Package ‘joker’

April 28, 2025

Title Probability Distributions and Parameter Estimation

Version 0.14.2

Description Implements an S4 distribution system and estimation methods for parameters of common distribution families. The common d, p, q, r function family for each distribution is enriched with the ll, e, and v counterparts, computing the log-likelihood, performing estimation, and calculating the asymptotic variance - covariance matrix, respectively. Parameter estimation is performed analytically whenever possible.

URL https://thechibo.github.io/joker/

BugReports https://github.com/thechibo/joker/issues/

Depends R (>= 4.0.0)

Imports ggh4x, ggplot2, grDevices, Matrix, methods, stats, utils

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

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

RoxygenNote 7.3.2

NeedsCompilation no

Author Ioannis Oikonomidis [aut, cre] (<https://orcid.org/0000-0001-8130-2104>),
Samis Trevezas [aut, ths] (<https://orcid.org/0000-0003-2262-8299>)

Maintainer Ioannis Oikonomidis <goikon@math.uoa.gr>

Repository CRAN

Date/Publication 2025-04-28 12:40:01 UTC

Contents

Bern 2
Beta 6
Binom 11
calculus 14
Cat 16
Cauchy 19
Chisq 23
Dir 27
distributions 31
estimation 34
Exp 37
Fisher 41
functionals 44
Gam 46
Geom 51
idigamma 54
Laplace 55
LargeMetrics 59
Lnorm 61
loglikelihood 64
moments 67
Multigam 70
Multinom 73
Nbinom 77
Norm 80
plot 84
Pois 86
SmallMetrics 90
Stud 92
Unif 95
variance 98
Weib 101

Index **105**

Bern *Bern Distribution*

Description

The Bernoulli distribution is a discrete probability distribution which takes the value 1 with probability p and the value 0 with probability $1 - p$, where $0 \leq p \leq 1$.

Usage

```
Bern(prob = 0.5)

dbern(x, prob, log = FALSE)

pbern(q, prob, lower.tail = TRUE, log.p = FALSE)

qbern(p, prob, lower.tail = TRUE, log.p = FALSE)

rbern(n, prob)

## S4 method for signature 'Bern,numeric'
d(distr, x, log = FALSE)

## S4 method for signature 'Bern,numeric'
p(distr, q, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Bern,numeric'
qn(distr, p, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Bern,numeric'
r(distr, n)

## S4 method for signature 'Bern'
mean(x)

## S4 method for signature 'Bern'
median(x)

## S4 method for signature 'Bern'
mode(x)

## S4 method for signature 'Bern'
var(x)

## S4 method for signature 'Bern'
sd(x)

## S4 method for signature 'Bern'
skew(x)

## S4 method for signature 'Bern'
kurt(x)

## S4 method for signature 'Bern'
entro(x)

## S4 method for signature 'Bern'
```

```
finf(x)

llbern(x, prob)

## S4 method for signature 'Bern,numeric'
ll(distr, x)

ebern(x, type = "mle", ...)

## S4 method for signature 'Bern,numeric'
mle(distr, x, na.rm = FALSE)

## S4 method for signature 'Bern,numeric'
me(distr, x, na.rm = FALSE)

vbern(prob, type = "mle")

## S4 method for signature 'Bern'
avar_mle(distr)

## S4 method for signature 'Bern'
avar_me(distr)
```

Arguments

`prob`	numeric. Probability of success.
`x`	For the density function, `x` is a numeric vector of quantiles. For the moments functions, `x` is an object of class `Bern`. For the log-likelihood and the estimation functions, `x` is the sample of observations.
`log`, `log.p`	logical. Should the logarithm of the probability be returned?
`q`	numeric. Vector of quantiles.
`lower.tail`	logical. If TRUE (default), probabilities are $P(X \leq x)$, otherwise $P(X > x)$.
`p`	numeric. Vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`distr`	an object of class `Bern`.
`type`	character, case ignored. The estimator type (mle or me).
`...`	extra arguments.
`na.rm`	logical. Should the `NA` values be removed?

Details

The probability mass function (PMF) of the Bernoulli distribution is given by:

$$f(x; p) = p^x (1-p)^{1-x}, \quad p \in (0, 1), \quad x \in \{0, 1\}.$$

Value

Each type of function returns a different type of object:

- Distribution Functions: When supplied with one argument (`distr`), the `d()`, `p()`, `q()`, `r()`, `ll()` functions return the density, cumulative probability, quantile, random sample generator, and log-likelihood functions, respectively. When supplied with both arguments (`distr` and `x`), they evaluate the aforementioned functions directly.
- Moments: Returns a numeric, either vector or matrix depending on the moment and the distribution. The `moments()` function returns a list with all the available methods.
- Estimation: Returns a list, the estimators of the unknown parameters. Note that in distribution families like the binomial, multinomial, and negative binomial, the size is not returned, since it is considered known.
- Variance: Returns a named matrix. The asymptotic covariance matrix of the estimator.

See Also

Functions from the `stats` package: `dbinom()`, `pbinom()`, `qbinom()`, `rbinom()`

Examples

```
# -----------------------------------------------------
# Bernoulli Distribution Example
# -----------------------------------------------------

# Create the distribution
p <- 0.7
D <- Bern(p)

# ------------------
# dpqr Functions
# ------------------

d(D, c(0, 1)) # density function
p(D, c(0, 1)) # distribution function
qn(D, c(0.4, 0.8)) # inverse distribution function
x <- r(D, 100) # random generator function

# alternative way to use the function
df <- d(D) ; df(x) # df is a function itself

# ------------------
# Moments
# ------------------

mean(D) # Expectation
median(D) # Median
mode(D) # Mode
var(D) # Variance
sd(D) # Standard Deviation
skew(D) # Skewness
```

```
kurt(D) # Excess Kurtosis
entro(D) # Entropy
finf(D) # Fisher Information Matrix

# List of all available moments
mom <- moments(D)
mom$mean # expectation

# ------------------
# Point Estimation
# ------------------

ll(D, x)
llbern(x, p)

ebern(x, type = "mle")
ebern(x, type = "me")

mle(D, x)
me(D, x)
e(D, x, type = "mle")

mle("bern", x) # the distr argument can be a character

# ------------------
# Estimator Variance
# ------------------

vbern(p, type = "mle")
vbern(p, type = "me")

avar_mle(D)
avar_me(D)

v(D, type = "mle")
```

Beta *Beta Distribution*

Description

The Beta distribution is an absolute continuous probability distribution with support $S = [0, 1]$, parameterized by two shape parameters, $\alpha > 0$ and $\beta > 0$.

Usage

```
Beta(shape1 = 1, shape2 = 1)

## S4 method for signature 'Beta,numeric'
d(distr, x, log = FALSE)
```

```
## S4 method for signature 'Beta,numeric'
p(distr, q, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Beta,numeric'
qn(distr, p, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Beta,numeric'
r(distr, n)

## S4 method for signature 'Beta'
mean(x)

## S4 method for signature 'Beta'
median(x)

## S4 method for signature 'Beta'
mode(x)

## S4 method for signature 'Beta'
var(x)

## S4 method for signature 'Beta'
sd(x)

## S4 method for signature 'Beta'
skew(x)

## S4 method for signature 'Beta'
kurt(x)

## S4 method for signature 'Beta'
entro(x)

## S4 method for signature 'Beta'
finf(x)

llbeta(x, shape1, shape2)

## S4 method for signature 'Beta,numeric'
ll(distr, x)

ebeta(x, type = "mle", ...)

## S4 method for signature 'Beta,numeric'
mle(
  distr,
  x,
```

```
  par0 = "same",
  method = "L-BFGS-B",
  lower = 1e-05,
  upper = Inf,
  na.rm = FALSE
)

## S4 method for signature 'Beta,numeric'
me(distr, x, na.rm = FALSE)

## S4 method for signature 'Beta,numeric'
same(distr, x, na.rm = FALSE)

vbeta(shape1, shape2, type = "mle")

## S4 method for signature 'Beta'
avar_mle(distr)

## S4 method for signature 'Beta'
avar_me(distr)

## S4 method for signature 'Beta'
avar_same(distr)
```

Arguments

shape1, shape2	numeric. The non-negative distribution parameters.
distr	an object of class Beta.
x	For the density function, x is a numeric vector of quantiles. For the moments functions, x is an object of class Beta. For the log-likelihood and the estimation functions, x is the sample of observations.
log, log.p	logical. Should the logarithm of the probability be returned?
q	numeric. Vector of quantiles.
lower.tail	logical. If TRUE (default), probabilities are $P(X \leq x)$, otherwise $P(X > x)$.
p	numeric. Vector of probabilities.
n	number of observations. If length(n) > 1, the length is taken to be the number required.
type	character, case ignored. The estimator type (mle, me, or same).
...	extra arguments.
par0, method, lower, upper	arguments passed to optim for the mle optimization. See Details.
na.rm	logical. Should the NA values be removed?

