normal(0, 1), normal(5, 1)), c(0.5, 0.5))
s <- sampler(m)
set.seed(42)
s(6)
```

---

`sampler.mvn` *Function generator for sampling from a* `mvn` *(multivariate normal) object.*

---

## Description

Function generator for sampling from a `mvn` (multivariate normal) object.

## Usage

```
## S3 method for class 'mvn'
sampler(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | The `mvn` object to sample from |
| `...` | Additional arguments passed to `rmvnorm` on every call. |

## Examples

```
X <- mvn(c(0, 0), diag(2))
s <- sampler(X)
set.seed(42)
s(3)
```

---

`sampler.normal` *Method for sampling from a* `normal` *object.*

---

## Description

Method for sampling from a `normal` object.

## Usage

```
## S3 method for class 'normal'
sampler(x, ...)
```

### Arguments

| | |
|---|---|
| x | The normal object to sample from |
| ... | Additional arguments to pass (not used) |

### Value

A function function(n = 1, ...) that draws n samples from the normal distribution.

### Examples

```
x <- normal(0, 1)
s <- sampler(x)
set.seed(42)
s(5)
```

---

## sampler.poisson_dist *Sampler for a Poisson distribution.*

---

### Description

Returns a function that draws n independent samples from the Poisson distribution.

### Usage

```
## S3 method for class 'poisson_dist'
sampler(x, ...)
```

### Arguments

| | |
|---|---|
| x | A poisson_dist object. |
| ... | Additional arguments (not used). |

### Value

A function function(n = 1, ...) returning an integer vector of length n.

### Examples

```
x <- poisson_dist(5)
s <- sampler(x)
set.seed(42)
s(5)
```

## sampler.uniform_dist *Sampler for a uniform distribution.*

### Description

Returns a function that draws n independent samples from the uniform distribution.

### Usage

```
## S3 method for class 'uniform_dist'
sampler(x, ...)
```

### Arguments

| | |
|---|---|
| x | A uniform_dist object. |
| ... | Additional arguments (not used). |

### Value

A function function(n = 1, ...) returning a numeric vector of length n.

### Examples

```
x <- uniform_dist(0, 10)
s <- sampler(x)
set.seed(42)
s(5)
```

## sampler.weibull_dist *Sampler for a Weibull distribution.*

### Description

Returns a function that draws n independent samples from the Weibull distribution.

### Usage

```
## S3 method for class 'weibull_dist'
sampler(x, ...)
```

### Arguments

| | |
|---|---|
| x | A weibull_dist object. |
| ... | Additional arguments (not used). |

### Value

A function `function(n = 1, ...)` returning a numeric vector of length `n`.

### Examples

```
x <- weibull_dist(shape = 2, scale = 3)
s <- sampler(x)
set.seed(42)
s(5)
```

---

`sample_mvn_region` *Function for obtaining sample points for an* `mvn` *object that is within the* `p`*-probability region. That is, it samples from the smallest region of the distribution that contains* `p` *probability mass. This is done by first sampling from the entire distribution, then rejecting samples that are not in the probability region (using the statistical distance* `mahalanobis` *from* `mu`*).*

---

### Description

Function for obtaining sample points for an `mvn` object that is within the `p`-probability region. That is, it samples from the smallest region of the distribution that contains `p` probability mass. This is done by first sampling from the entire distribution, then rejecting samples that are not in the probability region (using the statistical distance `mahalanobis` from `mu`).

### Usage

```
sample_mvn_region(n, mu, sigma, p = 0.95, ...)
```

### Arguments

| | |
|---|---|
| `n` | the sample size |
| `mu` | mean vector |
| `sigma` | variance-covariance matrix |
| `p` | the probability region |
| `...` | additional arguments to pass into `mahalanobis` |

### Value

An `n` by `length(mu)` matrix of samples within the probability region.

### Examples

```
set.seed(42)
pts <- sample_mvn_region(10, mu = c(0, 0), sigma = diag(2), p = 0.95)
dim(pts)
```

## simplify *Generic method for simplifying distributions.*

### Description

Generic method for simplifying distributions.

### Usage

```
simplify(x, ...)
```

### Arguments

| | |
|---|---|
| x | The distribution to simplify |
| ... | Additional arguments to pass |

### Value

The simplified distribution

### Examples

```
# Simplify dispatches to the appropriate method
simplify(normal(0, 1))  # unchanged (already simplified)
```

## simplify.dist *Default Method for simplifying a* dist *object. Just returns the object.*

### Description

Default Method for simplifying a dist object. Just returns the object.