Details

The probability density function (PDF) of the Beta distribution is given by:

$$f(x; \alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)}, \quad \alpha \in \mathbb{R}_+, \, \beta \in \mathbb{R}_+,$$

for $x \in S = [0, 1]$, where $B(\alpha, \beta)$ is the Beta function:

$$B(\alpha, \beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1} dt.$$

The MLE of the beta distribution parameters is not available in closed form and has to be approximated numerically. This is done with `optim()`. Specifically, instead of solving a bivariate optimization problem w.r.t (α, β), the optimization can be performed on the parameter sum $\alpha_0 := \alpha + \beta \in (0, +\infty)$. The default method used is the L-BFGS-B method with lower bound `1e-5` and upper bound `Inf`. The `par0` argument can either be a numeric (satisfying `lower <= par0 <= upper`) or a character specifying the closed-form estimator to be used as initialization for the algorithm (`"me"` or `"same"` - the default value).

Value

Each type of function returns a different type of object:

- Distribution Functions: When supplied with one argument (`distr`), the `d()`, `p()`, `q()`, `r()`, `ll()` functions return the density, cumulative probability, quantile, random sample generator, and log-likelihood functions, respectively. When supplied with both arguments (`distr` and `x`), they evaluate the aforementioned functions directly.
- Moments: Returns a numeric, either vector or matrix depending on the moment and the distribution. The `moments()` function returns a list with all the available methods.
- Estimation: Returns a list, the estimators of the unknown parameters. Note that in distribution families like the binomial, multinomial, and negative binomial, the size is not returned, since it is considered known.
- Variance: Returns a named matrix. The asymptotic covariance matrix of the estimator.

References

- Tamae, H., Irie, K. & Kubokawa, T. (2020), A score-adjusted approach to closed-form estimators for the gamma and beta distributions, Japanese Journal of Statistics and Data Science 3, 543–561.
- Papadatos, N. (2022), On point estimators for gamma and beta distributions, arXiv preprint arXiv:2205.10799.

See Also

Functions from the `stats` package: `dbeta()`, `pbeta()`, `qbeta()`, `rbeta()`

Examples

```
# -----------------------------------------------------
# Beta Distribution Example
# -----------------------------------------------------

# Create the distribution
a <- 3
b <- 5
D <- Beta(a, b)

# ------------------
# dpqr Functions
# ------------------

d(D, c(0.3, 0.8, 0.5)) # density function
p(D, c(0.3, 0.8, 0.5)) # distribution function
qn(D, c(0.4, 0.8)) # inverse distribution function
x <- r(D, 100) # random generator function

# alternative way to use the function
df <- d(D) ; df(x) # df is a function itself

# ------------------
# Moments
# ------------------

mean(D) # Expectation
var(D) # Variance
sd(D) # Standard Deviation
skew(D) # Skewness
kurt(D) # Excess Kurtosis
entro(D) # Entropy
finf(D) # Fisher Information Matrix

# List of all available moments
mom <- moments(D)
mom$mean # expectation

# ------------------
# Point Estimation
# ------------------

ll(D, x)
llbeta(x, a, b)

ebeta(x, type = "mle")
ebeta(x, type = "me")
ebeta(x, type = "same")

mle(D, x)
me(D, x)
same(D, x)
```

```
e(D, x, type = "mle")

mle("beta", x) # the distr argument can be a character

# ------------------
# Estimator Variance
# ------------------

vbeta(a, b, type = "mle")
vbeta(a, b, type = "me")
vbeta(a, b, type = "same")

avar_mle(D)
avar_me(D)
avar_same(D)

v(D, type = "mle")
```

Binom *Binom Distribution*

Description

The binomial distribution is a discrete probability distribution which models the probability of having x successes in n independent Bernoulli trials with success probability p.

Usage

```
Binom(size = 1, prob = 0.5)

## S4 method for signature 'Binom,numeric'
d(distr, x, log = FALSE)

## S4 method for signature 'Binom,numeric'
p(distr, q, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Binom,numeric'
qn(distr, p, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Binom,numeric'
r(distr, n)

## S4 method for signature 'Binom'
mean(x)

## S4 method for signature 'Binom'
var(x)
```

```
## S4 method for signature 'Binom'
sd(x)

## S4 method for signature 'Binom'
skew(x)

## S4 method for signature 'Binom'
kurt(x)

## S4 method for signature 'Binom'
entro(x)

## S4 method for signature 'Binom'
finf(x)

llbinom(x, size, prob)

## S4 method for signature 'Binom,numeric'
ll(distr, x)

ebinom(x, size, type = "mle", ...)

## S4 method for signature 'Binom,numeric'
mle(distr, x, na.rm = FALSE)

## S4 method for signature 'Binom,numeric'
me(distr, x, na.rm = FALSE)

vbinom(size, prob, type = "mle")

## S4 method for signature 'Binom'
avar_mle(distr)

## S4 method for signature 'Binom'
avar_me(distr)
```

Arguments

size	number of trials (zero or more).
prob	numeric. Probability of success on each trial.
distr	an object of class Binom.
x	For the density function, x is a numeric vector of quantiles. For the moments functions, x is an object of class Binom. For the log-likelihood and the estimation functions, x is the sample of observations.
log, log.p	logical. Should the logarithm of the probability be returned?
q	numeric. Vector of quantiles.
lower.tail	logical. If TRUE (default), probabilities are $P(X \leq x)$, otherwise $P(X > x)$.

p	numeric. Vector of probabilities.
n	number of observations. If `length(n)` > 1, the length is taken to be the number required.
type	character, case ignored. The estimator type (mle or me).
...	extra arguments.
na.rm	logical. Should the `NA` values be removed?

Details

The probability mass function (PMF) of the binomial distribution is given by:

$$f(x; n, p) = \binom{n}{x} p^x (1-p)^{n-x}, \quad N \in \mathbb{N}, \quad p \in (0, 1),$$

with $x \in \{0, 1, \dots, N\}$.

Value

Each type of function returns a different type of object:

- Distribution Functions: When supplied with one argument (`distr`), the `d()`, `p()`, `q()`, `r()`, `ll()` functions return the density, cumulative probability, quantile, random sample generator, and log-likelihood functions, respectively. When supplied with both arguments (`distr` and `x`), they evaluate the aforementioned functions directly.
- Moments: Returns a numeric, either vector or matrix depending on the moment and the distribution. The `moments()` function returns a list with all the available methods.
- Estimation: Returns a list, the estimators of the unknown parameters. Note that in distribution families like the binomial, multinomial, and negative binomial, the size is not returned, since it is considered known.
- Variance: Returns a named matrix. The asymptotic covariance matrix of the estimator.

See Also

Functions from the `stats` package: `dbinom()`, `pbinom()`, `qbinom()`, `rbinom()`

Examples

```
# -----------------------------------------------------
# Binomial Distribution Example
# -----------------------------------------------------

# Create the distribution
N <- 10 ; p <- 0.7
D <- Binom(N, p)

# ------------------
# dpqr Functions
# ------------------
```

```
d(D, 0:N) # density function
p(D, 0:N) # distribution function
qn(D, c(0.4, 0.8)) # inverse distribution function
x <- r(D, 100) # random generator function

# alternative way to use the function
df <- d(D) ; df(x) # df is a function itself

# ------------------
# Moments
# ------------------

mean(D) # Expectation
var(D) # Variance
sd(D) # Standard Deviation
skew(D) # Skewness
kurt(D) # Excess Kurtosis
entro(D) # Entropy
finf(D) # Fisher Information Matrix

# List of all available moments
mom <- moments(D)
mom$mean # expectation

# ------------------
# Point Estimation
# ------------------

ll(D, x)
llbinom(x, N, p)

ebinom(x, size = N, type = "mle")
ebinom(x, size = N, type = "me")

mle(D, x)
me(D, x)
e(D, x, type = "mle")

# ------------------
# Estimator Variance
# ------------------

vbinom(N, p, type = "mle")
vbinom(N, p, type = "me")

avar_mle(D)
avar_me(D)

v(D, type = "mle")
```

calculus *Distributional Calculus*

Description

Arithmetic operators and functions for probability distribution objects. These methods define how standard operations like `+`, `-`, `*`, and `/` behave when applied to random variables, returning the resulting distribution based on known properties of common distribution families.

Usage

```
## S4 method for signature 'Norm,Norm'
e1 + e2

## S4 method for signature 'numeric,Norm'
e1 + e2

## S4 method for signature 'Norm,numeric'
e1 + e2

## S4 method for signature 'Norm,Norm'
e1 - e2

## S4 method for signature 'numeric,Norm'
e1 - e2

## S4 method for signature 'Norm,numeric'
e1 - e2

## S4 method for signature 'numeric,Norm'
e1 * e2

## S4 method for signature 'Norm,numeric'
e1 * e2

## S4 method for signature 'Norm,numeric'
e1 / e2

## S4 method for signature 'Norm,logical'
sum(x, ..., na.rm = FALSE)

## S4 method for signature 'Norm'
exp(x)
```

Arguments

`x, e1, e2`	objects of subclass `Distribution`.
`...`	extra arguments.
`na.rm`	logical. Should missing values be removed?

Value

All calculations return `Distribution` objects (specifically, objects of a class that is a subclass of `Distribution`), according to the property at hand.

Examples

```
# -----------------------------------------------------
# Distribution Calculus Example
# -----------------------------------------------------

# Normal location - scale transformation
x <- Norm(2, 3)
y <- 3 * x + 1 # Norm(7, 9)

# Addition of two independent Normal random variables
x1 <- Norm(1, 3)
x2 <- Norm(2, 4)
x3 <- x1 + x2 # Norm(3, 5)
```

`Cat` *Cat Distribution*

Description

The Categorical distribution is a discrete probability distribution that describes the probability of a single trial resulting in one of k possible categories. It is a generalization of the Bernoulli distribution and a special case of the multinomial distribution with $n = 1$.

Usage

```
Cat(prob = c(0.5, 0.5))

dcat(x, prob, log = FALSE)

rcat(n, prob)

## S4 method for signature 'Cat,numeric'
d(distr, x, log = FALSE)

## S4 method for signature 'Cat,numeric'
r(distr, n)

## S4 method for signature 'Cat'
mean(x)

## S4 method for signature 'Cat'
mode(x)
```

```
## S4 method for signature 'Cat'
var(x)

## S4 method for signature 'Cat'
entro(x)

## S4 method for signature 'Cat'
finf(x)

llcat(x, prob)

## S4 method for signature 'Cat,numeric'
ll(distr, x)

ecat(x, type = "mle", ...)

## S4 method for signature 'Cat,numeric'
mle(distr, x, dim = NULL, na.rm = FALSE)

## S4 method for signature 'Cat,numeric'
me(distr, x, dim = NULL, na.rm = FALSE)

vcat(prob, type = "mle")

## S4 method for signature 'Cat'
avar_mle(distr)

## S4 method for signature 'Cat'
avar_me(distr)
```

Arguments

prob	numeric. Probability vector of success for each category.
x	For the density function, x is a numeric vector of quantiles. For the moments functions, x is an object of class Cat. For the log-likelihood and the estimation functions, x is the sample of observations.
log	logical. Should the logarithm of the probability be returned?
n	number of observations. If length(n) > 1, the length is taken to be the number required.
distr	an object of class Cat.
type	character, case ignored. The estimator type (mle or me).
...	extra arguments.
dim	numeric. The probability vector dimension. See Details.
na.rm	logical. Should the NA values be removed?

Details

The probability mass function (PMF) of the categorical distribution is given by:

$$f(x;p) = \prod_{i=1}^{k} p_i^{x_i},$$

subject to $\sum_{i=1}^{k} x_i = n$.

The estimation of `prob` from a sample would by default return a vector of probabilities corresponding to the categories that appeared in the sample and 0 for the rest. However, the parameter dimension cannot be uncovered by the sample, it has to be provided separately. This can be done with the argument `dim`. If `dim` is not supplied, the dimension will be retrieved from the `distr` argument. Categories that did not appear in the sample will have 0 probabilities appended to the end of the prob vector.

Note that the actual dimension of the probability parameter vector is `k-1`, therefore the Fisher information matrix and the asymptotic variance - covariance matrix of the estimators is of dimension `(k-1)x(k-1)`.

Value

Each type of function returns a different type of object:

- Distribution Functions: When supplied with one argument (`distr`), the `d()`, `p()`, `q()`, `r()`, `ll()` functions return the density, cumulative probability, quantile, random sample generator, and log-likelihood functions, respectively. When supplied with both arguments (`distr` and `x`), they evaluate the aforementioned functions directly.
- Moments: Returns a numeric, either vector or matrix depending on the moment and the distribution. The `moments()` function returns a list with all the available methods.
- Estimation: Returns a list, the estimators of the unknown parameters. Note that in distribution families like the binomial, multinomial, and negative binomial, the size is not returned, since it is considered known.
- Variance: Returns a named matrix. The asymptotic covariance matrix of the estimator.

See Also

`dmultinom()`, `rmultinom()`

Examples

```
# -----------------------------------------------------
# Categorical Distribution Example
# -----------------------------------------------------

# Create the distribution
p <- c(0.1, 0.2, 0.7)
D <- Cat(p)

# ------------------
# dpqr Functions
```

```
# ------------------

d(D, 2) # density function
x <- r(D, 100) # random generator function

# alternative way to use the function
df <- d(D) ; df(x) # df is a function itself

# ------------------
# Moments
# ------------------

mean(D) # Expectation
mode(D) # Mode
var(D) # Variance
entro(D) # Entropy
finf(D) # Fisher Information Matrix

# List of all available moments
mom <- moments(D)
mom$mean # expectation

# ------------------
# Point Estimation
# ------------------

ll(D, x)
llcat(x, p)

ecat(x, dim = 3, type = "mle")
ecat(x, dim = 3, type = "me")

mle(D, x)
me(D, x)
e(D, x, type = "mle")

mle("cat", dim = 3, x) # the distr argument can be a character

# ------------------
# Estimator Variance
# ------------------

vcat(p, type = "mle")
vcat(p, type = "me")

avar_mle(D)
avar_me(D)

v(D, type = "mle")
```

Cauchy *Cauchy Distribution*

Description

The Cauchy distribution is an absolute continuous probability distribution characterized by its location parameter x_0 and scale parameter $\gamma > 0$.

Usage

```
Cauchy(location = 0, scale = 1)

## S4 method for signature 'Cauchy,numeric'
d(distr, x, log = FALSE)

## S4 method for signature 'Cauchy,numeric'
p(distr, q, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Cauchy,numeric'
qn(distr, p, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Cauchy,numeric'
r(distr, n)

## S4 method for signature 'Cauchy'
mean(x)

## S4 method for signature 'Cauchy'
median(x)

## S4 method for signature 'Cauchy'
mode(x)

## S4 method for signature 'Cauchy'
var(x)

## S4 method for signature 'Cauchy'
sd(x)

## S4 method for signature 'Cauchy'
skew(x)

## S4 method for signature 'Cauchy'
kurt(x)

## S4 method for signature 'Cauchy'
entro(x)

## S4 method for signature 'Cauchy'
finf(x)

llcauchy(x, location, scale)
```

```
## S4 method for signature 'Cauchy,numeric'
ll(distr, x)

ecauchy(x, type = "mle", ...)

## S4 method for signature 'Cauchy,numeric'
mle(
  distr,
  x,
  par0 = "me",
  method = "L-BFGS-B",
  lower = c(-Inf, 1e-05),
  upper = c(Inf, Inf),
  na.rm = FALSE
)

## S4 method for signature 'Cauchy,numeric'
me(distr, x, na.rm = FALSE)

vcauchy(location, scale, type = "mle")

## S4 method for signature 'Cauchy'
avar_mle(distr)
```

Arguments

`location, scale`	numeric. Location and scale parameters.
`distr`	an object of class `Cauchy`.
`x`	For the density function, `x` is a numeric vector of quantiles. For the moments functions, `x` is an object of class `Cauchy`. For the log-likelihood and the estimation functions, `x` is the sample of observations.
`log, log.p`	logical. Should the logarithm of the probability be returned?
`q`	numeric. Vector of quantiles.
`lower.tail`	logical. If TRUE (default), probabilities are $P(X \leq x)$, otherwise $P(X > x)$.
`p`	numeric. Vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`type`	character, case ignored. The estimator type (mle or me).
`...`	extra arguments.
`par0, method, lower, upper`	arguments passed to optim for the mle optimization.
`na.rm`	logical. Should the `NA` values be removed?

Details

The probability density function (PDF) of the Cauchy distribution is given by:

$$f(x; x_0, \gamma) = \frac{1}{\pi\gamma \left[1 + \left(\frac{x-x_0}{\gamma}\right)^2\right]}.$$

The MLE of the Cauchy distribution parameters is not available in closed form and has to be approximated numerically. This is done with `optim()`. The default method used is the L-BFGS-B method with lower bounds `c(-Inf, 1e-5)` and upper bounds `c(Inf, Inf)`. The `par0` argument can either be a numeric (both elements satisfying `lower <= par0 <= upper`) or a character specifying the closed-form estimator to be used as initialization for the algorithm (`"me"` - the default value).

Note that the `me()` estimator for the Cauchy distribution is not a *moment* estimator; it utilizes the sample median instead of the sample mean.