### Usage

```
## S3 method for class 'dist'
simplify(x, ...)
```

### Arguments

| | |
|---|---|
| x | The dist object to simplify |
| ... | Additional arguments to pass (not used) |

### Value

The dist object

### Examples

```
x <- normal(0, 1)
identical(simplify(x), x)  # TRUE, returns unchanged
```

---

`simplify.edist` *Method for simplifying an* `edist` *object.*

---

### Description

Attempts to reduce expression distributions to closed-form distributions when mathematical identities apply. Supported rules include:

### Usage

```
## S3 method for class 'edist'
simplify(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | The `edist` object to simplify |
| `...` | Additional arguments to pass (not used) |

### Details

Single-variable:

- c * Normal(mu, v) -> Normal(c*mu, c*^2v)
- c * Gamma(a, r) -> Gamma(a, r/c) for c > 0
- c * Exponential(r) -> Gamma(1, r/c) for c > 0
- c * Uniform(a, b) -> Uniform(min(ca,cb), max(ca,cb)) for c != 0
- c * Weibull(k, lam) -> Weibull(k, c*lam) for c > 0
- c * ChiSq(df) -> Gamma(df/2, 1/(2c)) for c > 0
- c * LogNormal(ml, sl) -> LogNormal(ml + log(c), sl) for c > 0
- Normal(mu, v) + c -> Normal(mu + c, v)
- Normal(mu, v) - c -> Normal(mu - c, v)
- Uniform(a, b) + c -> Uniform(a + c, b + c)
- Uniform(a, b) - c -> Uniform(a - c, b - c)
- Normal(0, 1) ^ 2 -> ChiSquared(1)
- exp(Normal(mu, v)) -> LogNormal(mu, sqrt(v))
- log(LogNormal(ml, sl)) -> Normal(ml, sl^2)

Two-variable:

- Normal + Normal -> Normal

- Normal - Normal -> Normal
- Gamma + Gamma (same rate) -> Gamma
- Exponential + Exponential (same rate) -> Gamma(2, rate)
- Gamma + Exponential (same rate) -> Gamma(a+1, rate)
- ChiSquared + ChiSquared -> ChiSquared
- Poisson + Poisson -> Poisson
- LogNormal * LogNormal -> LogNormal

## Value

The simplified distribution, or unchanged `edist` if no rule applies

## Examples

```
# Normal + Normal simplifies to a Normal
z <- normal(0, 1) + normal(2, 3)
is_normal(z)  # TRUE
z             # Normal(mu = 2, var = 4)

# exp(Normal) simplifies to LogNormal
w <- exp(normal(0, 1))
is_lognormal(w)  # TRUE
```

---

`Summary.dist` *Summary group generic for distribution objects.*

---

## Description

Handles sum(), prod(), min(), max() of distributions.

## Usage

```
## S3 method for class 'dist'
Summary(..., na.rm = FALSE)
```

## Arguments

| | |
|---|---|
| `...` | dist objects |
| `na.rm` | ignored |

## Value

A simplified distribution or edist

## Examples

```
# sum() reduces via + operator
z <- sum(normal(0, 1), normal(2, 3))
z  # Normal(mu = 2, var = 4)

# min() of exponentials simplifies
w <- min(exponential(1), exponential(2))
w  # Exponential(rate = 3)
```

---

`summary.dist` *Method for obtaining a summary of a* `dist` *object.*

---

## Description

Method for obtaining a summary of a `dist` object.

## Usage

```
## S3 method for class 'dist'
summary(object, ..., name = NULL, nobs = NULL)
```

## Arguments

| | |
|---|---|
| `object` | The object to obtain the summary of |
| `...` | Additional arguments to pass |
| `name` | The name of the distribution, defaults to the class of the object. |
| `nobs` | The number of observations to report for the summary, if applicable. |

## Value

A `summary_dist` object

## Examples

```
summary(normal(0, 1))
```

## summary_dist *Method for constructing a* summary_dist *object.*

### Description

Method for constructing a summary_dist object.

### Usage

```
summary_dist(name, mean, vcov, nobs = NULL)
```

### Arguments

| | |
|---|---|
| name | The name of the distribution |
| mean | The mean of the distribution |
| vcov | The variance of the distribution |
| nobs | The number of observations used to construct the distribution, if applicable. |

### Value

A summary_dist object

### Examples

```
s <- summary_dist(name = "my_dist", mean = 0, vcov = 1)
print(s)
```

## sup *Generic method for retrieving the support of a (dist) object* x.

### Description

The returned value should have the following operations:

- min: a vector, the minimum value of the support for each component.
- max: a vector, the maximum value of the support for each component.
- call: a predicate function, which returns TRUE if the value is in the support, and FALSE otherwise.
- sample: a function, which returns a sample from the support. Note that the returned value is not guaranteed to be in the support of x. You may need to call call to check.

### Usage

```
sup(x)
```

### Arguments

| | |
|---|---|
| x | The object to obtain the support of. |

### Value

A support object for x.

### Examples

```
x <- normal(0, 1)
S <- sup(x)
infimum(S)   # -Inf
supremum(S)  # Inf

y <- exponential(1)
S2 <- sup(y)
infimum(S2)  # 0
```

---

## sup.beta_dist *Support of a beta distribution.*

---

### Description

The beta distribution is supported on the open interval $(0, 1)$.

### Usage

```
## S3 method for class 'beta_dist'
sup(x)
```

### Arguments

| | |
|---|---|
| x | A beta_dist object. |

### Value

An interval object representing $(0, 1)$.

### Examples

```
sup(beta_dist(2, 5))
```

## sup.chi_squared *Support for chi-squared distribution, the positive real numbers (0, Inf).*

### Description

Support for chi-squared distribution, the positive real numbers (0, Inf).

### Usage

```
## S3 method for class 'chi_squared'
sup(x)
```

### Arguments

| | |
|---|---|
| x | The chi_squared object |

### Value

An interval object representing (0, Inf)

### Examples

```
sup(chi_squared(5))
```

## sup.edist *Support for expression distributions.*

### Description

Falls back to realize and delegates to sup.empirical_dist.

### Usage

```
## S3 method for class 'edist'
sup(x)
```

### Arguments

| | |
|---|---|
| x | An edist object. |

### Value

A finite_set support object.

## Examples

```
set.seed(1)
z <- normal(0, 1) * exponential(1)
sup(z)
```

---

`sup.empirical_dist` *Method for obtaining the support of* `empirical_dist` *object* `x`.

---

## Description

Method for obtaining the support of `empirical_dist` object `x`.

## Usage

```
## S3 method for class 'empirical_dist'
sup(x)
```

## Arguments

| | |
|---|---|
| `x` | The empirical distribution object. |

## Value

A `finite_set` object containing the support of `x`.

## Examples

```
ed <- empirical_dist(c(1, 2, 2, 3))
s <- sup(ed)
s$has(2) # TRUE
s$has(4) # FALSE
```

---

`sup.exponential` *Support for exponential distribution, the positive real numbers, (0, Inf).*

---

## Description

Support for exponential distribution, the positive real numbers, (0, Inf).

## Usage

```
## S3 method for class 'exponential'
sup(x)
```

### Arguments

| | |
|---|---|
| x | The object to obtain the support of |

### Value

An `interval` object representing the support of the exponential

### Examples

```
x <- exponential(rate = 1)
sup(x)
```

---

## sup.gamma_dist *Support for gamma distribution, the positive real numbers (0, Inf).*

---

### Description

Support for gamma distribution, the positive real numbers (0, Inf).

### Usage

```
## S3 method for class 'gamma_dist'
sup(x)
```

### Arguments

| | |
|---|---|
| x | The `gamma_dist` object |

### Value

An `interval` object representing (0, Inf)

### Examples

```
sup(gamma_dist(2, 1))
```

## sup.lognormal *Support of a log-normal distribution.*

### Description

The log-normal distribution is supported on $(0, \infty)$.

### Usage

```
## S3 method for class 'lognormal'
sup(x)
```

### Arguments

| | |
|---|---|
| x | A lognormal object. |

### Value

An interval object representing $(0, \infty)$.

### Examples

```
sup(lognormal(0, 1))
```

## sup.mixture *Support of a mixture distribution.*

### Description

Returns an interval spanning the widest range of all component supports (from the smallest infimum to the largest supremum).

### Usage

```
## S3 method for class 'mixture'
sup(x)
```

### Arguments

| | |
|---|---|
| x | A mixture object. |

### Value

An interval object.

### Examples

```
m <- mixture(list(normal(0, 1), exponential(1)), c(0.5, 0.5))
sup(m)
```

## sup.mvn — *Method for obtaining the support of a* mvn *object, where the support is defined as values that have non-zero probability density.*

### Description

Method for obtaining the support of a mvn object, where the support is defined as values that have non-zero probability density.

### Usage

```
## S3 method for class 'mvn'
sup(x, ...)
```

### Arguments

| | |
|---|---|
| x | The mvn object to obtain the support of |
| ... | Additional arguments to pass (not used) |

### Value

A support-type object (see support.R), in this case an interval object for each component.

### Examples

```
X <- mvn(c(0, 0))
sup(X)
```

## sup.normal — *Method for obtaining the support of a* normal *object, where the support is defined as values that have non-zero probability density.*

### Description

Method for obtaining the support of a normal object, where the support is defined as values that have non-zero probability density.

### Usage