Value

Each type of function returns a different type of object:

- Distribution Functions: When supplied with one argument (`distr`), the `d()`, `p()`, `q()`, `r()`, `ll()` functions return the density, cumulative probability, quantile, random sample generator, and log-likelihood functions, respectively. When supplied with both arguments (`distr` and `x`), they evaluate the aforementioned functions directly.
- Moments: Returns a numeric, either vector or matrix depending on the moment and the distribution. The `moments()` function returns a list with all the available methods.
- Estimation: Returns a list, the estimators of the unknown parameters. Note that in distribution families like the binomial, multinomial, and negative binomial, the size is not returned, since it is considered known.
- Variance: Returns a named matrix. The asymptotic covariance matrix of the estimator.

See Also

Functions from the `stats` package: `dcauchy()`, `pcauchy()`, `qcauchy()`, `rcauchy()`

Examples

```
# -----------------------------------------------------
# Cauchy Distribution Example
# -----------------------------------------------------

# Create the distribution
x0 <- 3 ; scale <- 5
D <- Cauchy(x0, scale)

# ------------------
# dpqr Functions
# ------------------

d(D, c(-5, 3, 10)) # density function
```

```
p(D, c(-5, 3, 10)) # distribution function
qn(D, c(0.4, 0.8)) # inverse distribution function
x <- r(D, 100) # random generator function

# alternative way to use the function
df <- d(D) ; df(x) # df is a function itself

# ------------------
# Moments
# ------------------

median(D) # Median
mode(D) # Mode
entro(D) # Entropy
finf(D) # Fisher Information Matrix

# ------------------
# Point Estimation
# ------------------

ll(D, x)
llcauchy(x, x0, scale)

ecauchy(x, type = "mle")
ecauchy(x, type = "me")

mle(D, x)
me(D, x)
e(D, x, type = "mle")

mle("cauchy", x) # the distr argument can be a character

# ------------------
# Estimator Variance
# ------------------

vcauchy(x0, scale, type = "mle")
avar_mle(D)
v(D, type = "mle")
```

`Chisq` *Chi-Square Distribution*

Description

The Chi-Square distribution is a continuous probability distribution commonly used in statistical inference, particularly in hypothesis testing and confidence interval estimation. It is defined by the degrees of freedom parameter $k > 0$.

Usage

```
Chisq(df = 1)

## S4 method for signature 'Chisq,numeric'
d(distr, x, log = FALSE)

## S4 method for signature 'Chisq,numeric'
p(distr, q, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Chisq,numeric'
qn(distr, p, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Chisq,numeric'
r(distr, n)

## S4 method for signature 'Chisq'
mean(x)

## S4 method for signature 'Chisq'
median(x)

## S4 method for signature 'Chisq'
mode(x)

## S4 method for signature 'Chisq'
var(x)

## S4 method for signature 'Chisq'
sd(x)

## S4 method for signature 'Chisq'
skew(x)

## S4 method for signature 'Chisq'
kurt(x)

## S4 method for signature 'Chisq'
entro(x)

## S4 method for signature 'Chisq'
finf(x)

llchisq(x, df)

## S4 method for signature 'Chisq,numeric'
ll(distr, x)

echisq(x, type = "mle", ...)
```

```
## S4 method for signature 'Chisq,numeric'
mle(distr, x, na.rm = FALSE)

## S4 method for signature 'Chisq,numeric'
me(distr, x, na.rm = FALSE)

vchisq(df, type = "mle")

## S4 method for signature 'Chisq'
avar_mle(distr)

## S4 method for signature 'Chisq'
avar_me(distr)
```

Arguments

`df`	numeric. The distribution degrees of freedom parameter.
`distr`	an object of class `Chisq`.
`x`	For the density function, `x` is a numeric vector of quantiles. For the moments functions, `x` is an object of class `Chisq`. For the log-likelihood and the estimation functions, `x` is the sample of observations.
`log, log.p`	logical. Should the logarithm of the probability be returned?
`q`	numeric. Vector of quantiles.
`lower.tail`	logical. If TRUE (default), probabilities are $P(X \le x)$, otherwise $P(X > x)$.
`p`	numeric. Vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`type`	character, case ignored. The estimator type (mle or me).
`...`	extra arguments.
`na.rm`	logical. Should the `NA` values be removed?

Details

The probability density function (PDF) of the Chi-Square distribution is given by:

$$f(x;k) = \frac{1}{2^{k/2}\Gamma(k/2)} x^{k/2-1} e^{-x/2}, \quad x > 0.$$

Value

Each type of function returns a different type of object:

- Distribution Functions: When supplied with one argument (`distr`), the `d()`, `p()`, `q()`, `r()`, `ll()` functions return the density, cumulative probability, quantile, random sample generator, and log-likelihood functions, respectively. When supplied with both arguments (`distr` and `x`), they evaluate the aforementioned functions directly.

- Moments: Returns a numeric, either vector or matrix depending on the moment and the distribution. The `moments()` function returns a list with all the available methods.
- Estimation: Returns a list, the estimators of the unknown parameters. Note that in distribution families like the binomial, multinomial, and negative binomial, the size is not returned, since it is considered known.
- Variance: Returns a named matrix. The asymptotic covariance matrix of the estimator.

See Also

Functions from the `stats` package: `dchisq()`, `pchisq()`, `qchisq()`, `rchisq()`

Examples

```
# -----------------------------------------------------
# Chi-Square Distribution Example
# -----------------------------------------------------

# Create the distribution
df <- 4
D <- Chisq(df)

# ------------------
# dpqr Functions
# ------------------

d(D, c(0.3, 2, 20)) # density function
p(D, c(0.3, 2, 20)) # distribution function
qn(D, c(0.4, 0.8)) # inverse distribution function
x <- r(D, 100) # random generator function

# alternative way to use the function
den <- d(D) ; den(x) # den is a function itself

# ------------------
# Moments
# ------------------

mean(D) # Expectation
var(D) # Variance
sd(D) # Standard Deviation
skew(D) # Skewness
kurt(D) # Excess Kurtosis
entro(D) # Entropy
finf(D) # Fisher Information Matrix

# List of all available moments
mom <- moments(D)
mom$mean # expectation

# ------------------
# Point Estimation
# ------------------
```

```
ll(D, x)
llchisq(x, df)

echisq(x, type = "mle")
echisq(x, type = "me")

mle(D, x)
me(D, x)
e(D, x, type = "mle")

mle("chisq", x) # the distr argument can be a character

# ------------------
# Estimator Variance
# ------------------

vchisq(df, type = "mle")
vchisq(df, type = "me")

avar_mle(D)
avar_me(D)

v(D, type = "mle")
```

Dir *Dirichlet Distribution*

Description

The Dirichlet distribution is an absolute continuous probability, specifically a multivariate generalization of the beta distribution, parameterized by a vector $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_k)$ with $\alpha_i > 0$.

Usage

```
Dir(alpha = c(1, 1))

ddir(x, alpha, log = FALSE)

rdir(n, alpha)

## S4 method for signature 'Dir,numeric'
d(distr, x, log = FALSE)

## S4 method for signature 'Dir,matrix'
d(distr, x)

## S4 method for signature 'Dir,numeric'
r(distr, n)
```

```
## S4 method for signature 'Dir'
mean(x)

## S4 method for signature 'Dir'
mode(x)

## S4 method for signature 'Dir'
var(x)

## S4 method for signature 'Dir'
entro(x)

## S4 method for signature 'Dir'
finf(x)

lldir(x, alpha)

## S4 method for signature 'Dir,matrix'
ll(distr, x)

edir(x, type = "mle", ...)

## S4 method for signature 'Dir,matrix'
mle(
  distr,
  x,
  par0 = "same",
  method = "L-BFGS-B",
  lower = 1e-05,
  upper = Inf,
  na.rm = FALSE
)

## S4 method for signature 'Dir,matrix'
me(distr, x, na.rm = FALSE)

## S4 method for signature 'Dir,matrix'
same(distr, x, na.rm = FALSE)

vdir(alpha, type = "mle")

## S4 method for signature 'Dir'
avar_mle(distr)

## S4 method for signature 'Dir'
avar_me(distr)
```

```
## S4 method for signature 'Dir'
avar_same(distr)
```

Arguments

`alpha`	numeric. The non-negative distribution parameter vector.
`x`	For the density function, `x` is a numeric vector of quantiles. For the moments functions, `x` is an object of class `Dir`. For the log-likelihood and the estimation functions, `x` is the sample of observations.
`log`	logical. Should the logarithm of the probability be returned?
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`distr`	an object of class `Dir`.
`type`	character, case ignored. The estimator type (mle, me, or same).
`...`	extra arguments.
`par0`, `method`, `lower`, `upper`	arguments passed to optim for the mle optimization.
`na.rm`	logical. Should the `NA` values be removed?

Details

The probability density function (PDF) of the Dirichlet distribution is given by:

$$f(x_1, ..., x_k; \alpha_1, ..., \alpha_k) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{i=1}^k x_i^{\alpha_i - 1},$$

where $B(\boldsymbol{\alpha})$ is the multivariate Beta function:

$$B(\boldsymbol{\alpha}) = \frac{\prod_{i=1}^k \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^k \alpha_i\right)}$$

and $\sum_{i=1}^k x_i = 1$, $x_i > 0$.

Value

Each type of function returns a different type of object:

- Distribution Functions: When supplied with one argument (`distr`), the `d()`, `p()`, `q()`, `r()`, `ll()` functions return the density, cumulative probability, quantile, random sample generator, and log-likelihood functions, respectively. When supplied with both arguments (`distr` and `x`), they evaluate the aforementioned functions directly.
- Moments: Returns a numeric, either vector or matrix depending on the moment and the distribution. The `moments()` function returns a list with all the available methods.
- Estimation: Returns a list, the estimators of the unknown parameters. Note that in distribution families like the binomial, multinomial, and negative binomial, the size is not returned, since it is considered known.
- Variance: Returns a named matrix. The asymptotic covariance matrix of the estimator.

References

- Oikonomidis, I. & Trevezas, S. (2025), Moment-Type Estimators for the Dirichlet and the Multivariate Gamma Distributions, arXiv, https://arxiv.org/abs/2311.15025

Examples

```
# -----------------------------------------------------
# Dir Distribution Example
# -----------------------------------------------------

# Create the distribution
a <- c(0.5, 2, 5)
D <- Dir(a)

# ------------------
# dpqr Functions
# ------------------

d(D, c(0.3, 0.2, 0.5)) # density function
x <- r(D, 100) # random generator function

# alternative way to use the function
df <- d(D) ; df(x) # df is a function itself

# ------------------
# Moments
# ------------------

mean(D) # Expectation
mode(D) # Mode
var(D) # Variance
entro(D) # Entropy
finf(D) # Fisher Information Matrix

# List of all available moments
mom <- moments(D)
mom$mean # expectation

# ------------------
# Point Estimation
# ------------------

ll(D, x)
lldir(x, a)

edir(x, type = "mle")
edir(x, type = "me")

mle(D, x)
me(D, x)
e(D, x, type = "mle")
```

```
mle("dir", x) # the distr argument can be a character

# ------------------
# Estimator Variance
# ------------------

vdir(a, type = "mle")
vdir(a, type = "me")

avar_mle(D)
avar_me(D)

v(D, type = "mle")
```

---

distributions *Distribution S4 Classes*

Description

A collection of S4 classes that provide a flexible and structured way to work with probability distributions.

Usage

```
d(distr, x, ...)

p(distr, q, ...)

qn(distr, p, ...)

r(distr, n, ...)
```

Arguments

distr	an object of class `Distribution` or one of its subclasses.
x	For the density function, `x` is a numeric vector of quantiles. For the moments functions, `x` is an object of class `Distribution` or one of its subclasses. For the log-likelihood and the estimation functions, `x` is the sample of observations.
...	extra arguments.
q	numeric. Vector of quantiles.
p	numeric. Vector of probabilities.
n	number of observations. If `length(n)` > 1, the length is taken to be the number required.

Details

These S4 generic methods can work both as functions and as functionals (functions that return functions). The available distribution families are coded as S4 classes, specifically subclasses of the `Distribution` superclass. The methods can be used in two ways:

Option 1: If both the `distr` argument and `x` or `n` are supplied, then the function is evaluated directly, as usual.

Option 2: If only the `distr` argument is supplied, the method returns a function that takes as input the missing argument `x` or `n`, allowing the user to work with the function object itself. See examples.

Looking for a specific distribution family? This help page is general. Use the help page of each distribution to see the available methods for the class, details, and examples. Check the See Also section.

Value

Each type of function returns a different type of object:

- Distribution Functions: When supplied with one argument (`distr`), the `d()`, `p()`, `q()`, `r()`, `ll()` functions return the density, cumulative probability, quantile, random sample generator, and log-likelihood functions, respectively. When supplied with both arguments (`distr` and `x`), they evaluate the aforementioned functions directly.
- Moments: Returns a numeric, either vector or matrix depending on the moment and the distribution. The `moments()` function returns a list with all the available methods.
- Estimation: Returns a list, the estimators of the unknown parameters. Note that in distribution families like the binomial, multinomial, and negative binomial, the size is not returned, since it is considered known.
- Variance: Returns a named matrix. The asymptotic covariance matrix of the estimator.

Functions

- `d()`: density function
- `p()`: cumulative distribution function
- `qn()`: generalized inverse distribution function
- `r()`: random sample generator function

See Also

moments, loglikelihood, estimation, Bern, Beta, Binom, Cat, Cauchy, Chisq, Dir, Exp, Fisher, Gam, Geom, Laplace, Lnorm, Multigam, Multinom, Nbinom, Norm, Pois, Stud, Unif, Weib

Examples

```
# -----------------------------------------------------
# Beta Distribution Example
# -----------------------------------------------------

# Create the distribution
a <- 3
```

```
b <- 5
D <- Beta(a, b)

# ------------------
# dpqr Functions
# ------------------

d(D, c(0.3, 0.8, 0.5)) # density function
p(D, c(0.3, 0.8, 0.5)) # distribution function
qn(D, c(0.4, 0.8)) # inverse distribution function
x <- r(D, 100) # random generator function

# alternative way to use the function
df <- d(D) ; df(x) # df is a function itself

# ------------------
# Moments
# ------------------

mean(D) # Expectation
var(D) # Variance
sd(D) # Standard Deviation
skew(D) # Skewness
kurt(D) # Excess Kurtosis
entro(D) # Entropy
finf(D) # Fisher Information Matrix

# List of all available moments
mom <- moments(D)
mom$mean # expectation

# ------------------
# Point Estimation
# ------------------

ll(D, x)
llbeta(x, a, b)

ebeta(x, type = "mle")
ebeta(x, type = "me")
ebeta(x, type = "same")

mle(D, x)
me(D, x)
same(D, x)
e(D, x, type = "mle")

mle("beta", x) # the distr argument can be a character

# ------------------
# Estimator Variance
# ------------------
```

```
vbeta(a, b, type = "mle")
vbeta(a, b, type = "me")
vbeta(a, b, type = "same")

avar_mle(D)
avar_me(D)
avar_same(D)

v(D, type = "mle")
```

`estimation` *Parameter Estimation*

Description

This set of functions estimates the parameters of a random sample according to a specified family of distributions. See details.

Usage

```
e(distr, x, type = "mle", ...)

mle(distr, x, ...)

## S4 method for signature 'character,ANY'
mle(distr, x, ...)

me(distr, x, ...)

## S4 method for signature 'character,ANY'
me(distr, x, ...)

same(distr, x, ...)

## S4 method for signature 'character,ANY'
same(distr, x, ...)
```

Arguments

`distr`	A `Distribution` object or a `character`. The distribution family assumed.
`x`	numeric. A sample under estimation.
`type`	character, case ignored. The estimator type.
`...`	extra arguments.

Details

The package covers three major estimation methods: maximum likelihood estimation (MLE), moment estimation (ME), and score-adjusted estimation (SAME).

In order to perform parameter estimation, a new `e<name>()` member is added to the `d()`, `p()`, `q()`, `r()` family, following the standard `stats` name convention. These functions take two arguments, the observations `x` (an atomic vector for univariate or a matrix for multivariate distributions) and the `type` of estimation method to use (a character with possible values `"mle"`, `"me"`, and `"same"`.)

Point estimation functions are available in two versions, the distribution specific one, e.g. `ebeta()`, and the S4 generic ones, namely `mle()`, `me()`, and `same()`. A general function called `e()` is also implemented, covering all distributions and estimators.

Value

list. The estimator of the unknown parameters. Note that in distribution families like the binomial, multinomial, and negative binomial, the size is not returned, since it is considered known.

Functions

- `mle()`: Maximum Likelihood Estimator
- `me()`: Moment Estimator
- `same()`: Score - Adjusted Moment Estimation

References

General Textbooks

- Van der Vaart, A. W. (2000), Asymptotic statistics, Vol. 3, Cambridge university press.