```
## S3 method for class 'normal'
sup(x)
```

### Arguments

| | |
|---|---|
| x | The normal object to obtain the support of |

### Value

A support-type object (see `support.R`), in this case an `interval` object for each component.

### Examples

```
x <- normal(0, 1)
sup(x)
```

---

## sup.poisson_dist *Support of a Poisson distribution.*

---

### Description

The Poisson distribution is supported on the non-negative integers $\{0, 1, 2, \ldots\}$.

### Usage

```
## S3 method for class 'poisson_dist'
sup(x)
```

### Arguments

| | |
|---|---|
| `x` | A `poisson_dist` object. |

### Value

A `countable_set` object with lower bound 0.

### Examples

```
sup(poisson_dist(5))
```

---

## sup.uniform_dist *Support of a uniform distribution.*

---

### Description

The uniform distribution is supported on $[min, max]$.

### Usage

```
## S3 method for class 'uniform_dist'
sup(x)
```

### Arguments

| | |
|---|---|
| `x` | A `uniform_dist` object. |

### Value

An `interval` object representing $[min, max]$.

### Examples

```
sup(uniform_dist(0, 10))
```

---

## sup.weibull_dist *Support of a Weibull distribution.*

---

### Description

The Weibull distribution is supported on $(0, \infty)$.

### Usage

```
## S3 method for class 'weibull_dist'
sup(x)
```

### Arguments

| | |
|---|---|
| `x` | A `weibull_dist` object. |

### Value

An `interval` object representing $(0, \infty)$.

### Examples

```
sup(weibull_dist(2, 3))
```

---

## supremum *Get the supremum of the support.*

---

### Description

Get the supremum of the support.

### Usage

```
supremum(object)
```

### Arguments

| | |
|---|---|
| `object` | A support object. |

## Value

The supremum (least upper bound) of the support.

## Examples

```
I <- interval$new(0, 10)
supremum(I)  # 10

S <- finite_set$new(c(3, 7, 11))
supremum(S)  # 11
```

---

`supremum.countable_set` *Get the supremum of a countable set.*

---

## Description

Get the supremum of a countable set.

## Usage

```
## S3 method for class 'countable_set'
supremum(object)
```

## Arguments

| | |
|---|---|
| `object` | A `countable_set` object. |

## Value

`Inf` (the set is unbounded above).

## Examples

```
cs <- countable_set$new(0L)
supremum(cs)  # Inf
```

## supremum.finite_set *Return the supremum of the finite set.*

### Description

Return the supremum of the finite set.

### Usage

```
## S3 method for class 'finite_set'
supremum(object)
```

### Arguments

| | |
|---|---|
| object | A finite set. |

### Value

Numeric; the maximum value(s).

### Examples

```
fs <- finite_set$new(c(1, 3, 5, 7))
supremum(fs) # 7
```

## supremum.interval *Return the (vector of) supremum of the interval.*

### Description

Return the (vector of) supremum of the interval.

### Usage

```
## S3 method for class 'interval'
supremum(object)
```

### Arguments

| | |
|---|---|
| object | An interval object. |

### Value

Numeric vector of upper bounds.

### Examples

```
iv <- interval$new(lower = 0, upper = 1)
supremum(iv) # 1
```

## surv.chi_squared *Method for obtaining the survival function of a* chi_squared *object.*

### Description

Method for obtaining the survival function of a chi_squared object.

### Usage

```
## S3 method for class 'chi_squared'
surv(x, ...)
```

### Arguments

| | |
|---|---|
| x | The chi_squared object |
| ... | Additional arguments (not used) |

### Value

A function that computes S(t) = P(X > t)

### Examples

```
x <- chi_squared(5)
S <- surv(x)
S(5)
```

## surv.continuous_dist *Default survival function for continuous distributions.*

### Description

Computes $S(t) = 1 - F(t)$ from the CDF.

### Usage

```
## S3 method for class 'continuous_dist'
surv(x, ...)

surv(x, ...)
```

### Arguments

| | |
|---|---|
| x | The object to obtain the survival function of. |
| ... | Additional arguments to pass. |

## Value

A function `function(t, ...)` returning the survival probability.

A function computing the survival function $S(t) = P(X > t)$.

## Examples

```
x <- beta_dist(2, 5)
S <- surv(x)
S(0.5)
x <- exponential(1)
S <- surv(x)
S(0)  # 1 (survival at time 0)
S(1)  # exp(-1), approximately 0.368
```

---

# surv.exponential *Method to obtain the survival function of an* `exponential` *object.*

---

## Description

Method to obtain the survival function of an `exponential` object.

## Usage

```
## S3 method for class 'exponential'
surv(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | The object to obtain the survival function of |
| `...` | Additional arguments (not used) |

## Value

A function `function(t, log.p = FALSE, ...)` that computes the survival function $S(t) = P(X > t)$.