Beta and gamma distribution families

- Ye, Z.-S. & Chen, N. (2017), Closed-form estimators for the gamma distribution derived from likelihood equations, The American Statistician 71(2), 177–181.
- Tamae, H., Irie, K. & Kubokawa, T. (2020), A score-adjusted approach to closed-form estimators for the gamma and beta distributions, Japanese Journal of Statistics and Data Science 3, 543–561.
- Mathal, A. & Moschopoulos, P. (1992), A form of multivariate gamma distribution, Annals of the Institute of Statistical Mathematics 44, 97–106.
- Oikonomidis, I. & Trevezas, S. (2023), Moment-Type Estimators for the Dirichlet and the Multivariate Gamma Distributions, arXiv, https://arxiv.org/abs/2311.15025

See Also

mle, me, same

Examples

```
# -----------------------------------------------------
# Beta Distribution Example
# -----------------------------------------------------

# Create the distribution
a <- 3
b <- 5
D <- Beta(a, b)

# ------------------
# dpqr Functions
# ------------------

d(D, c(0.3, 0.8, 0.5)) # density function
p(D, c(0.3, 0.8, 0.5)) # distribution function
qn(D, c(0.4, 0.8)) # inverse distribution function
x <- r(D, 100) # random generator function

# alternative way to use the function
df <- d(D) ; df(x) # df is a function itself

# ------------------
# Moments
# ------------------

mean(D) # Expectation
var(D) # Variance
sd(D) # Standard Deviation
skew(D) # Skewness
kurt(D) # Excess Kurtosis
entro(D) # Entropy
finf(D) # Fisher Information Matrix

# List of all available moments
mom <- moments(D)
mom$mean # expectation

# ------------------
# Point Estimation
# ------------------

ll(D, x)
llbeta(x, a, b)

ebeta(x, type = "mle")
ebeta(x, type = "me")
ebeta(x, type = "same")

mle(D, x)
me(D, x)
same(D, x)
```

```
e(D, x, type = "mle")

mle("beta", x) # the distr argument can be a character

# ------------------
# Estimator Variance
# ------------------

vbeta(a, b, type = "mle")
vbeta(a, b, type = "me")
vbeta(a, b, type = "same")

avar_mle(D)
avar_me(D)
avar_same(D)

v(D, type = "mle")
```

Exp *Exponential Distribution*

Description

The Exponential distribution is a continuous probability distribution often used to model the time between independent events that occur at a constant average rate. It is defined by the rate parameter $\lambda > 0$.

Usage

```
Exp(rate = 1)

## S4 method for signature 'Exp,numeric'
d(distr, x, log = FALSE)

## S4 method for signature 'Exp,numeric'
p(distr, q, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Exp,numeric'
qn(distr, p, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Exp,numeric'
r(distr, n)

## S4 method for signature 'Exp'
mean(x)

## S4 method for signature 'Exp'
median(x)
```

```
## S4 method for signature 'Exp'
mode(x)

## S4 method for signature 'Exp'
var(x)

## S4 method for signature 'Exp'
sd(x)

## S4 method for signature 'Exp'
skew(x)

## S4 method for signature 'Exp'
kurt(x)

## S4 method for signature 'Exp'
entro(x)

## S4 method for signature 'Exp'
finf(x)

llexp(x, rate)

## S4 method for signature 'Exp,numeric'
ll(distr, x)

eexp(x, type = "mle", ...)

## S4 method for signature 'Exp,numeric'
mle(distr, x, na.rm = FALSE)

## S4 method for signature 'Exp,numeric'
me(distr, x, na.rm = FALSE)

vexp(rate, type = "mle")

## S4 method for signature 'Exp'
avar_mle(distr)

## S4 method for signature 'Exp'
avar_me(distr)
```

Arguments

rate	numeric. The distribution parameter.
distr	an object of class Exp.
x	For the density function, x is a numeric vector of quantiles. For the moments

	functions, x is an object of class Exp. For the log-likelihood and the estimation functions, x is the sample of observations.
log, log.p	logical. Should the logarithm of the probability be returned?
q	numeric. Vector of quantiles.
lower.tail	logical. If TRUE (default), probabilities are $P(X \leq x)$, otherwise $P(X > x)$.
p	numeric. Vector of probabilities.
n	number of observations. If length(n) > 1, the length is taken to be the number required.
type	character, case ignored. The estimator type (mle or me).
...	extra arguments.
na.rm	logical. Should the NA values be removed?

Details

The probability density function (PDF) of the Exponential distribution is given by:

$$f(x; \lambda) = \lambda e^{-\lambda x}, \quad x \geq 0.$$

Value

Each type of function returns a different type of object:

- Distribution Functions: When supplied with one argument (distr), the d(), p(), q(), r(), ll() functions return the density, cumulative probability, quantile, random sample generator, and log-likelihood functions, respectively. When supplied with both arguments (distr and x), they evaluate the aforementioned functions directly.
- Moments: Returns a numeric, either vector or matrix depending on the moment and the distribution. The moments() function returns a list with all the available methods.
- Estimation: Returns a list, the estimators of the unknown parameters. Note that in distribution families like the binomial, multinomial, and negative binomial, the size is not returned, since it is considered known.
- Variance: Returns a named matrix. The asymptotic covariance matrix of the estimator.

See Also

Functions from the stats package: dexp(), pexp(), qexp(), rexp()

Examples

```
# -----------------------------------------------------
# Exp Distribution Example
# -----------------------------------------------------

# Create the distribution
rate <- 5
D <- Exp(rate)
```

```
# ------------------
# dpqr Functions
# ------------------

d(D, c(0.3, 2, 10)) # density function
p(D, c(0.3, 2, 10)) # distribution function
qn(D, c(0.4, 0.8)) # inverse distribution function
x <- r(D, 100) # random generator function

# alternative way to use the function
df <- d(D) ; df(x) # df is a function itself

# ------------------
# Moments
# ------------------

mean(D) # Expectation
median(D) # Median
mode(D) # Mode
var(D) # Variance
sd(D) # Standard Deviation
skew(D) # Skewness
kurt(D) # Excess Kurtosis
entro(D) # Entropy
finf(D) # Fisher Information Matrix

# List of all available moments
mom <- moments(D)
mom$mean # expectation

# ------------------
# Point Estimation
# ------------------

ll(D, x)
llexp(x, rate)

eexp(x, type = "mle")
eexp(x, type = "me")

mle(D, x)
me(D, x)
e(D, x, type = "mle")

mle("exp", x) # the distr argument can be a character

# ------------------
# Estimator Variance
# ------------------

vexp(rate, type = "mle")
vexp(rate, type = "me")
```

```
avar_mle(D)
avar_me(D)

v(D, type = "mle")
```

Fisher *Fisher Distribution*

Description

The Fisher (F) distribution is an absolute continuous probability distribution that arises frequently in the analysis of variance (ANOVA) and in hypothesis testing. It is defined by two degrees of freedom parameters $d_1 > 0$ and $d_2 > 0$.

Usage

```
Fisher(df1 = 1, df2 = 1)

## S4 method for signature 'Fisher,numeric'
d(distr, x, log = FALSE)

## S4 method for signature 'Fisher,numeric'
p(distr, q, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Fisher,numeric'
qn(distr, p, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Fisher,numeric'
r(distr, n)

## S4 method for signature 'Fisher'
mean(x)

## S4 method for signature 'Fisher'
median(x)

## S4 method for signature 'Fisher'
mode(x)

## S4 method for signature 'Fisher'
var(x)

## S4 method for signature 'Fisher'
sd(x)

## S4 method for signature 'Fisher'
skew(x)
```

```
## S4 method for signature 'Fisher'
kurt(x)

## S4 method for signature 'Fisher'
entro(x)

llf(x, df1, df2)

## S4 method for signature 'Fisher,numeric'
ll(distr, x)
```

Arguments

`df1, df2`	numeric. The distribution degrees of freedom parameters.
`distr`	an object of class `Fisher`.
`x`	For the density function, `x` is a numeric vector of quantiles. For the moments functions, `x` is an object of class `Fisher`. For the log-likelihood functions, `x` is the sample of observations.
`log, log.p`	logical. Should the logarithm of the probability be returned?
`q`	numeric. Vector of quantiles.
`lower.tail`	logical. If TRUE (default), probabilities are $P(X \leq x)$, otherwise $P(X > x)$.
`p`	numeric. Vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.

Details

The probability density function (PDF) of the F-distribution is given by:

$$f(x; d_1, d_2) = \frac{\sqrt{\left(\frac{d_1 x}{d_1 x + d_2}\right)^{d_1} \left(\frac{d_2}{d_1 x + d_2}\right)^{d_2}}}{x B(d_1/2, d_2/2)}, \quad x > 0.$$

Value

Each type of function returns a different type of object:

- Distribution Functions: When supplied with one argument (`distr`), the `d()`, `p()`, `q()`, `r()`, `ll()` functions return the density, cumulative probability, quantile, random sample generator, and log-likelihood functions, respectively. When supplied with both arguments (`distr` and `x`), they evaluate the aforementioned functions directly.
- Moments: Returns a numeric, either vector or matrix depending on the moment and the distribution. The `moments()` function returns a list with all the available methods.
- Estimation: Returns a list, the estimators of the unknown parameters. Note that in distribution families like the binomial, multinomial, and negative binomial, the size is not returned, since it is considered known.
- Variance: Returns a named matrix. The asymptotic covariance matrix of the estimator.