## Examples

```
x <- exponential(rate = 1)
S <- surv(x)
S(1)
S(2)
```

## surv.gamma_dist *Method for obtaining the survival function of a* gamma_dist *object.*

### Description

Method for obtaining the survival function of a gamma_dist object.

### Usage

```
## S3 method for class 'gamma_dist'
surv(x, ...)
```

### Arguments

| | |
|---|---|
| x | The gamma_dist object |
| ... | Additional arguments (not used) |

### Value

A function that computes S(t) = P(X > t)

### Examples

```
x <- gamma_dist(shape = 2, rate = 1)
S <- surv(x)
S(1)
```

## surv.lognormal *Survival function for a log-normal distribution.*

### Description

Returns a function that computes $S(t) = P(X > t)$ for the log-normal distribution.

### Usage

```
## S3 method for class 'lognormal'
surv(x, ...)
```

### Arguments

| | |
|---|---|
| x | A lognormal object. |
| ... | Additional arguments (not used). |

## Value

A function `function(t, log.p = FALSE, ...)` returning the survival probability (or log-survival probability) at `t`.

## Examples

```
x <- lognormal(0, 1)
S <- surv(x)
S(1)
S(2)
```

---

`surv.weibull_dist` *Survival function for a Weibull distribution.*

---

## Description

Returns a function that computes $S(t) = P(X > t)$ for the Weibull distribution.

## Usage

```
## S3 method for class 'weibull_dist'
surv(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A `weibull_dist` object. |
| `...` | Additional arguments (not used). |

## Value

A function `function(t, log.p = FALSE, ...)` returning the survival probability (or log-survival probability) at `t`.

## Examples

```
x <- weibull_dist(shape = 2, scale = 3)
S <- surv(x)
S(1)
S(3)
```

## uniform_dist *Construct a uniform distribution object.*

### Description

Creates an S3 object representing a continuous uniform distribution on the interval $[min, max]$. The PDF is $f(x) = 1/(max - min)$ for $min \leq x \leq max$.

### Usage

```
uniform_dist(min = 0, max = 1)
```

### Arguments

| | |
|---|---|
| min | Lower bound of the distribution (default 0). |
| max | Upper bound of the distribution (default 1). |

### Value

A uniform_dist object with classes c("uniform_dist", "univariate_dist", "continuous_dist", "dist").

### Examples

```
x <- uniform_dist(min = 0, max = 10)
mean(x)
vcov(x)
format(x)
```

## vcov.beta_dist *Variance of a beta distribution.*

### Description

Computes $\alpha\beta/((\alpha + \beta)^2(\alpha + \beta + 1))$.

### Usage

```
## S3 method for class 'beta_dist'
vcov(object, ...)
```

### Arguments

| | |
|---|---|
| object | A beta_dist object. |
| ... | Additional arguments (not used). |

## Value

The variance (scalar).

## Examples

```
vcov(beta_dist(2, 5))
```

---

`vcov.chi_squared` *Retrieve the variance of a* `chi_squared` *object.*

---

## Description

Retrieve the variance of a `chi_squared` object.

## Usage

```
## S3 method for class 'chi_squared'
vcov(object, ...)
```

## Arguments

| | |
|---|---|
| `object` | The `chi_squared` object |
| `...` | Additional arguments (not used) |

## Value

The variance, `2 * df`

## Examples

```
vcov(chi_squared(10))
```

---

`vcov.default` *Variance-covariance for non-dist objects (degenerate distributions).*

---

## Description

Variance-covariance for non-dist objects (degenerate distributions).

## Usage

```
## Default S3 method:
vcov(object, ...)
```

### Arguments

| | |
|---|---|
| object | The object (returns 0 for constants) |
| ... | Additional arguments to pass (not used) |

### Value

0 (degenerate distributions have no variance)

### Examples

```
vcov(42)  # returns 0
```

---

## vcov.edist *Method for obtaining the variance-covariance matrix (or scalar)*

---

### Description

Method for obtaining the variance-covariance matrix (or scalar)

### Usage

```
## S3 method for class 'edist'
vcov(object, n = 10000, ...)
```

### Arguments

| | |
|---|---|
| object | The edist object to retrieve the variance-covariance matrix from |
| n | The number of samples to take (default: 10000) |
| ... | Additional arguments to pass (not used) |

### Value

The variance-covariance matrix of the edist object

### Examples

```
set.seed(1)
z <- normal(0, 1) * exponential(2)
vcov(z)
```

## vcov.empirical_dist *Method for obtaining the variance of* empirical_dist *object* x.

### Description

Method for obtaining the variance of empirical_dist object x.

### Usage

```
## S3 method for class 'empirical_dist'
vcov(object, ...)
```

### Arguments

| | |
|---|---|
| object | The empirical distribution object. |
| ... | Additional arguments to pass (not used). |

### Value

The sample variance-covariance matrix.

### Examples

```
ed <- empirical_dist(c(1, 2, 3, 4, 5))
vcov(ed) # sample variance

ed_mv <- empirical_dist(matrix(rnorm(20), ncol = 2))
vcov(ed_mv) # 2x2 covariance matrix
```

## vcov.exponential *Retrieve the variance of a* exponential *object.*

### Description

Retrieve the variance of a exponential object.

### Usage

```
## S3 method for class 'exponential'
vcov(object, ...)
```

### Arguments

| | |
|---|---|
| object | The exponential object to retrieve the variance for |
| ... | Additional arguments to pass (not used) |

### Value

The variance-covariance matrix of the `normal` object

### Examples

```
x <- exponential(rate = 2)
vcov(x)
```

---

## vcov.gamma_dist *Retrieve the variance of a* `gamma_dist` *object.*

---

### Description

Retrieve the variance of a `gamma_dist` object.

### Usage

```
## S3 method for class 'gamma_dist'
vcov(object, ...)
```

### Arguments

| | |
|---|---|
| `object` | The `gamma_dist` object |
| `...` | Additional arguments (not used) |

### Value

The variance, `shape / rate^2`

### Examples

```
vcov(gamma_dist(shape = 3, rate = 2))
```

---

## vcov.lognormal *Variance of a log-normal distribution.*

---

### Description

Computes $(\exp(sdlog^2) - 1)\exp(2 \cdot meanlog + sdlog^2)$.

### Usage

```
## S3 method for class 'lognormal'
vcov(object, ...)
```

### Arguments

| | |
|---|---|
| object | A lognormal object. |
| ... | Additional arguments (not used). |

### Value

The variance (scalar).

### Examples

```
vcov(lognormal(0, 1))
```

---

## vcov.mixture *Variance of a mixture distribution.*

---

### Description

Uses the law of total variance: $Var(X) = E[Var(X|K)] + Var(E[X|K])$.

### Usage

```
## S3 method for class 'mixture'
vcov(object, ...)
```

### Arguments

| | |
|---|---|
| object | A mixture object. |
| ... | Additional arguments (not used). |

### Value

The variance (scalar for univariate mixtures).

### Examples

```
m <- mixture(list(normal(0, 1), normal(10, 1)), c(0.5, 0.5))
vcov(m)
```

## vcov.mvn *Retrieve the variance-covariance matrix of an* mvn *object.*

### Description

Retrieve the variance-covariance matrix of an mvn object.

### Usage

```
## S3 method for class 'mvn'
vcov(object, ...)
```

### Arguments

| | |
|---|---|
| object | The mvn object to retrieve the variance-covariance matrix of |
| ... | Additional arguments to pass (not used) |

### Value

The variance-covariance matrix of the mvn object

### Examples

```
X <- mvn(c(0, 0), diag(2))
vcov(X)
```

## vcov.normal *Retrieve the variance-covariance matrix (or scalar) of a* normal *object.*

### Description

Retrieve the variance-covariance matrix (or scalar) of a normal object.

### Usage

```
## S3 method for class 'normal'
vcov(object, ...)
```

### Arguments

| | |
|---|---|
| object | The normal object to retrieve the variance-covariance matrix from |
| ... | Additional arguments to pass (not used) |

### Value

The variance-covariance matrix of the `normal` object

### Examples

```
x <- normal(0, 4)
vcov(x)
```

---

## vcov.poisson_dist *Variance of a Poisson distribution.*

---

### Description

For the Poisson distribution, the variance equals `lambda`.

### Usage

```
## S3 method for class 'poisson_dist'
vcov(object, ...)
```

### Arguments

| | |
|---|---|
| `object` | A `poisson_dist` object. |
| `...` | Additional arguments (not used). |

### Value

The variance (scalar), equal to `lambda`.

### Examples

```
vcov(poisson_dist(5))
```

---

## vcov.uniform_dist *Variance of a uniform distribution.*

---

### Description

Computes $(max - min)^2/12$.

### Usage

```
## S3 method for class 'uniform_dist'
vcov(object, ...)
```

### Arguments

| | |
|---|---|
| object | A uniform_dist object. |
| ... | Additional arguments (not used). |

### Value

The variance (scalar).

### Examples

```
vcov(uniform_dist(0, 10))
```

---

## vcov.univariate_dist *Method for obtaining the variance of* univariate_dist *object.*

---

### Description

Method for obtaining the variance of univariate_dist object.

### Usage