See Also

Functions from the `stats` package: `df()`, `pf()`, `qf()`, `rf()`

Examples

```
# -----------------------------------------------------
# Fisher Distribution Example
# -----------------------------------------------------

# Create the distribution
df1 <- 14 ; df2 <- 20
D <- Fisher(df1, df2)

# ------------------
# dpqr Functions
# ------------------

d(D, c(0.3, 2, 10)) # density function
p(D, c(0.3, 2, 10)) # distribution function
qn(D, c(0.4, 0.8)) # inverse distribution function
x <- r(D, 100) # random generator function

# alternative way to use the function
df <- d(D) ; df(x) # df is a function itself

# ------------------
# Moments
# ------------------

mean(D) # Expectation
median(D) # Median
mode(D) # Mode
var(D) # Variance
sd(D) # Standard Deviation
skew(D) # Skewness
kurt(D) # Excess Kurtosis
entro(D) # Entropy

# List of all available moments
mom <- moments(D)
mom$mean # expectation

# ------------------
# Point Estimation
# ------------------

ll(D, x)
llf(x, df1, df2)
```

functionals *Distribution Functionals*

Description

A collection of S4 classes that provide a flexible and structured way to work with probability distributions.

Usage

```
## S4 method for signature 'Distribution,missing'
d(distr, x, ...)

## S4 method for signature 'Distribution,missing'
p(distr, q, ...)

## S4 method for signature 'Distribution,missing'
qn(distr, p, ...)

## S4 method for signature 'Distribution,missing'
r(distr, n, ...)

## S4 method for signature 'Distribution,missing'
ll(distr, x, ...)

## S4 method for signature 'Distribution,missing'
mle(distr, x, ...)

## S4 method for signature 'Distribution,missing'
me(distr, x, ...)

## S4 method for signature 'Distribution,missing'
same(distr, x, ...)
```

Arguments

`distr`	a `Distribution` object.
`x, q, p, n`	missing. Arguments not supplied.
`...`	extra arguments.

Details

When `x`, `q`, `p`, or `n` are missing, the methods return a function that takes as input the missing argument, allowing the user to work with the function object itself. See examples.

Value

When supplied with one argument, the `d()`, `p()`, `q()`, `r()` `ll()` functions return the density, cumulative probability, quantile, random sample generator, and log-likelihood functions, respectively.

See Also

moments, loglikelihood, estimation, Bern, Beta, Binom, Cat, Cauchy, Chisq, Dir, Exp, Fisher, Gam, Geom, Laplace, Lnorm, Multigam, Multinom, Nbinom, Norm, Pois, Stud, Unif, Weib

Examples

```
# -----------------------------------------------------
# Beta Distribution Example
# -----------------------------------------------------

# Create the distribution
a <- 3
b <- 5
D <- Beta(a, b)

# ------------------
# dpqr Functions
# ------------------

d(D, c(0.3, 0.8, 0.5)) # density function
p(D, c(0.3, 0.8, 0.5)) # distribution function
qn(D, c(0.4, 0.8)) # inverse distribution function
x <- r(D, 100) # random generator function

# alternative way to use the function
df <- d(D) ; df(x) # df is a function itself

# ------------------
# Moments
# ------------------

mean(D) # Expectation
var(D) # Variance
sd(D) # Standard Deviation
skew(D) # Skewness
kurt(D) # Excess Kurtosis
entro(D) # Entropy
finf(D) # Fisher Information Matrix

# List of all available moments
mom <- moments(D)
mom$mean # expectation

# ------------------
# Point Estimation
# ------------------
```

```
ll(D, x)
llbeta(x, a, b)

ebeta(x, type = "mle")
ebeta(x, type = "me")
ebeta(x, type = "same")

mle(D, x)
me(D, x)
same(D, x)
e(D, x, type = "mle")

mle("beta", x) # the distr argument can be a character

# ------------------
# Estimator Variance
# ------------------

vbeta(a, b, type = "mle")
vbeta(a, b, type = "me")
vbeta(a, b, type = "same")

avar_mle(D)
avar_me(D)
avar_same(D)

v(D, type = "mle")
```

Gam *Gamma Distribution*

Description

The Gamma distribution is an absolute continuous probability distribution with two parameters: shape $\alpha > 0$ and scale $\beta > 0$.

Usage

```
Gam(shape = 1, scale = 1)

## S4 method for signature 'Gam,numeric'
d(distr, x, log = FALSE)

## S4 method for signature 'Gam,numeric'
p(distr, q, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Gam,numeric'
qn(distr, p, lower.tail = TRUE, log.p = FALSE)
```

```
## S4 method for signature 'Gam,numeric'
r(distr, n)

## S4 method for signature 'Gam'
mean(x)

## S4 method for signature 'Gam'
median(x)

## S4 method for signature 'Gam'
mode(x)

## S4 method for signature 'Gam'
var(x)

## S4 method for signature 'Gam'
sd(x)

## S4 method for signature 'Gam'
skew(x)

## S4 method for signature 'Gam'
kurt(x)

## S4 method for signature 'Gam'
entro(x)

## S4 method for signature 'Gam'
finf(x)

llgamma(x, shape, scale)

## S4 method for signature 'Gam,numeric'
ll(distr, x)

egamma(x, type = "mle", ...)

## S4 method for signature 'Gam,numeric'
mle(
  distr,
  x,
  par0 = "same",
  method = "L-BFGS-B",
  lower = 1e-05,
  upper = Inf,
  na.rm = FALSE
)
```

```
## S4 method for signature 'Gam,numeric'
me(distr, x, na.rm = FALSE)

## S4 method for signature 'Gam,numeric'
same(distr, x, na.rm = FALSE)

vgamma(shape, scale, type = "mle")

## S4 method for signature 'Gam'
avar_mle(distr)

## S4 method for signature 'Gam'
avar_me(distr)

## S4 method for signature 'Gam'
avar_same(distr)
```

Arguments

shape, scale	numeric. The non-negative distribution parameters.
distr	an object of class Gam.
x	For the density function, x is a numeric vector of quantiles. For the moments functions, x is an object of class Gam. For the log-likelihood and the estimation functions, x is the sample of observations.
log, log.p	logical. Should the logarithm of the probability be returned?
q	numeric. Vector of quantiles.
lower.tail	logical. If TRUE (default), probabilities are $P(X \leq x)$, otherwise $P(X > x)$.
p	numeric. Vector of probabilities.
n	number of observations. If length(n) > 1, the length is taken to be the number required.
type	character, case ignored. The estimator type (mle, me, or same).
...	extra arguments.
par0, method, lower, upper	arguments passed to optim for the mle optimization. See Details.
na.rm	logical. Should the NA values be removed?

Details

The probability density function (PDF) of the Gamma distribution is given by:

$$f(x; \alpha, \beta) = \frac{\beta^{-\alpha} x^{\alpha-1} e^{-x/\beta}}{\Gamma(\alpha)}, \quad x > 0.$$

The MLE of the gamma distribution parameters is not available in closed form and has to be approximated numerically. This is done with `optim()`. The optimization can be performed on the shape parameter $\alpha \in (0, +\infty)$. The default method used is the L-BFGS-B method with lower bound `1e-5` and upper bound `Inf`. The `par0` argument can either be a numeric (satisfying `lower <= par0 <= upper`) or a character specifying the closed-form estimator to be used as initialization for the algorithm (`"me"` or `"same"` - the default value).

Value

Each type of function returns a different type of object:

- Distribution Functions: When supplied with one argument (`distr`), the `d()`, `p()`, `q()`, `r()`, `ll()` functions return the density, cumulative probability, quantile, random sample generator, and log-likelihood functions, respectively. When supplied with both arguments (`distr` and `x`), they evaluate the aforementioned functions directly.
- Moments: Returns a numeric, either vector or matrix depending on the moment and the distribution. The `moments()` function returns a list with all the available methods.
- Estimation: Returns a list, the estimators of the unknown parameters. Note that in distribution families like the binomial, multinomial, and negative binomial, the size is not returned, since it is considered known.
- Variance: Returns a named matrix. The asymptotic covariance matrix of the estimator.

References

- Wiens, D. P., Cheng, J., & Beaulieu, N. C. (2003). A class of method of moments estimators for the two-parameter gamma family. Pakistan Journal of Statistics, 19(1), 129-141.
- Ye, Z. S., & Chen, N. (2017). Closed-form estimators for the gamma distribution derived from likelihood equations. The American Statistician, 71(2), 177-181.
- Tamae, H., Irie, K. & Kubokawa, T. (2020), A score-adjusted approach to closed-form estimators for the gamma and beta distributions, Japanese Journal of Statistics and Data Science 3, 543–561.
- Papadatos, N. (2022), On point estimators for gamma and beta distributions, arXiv preprint arXiv:2205.10799.