```
## S3 method for class 'univariate_dist'
vcov(object, ...)
```

### Arguments

| | |
|---|---|
| object | The distribution object. |
| ... | Additional arguments to pass into expectation. |

### Value

Numeric scalar; the variance of the distribution.

### Examples

```
vcov(normal(0, 4))    # 4
vcov(exponential(2))  # 0.25
```

## vcov.weibull_dist *Variance of a Weibull distribution.*

### Description

Computes $scale^2(\Gamma(1 + 2/shape) - [\Gamma(1 + 1/shape)]^2)$.

### Usage

```
## S3 method for class 'weibull_dist'
vcov(object, ...)
```

### Arguments

| | |
|---|---|
| object | A weibull_dist object. |
| ... | Additional arguments (not used). |

### Value

The variance (scalar).

### Examples

```
vcov(weibull_dist(shape = 2, scale = 3))
```

## weibull_dist *Construct a Weibull distribution object.*

### Description

Creates an S3 object representing a Weibull distribution with the given shape and scale parameters. The Weibull PDF is

$$f(t) = (shape/scale)(t/scale)^{shape-1} \exp(-(t/scale)^{shape})$$

for $t > 0$.

### Usage

```
weibull_dist(shape, scale)
```

### Arguments

| | |
|---|---|
| shape | Positive scalar shape parameter. |
| scale | Positive scalar scale parameter. |

## Value

A `weibull_dist` object with classes `c("weibull_dist", "univariate_dist", "continuous_dist", "dist")`.

## Examples

```
x <- weibull_dist(shape = 2, scale = 3)
mean(x)
vcov(x)
format(x)
```

---

`^.dist` *Power operator for distribution objects.*

---

## Description

Power operator for distribution objects.

## Usage

```
## S3 method for class 'dist'
x ^ y
```

## Arguments

| | |
|---|---|
| `x` | a dist object (base) |
| `y` | a numeric scalar (exponent) |

## Value

A simplified distribution or edist

## Examples

```
# Standard normal squared yields chi-squared(1)
z <- normal(0, 1)^2
z
```

# Index


*.dist, 7
+.dist, 8
-.dist, 9
/.dist, 9
^.dist, 171

affine_transform, 10
as_dist, 11

beta_dist, 12

cdf, 12, *127*
cdf.beta_dist, 13
cdf.chi_squared, 14
cdf.edist, 14
cdf.empirical_dist, *14*, 15
cdf.exponential, 16
cdf.gamma_dist, 16
cdf.lognormal, 17
cdf.mixture, 18
cdf.mvn, 18
cdf.normal, 19
cdf.poisson_dist, 20
cdf.uniform_dist, 20
cdf.weibull_dist, 21
chi_squared, 22
clt, 22
conditional, 23, *127*
conditional.dist, 24
conditional.edist, 24
conditional.empirical_dist, *24*, 25
conditional.mixture, 26
conditional.mvn, 27
countable_set, 28

delta_clt, 29
density, *127*
density.beta_dist, 29
density.chi_squared, 30
density.edist, 31
density.empirical_dist, *31*, 31
density.exponential, 32
density.gamma_dist, 33
density.lognormal, 33
density.mixture, 34
density.mvn, 35
density.normal, 35
density.poisson_dist, 36
density.uniform_dist, 37
density.weibull_dist, 37
dim, *28*
dim.beta_dist, 38
dim.chi_squared, 38
dim.countable_set, 39
dim.edist, 40
dim.empirical_dist, 40
dim.exponential, 41
dim.finite_set, 41
dim.gamma_dist, 42
dim.interval, 42
dim.lognormal, 43
dim.mixture, 43
dim.mvn, 44
dim.normal, 45
dim.poisson_dist, 45
dim.uniform_dist, 46
dim.weibull_dist, 46

edist, 47
empirical_dist, *14*, 47, *127*
ensure_realized, *27*
expectation, 48
expectation.dist, 49
expectation.empirical_dist, 50
expectation.poisson_dist, 50
expectation.univariate_dist, 51
expectation_data, 52
exponential, 53

finite_set, 54

format.beta_dist, 55
format.chi_squared, 56
format.edist, 56
format.empirical_dist, 57
format.exponential, 57
format.gamma_dist, 58
format.lognormal, 58
format.mixture, 59
format.mvn, 60
format.normal, 60
format.poisson_dist, 61
format.realized_dist, 61
format.uniform_dist, 62
format.weibull_dist, 62

gamma_dist, 63

has, *28*, 64
has.countable_set, 65
has.finite_set, 65
has.interval, 66
hazard *(*hazard.continuous_dist*)*, 67
hazard.chi_squared, 67
hazard.continuous_dist, 67
hazard.exponential, 68
hazard.gamma_dist, 69
hazard.lognormal, 69
hazard.weibull_dist, 70

infimum, *28*, 71
infimum.countable_set, 71
infimum.finite_set, 72
infimum.interval, 72
interval, 73, *151*
inv_cdf, 75
inv_cdf.beta_dist, 75
inv_cdf.chi_squared, 76
inv_cdf.edist, 77
inv_cdf.empirical_dist, *77*, 77
inv_cdf.exponential, 78
inv_cdf.gamma_dist, 79
inv_cdf.lognormal, 79
inv_cdf.normal, 80
inv_cdf.poisson_dist, 81
inv_cdf.uniform_dist, 81
inv_cdf.weibull_dist, 82
is_beta_dist, 83
is_chi_squared, 83
is_dist, 84
is_edist, 84
is_empirical_dist, 85
is_exponential, 85
is_gamma_dist, 86
is_lognormal, 86
is_mixture, 87
is_mvn, 87
is_normal, 88
is_poisson_dist, 88
is_realized_dist, 89
is_uniform_dist, 89
is_weibull_dist, 90

lln, 90
lognormal, 91

marginal, 92, *93*
marginal.empirical_dist, 92
marginal.mixture, 93
marginal.mvn, 94
Math.dist, 94
mean.beta_dist, 95
mean.chi_squared, 96
mean.edist, 96
mean.empirical_dist, 97
mean.exponential, 97
mean.gamma_dist, 98
mean.lognormal, 99
mean.mixture, 99
mean.mvn, 100
mean.normal, 100
mean.poisson_dist, 101
mean.uniform_dist, 101
mean.univariate_dist, 102
mean.weibull_dist, 103
mixture, 103
mvn, 104

nobs.empirical_dist, 105
normal, 105
normal_approx, 106
nparams, 106
nparams.empirical_dist, 107
nparams.mixture, 108

obs, 108
obs.empirical_dist, 109

params, 109

params.beta_dist, 110
params.chi_squared, 111
params.edist, 111
params.empirical_dist, 112
params.exponential, 112
params.gamma_dist, 113
params.lognormal, 113
params.mixture, 114
params.mvn, 114
params.normal, 115
params.poisson_dist, 115
params.uniform_dist, 116
params.weibull_dist, 117
poisson_dist, 117
print.beta_dist, 118
print.chi_squared, 118
print.edist, 119
print.empirical_dist, 119
print.exponential, 120
print.gamma_dist, 120
print.interval, 121
print.lognormal, 122
print.mixture, 122
print.mvn, 123
print.normal, 123
print.poisson_dist, 124
print.realized_dist, 124
print.summary_dist, 125
print.uniform_dist, 126
print.weibull_dist, 126

realize, *14*, *24*, *31*, *77*, 127, *129*, *148*
realized_dist, *127*
rmap, 128
rmap.dist, 129
rmap.edist, 129
rmap.empirical_dist, *129*, 130
rmap.mvn, 131

sample_mvn_region, 141
sampler, 131
sampler.beta_dist, 132
sampler.chi_squared, 133
sampler.default, 133
sampler.edist, 134
sampler.empirical_dist, 135
sampler.exponential, 135
sampler.gamma_dist, 136
sampler.lognormal, 137
sampler.mixture, 137
sampler.mvn, 138
sampler.normal, 138
sampler.poisson_dist, 139
sampler.uniform_dist, 140
sampler.weibull_dist, 140
simplify, 142
simplify.dist, 142
simplify.edist, 143
Summary.dist, 144
summary.dist, 145
summary_dist, 146
sup, 146
sup.beta_dist, 147
sup.chi_squared, 148
sup.edist, 148
sup.empirical_dist, *148*, 149
sup.exponential, 149
sup.gamma_dist, 150
sup.lognormal, 151
sup.mixture, 151
sup.mvn, 152
sup.normal, 152
sup.poisson_dist, 153
sup.uniform_dist, 153
sup.weibull_dist, 154
supremum, *28*, 154
supremum.countable_set, 155
supremum.finite_set, 156
supremum.interval, 156
surv *(*surv.continuous_dist*)*, 157
surv.chi_squared, 157
surv.continuous_dist, 157
surv.exponential, 158
surv.gamma_dist, 159
surv.lognormal, 159
surv.weibull_dist, 160

uniform_dist, 161

vcov.beta_dist, 161
vcov.chi_squared, 162
vcov.default, 162
vcov.edist, 163
vcov.empirical_dist, 164
vcov.exponential, 164
vcov.gamma_dist, 165
vcov.lognormal, 165
vcov.mixture, 166

vcov.mvn, 167
vcov.normal, 167
vcov.poisson_dist, 168
vcov.uniform_dist, 168
vcov.univariate_dist, 169
vcov.weibull_dist, 170

weibull_dist, 170