See Also

Functions from the `stats` package: `dgamma()`, `pgamma()`, `qgamma()`, `rgamma()`

Examples

```
# -----------------------------------------------------
# Gamma Distribution Example
# -----------------------------------------------------

# Create the distribution
a <- 3 ; b <- 5
D <- Gam(a, b)

# ------------------
# dpqr Functions
# ------------------
```

```
d(D, c(0.3, 2, 10)) # density function
p(D, c(0.3, 2, 10)) # distribution function
qn(D, c(0.4, 0.8)) # inverse distribution function
x <- r(D, 100) # random generator function

# alternative way to use the function
df <- d(D) ; df(x) # df is a function itself

# ------------------
# Moments
# ------------------

mean(D) # Expectation
median(D) # Median
mode(D) # Mode
var(D) # Variance
sd(D) # Standard Deviation
skew(D) # Skewness
kurt(D) # Excess Kurtosis
entro(D) # Entropy
finf(D) # Fisher Information Matrix

# List of all available moments
mom <- moments(D)
mom$mean # expectation

# ------------------
# Point Estimation
# ------------------

ll(D, x)
llgamma(x, a, b)

egamma(x, type = "mle")
egamma(x, type = "me")
egamma(x, type = "same")

mle(D, x)
me(D, x)
same(D, x)
e(D, x, type = "mle")

mle("gam", x) # the distr argument can be a character

# ------------------
# Estimator Variance
# ------------------

vgamma(a, b, type = "mle")
vgamma(a, b, type = "me")
vgamma(a, b, type = "same")

avar_mle(D)
```

```
avar_me(D)
avar_same(D)

v(D, type = "mle")
```

Geom *Geometric Distribution*

Description

The Geometric distribution is a discrete probability distribution that models the number of failures before the first success in a sequence of independent Bernoulli trials, each with success probability $0 < p \leq 1$.

Usage

```
Geom(prob = 0.5)

## S4 method for signature 'Geom,numeric'
d(distr, x, log = FALSE)

## S4 method for signature 'Geom,numeric'
p(distr, q, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Geom,numeric'
qn(distr, p, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Geom,numeric'
r(distr, n)

## S4 method for signature 'Geom'
mean(x)

## S4 method for signature 'Geom'
median(x)

## S4 method for signature 'Geom'
mode(x)

## S4 method for signature 'Geom'
var(x)

## S4 method for signature 'Geom'
sd(x)

## S4 method for signature 'Geom'
skew(x)
```

```
## S4 method for signature 'Geom'
kurt(x)

## S4 method for signature 'Geom'
entro(x)

## S4 method for signature 'Geom'
finf(x)

llgeom(x, prob)

## S4 method for signature 'Geom,numeric'
ll(distr, x)

egeom(x, type = "mle", ...)

## S4 method for signature 'Geom,numeric'
mle(distr, x, na.rm = FALSE)

## S4 method for signature 'Geom,numeric'
me(distr, x, na.rm = FALSE)

vgeom(prob, type = "mle")

## S4 method for signature 'Geom'
avar_mle(distr)

## S4 method for signature 'Geom'
avar_me(distr)
```

Arguments

`prob`	numeric. Probability of success.
`distr`	an object of class `Geom`.
`x`	For the density function, `x` is a numeric vector of quantiles. For the moments functions, `x` is an object of class `Geom`. For the log-likelihood and the estimation functions, `x` is the sample of observations.
`log, log.p`	logical. Should the logarithm of the probability be returned?
`q`	numeric. Vector of quantiles.
`lower.tail`	logical. If TRUE (default), probabilities are $P(X \leq x)$, otherwise $P(X > x)$.
`p`	numeric. Vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`type`	character, case ignored. The estimator type (mle or me).
`...`	extra arguments.
`na.rm`	logical. Should the `NA` values be removed?

Details

The probability mass function (PMF) of the Geometric distribution is:

$$P(X = k) = (1 - p)^k p, \quad k \in \mathbb{N}_0.$$

Value

Each type of function returns a different type of object:

- Distribution Functions: When supplied with one argument (`distr`), the `d()`, `p()`, `q()`, `r()`, `ll()` functions return the density, cumulative probability, quantile, random sample generator, and log-likelihood functions, respectively. When supplied with both arguments (`distr` and `x`), they evaluate the aforementioned functions directly.
- Moments: Returns a numeric, either vector or matrix depending on the moment and the distribution. The `moments()` function returns a list with all the available methods.
- Estimation: Returns a list, the estimators of the unknown parameters. Note that in distribution families like the binomial, multinomial, and negative binomial, the size is not returned, since it is considered known.
- Variance: Returns a named matrix. The asymptotic covariance matrix of the estimator.

See Also

Functions from the `stats` package: `dgeom()`, `pgeom()`, `qgeom()`, `rgeom()`

Examples

```
# -----------------------------------------------------
# Geom Distribution Example
# -----------------------------------------------------

# Create the distribution
p <- 0.4
D <- Geom(p)

# ------------------
# dpqr Functions
# ------------------

d(D, 0:4) # density function
p(D, 0:4) # distribution function
qn(D, c(0.4, 0.8)) # inverse distribution function
x <- r(D, 100) # random generator function

# alternative way to use the function
df <- d(D) ; df(x) # df is a function itself

# ------------------
# Moments
# ------------------
```

```
mean(D) # Expectation
median(D) # Median
mode(D) # Mode
var(D) # Variance
sd(D) # Standard Deviation
skew(D) # Skewness
kurt(D) # Excess Kurtosis
entro(D) # Entropy
finf(D) # Fisher Information Matrix

# List of all available moments
mom <- moments(D)
mom$mean # expectation

# ------------------
# Point Estimation
# ------------------

ll(D, x)
llgeom(x, p)

egeom(x, type = "mle")
egeom(x, type = "me")

mle(D, x)
me(D, x)
e(D, x, type = "mle")

mle("geom", x) # the distr argument can be a character

# ------------------
# Estimator Variance
# ------------------

vgeom(p, type = "mle")
vgeom(p, type = "me")

avar_mle(D)
avar_me(D)

v(D, type = "mle")
```

idigamma *Inverse Digamma Function*

Description

The inverse of the digamma function, i.e. the derivative of the log-gamma function.

Usage

```
idigamma(x, ...)
```

Arguments

`x`	numeric. The point to evaluate the function.
`...`	extra arguments passed to `optim()`.

Details

The `idigamma()` function implements the inverse of the digamma function ψ. It is a numerical approximation based on the Brent optimization algorithm. Specifically, `idigamma()` makes a call to `optim()` in order to solve the equation $\psi(x) = y$; more accurately, to find the minimum of $f(x) = \log\Gamma(x) - xy$, whose derivative is $f'(x) = \psi(x) - y$. The optimization is restricted within the tight bounds derived by Batir (2017). The function is vectorized.

Value

numeric. The evaluated function.

References

Necdet Batir (2017), INEQUALITIES FOR THE INVERSES OF THE POLYGAMMA FUNCTIONS https://arxiv.org/pdf/1705.06547

Oikonomidis, I. & Trevezas, S. (2023), Moment-Type Estimators for the Dirichlet and the Multivariate Gamma Distributions, arXiv, https://arxiv.org/abs/2311.15025

See Also

`optim()`

Examples

```
idigamma(2)
```

`Laplace` *Laplace Distribution*

Description

The Laplace distribution, also known as the double exponential distribution, is a continuous probability distribution that is often used to model data with sharp peaks and heavy tails. It is parameterized by a location parameter μ and a scale parameter $b > 0$.

Usage

```
Laplace(mu = 0, sigma = 1)

dlaplace(x, mu, sigma, log = FALSE)

plaplace(q, mu, sigma, lower.tail = TRUE, log.p = FALSE)

qlaplace(p, mu, sigma, lower.tail = TRUE, log.p = FALSE)

rlaplace(n, mu, sigma)

## S4 method for signature 'Laplace,numeric'
d(distr, x, log = FALSE)

## S4 method for signature 'Laplace,numeric'
p(distr, q, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Laplace,numeric'
qn(distr, p, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Laplace,numeric'
r(distr, n)

## S4 method for signature 'Laplace'
mean(x)

## S4 method for signature 'Laplace'
median(x)

## S4 method for signature 'Laplace'
mode(x)

## S4 method for signature 'Laplace'
var(x)

## S4 method for signature 'Laplace'
sd(x)

## S4 method for signature 'Laplace'
skew(x)

## S4 method for signature 'Laplace'
kurt(x)

## S4 method for signature 'Laplace'
entro(x)

## S4 method for signature 'Laplace'
```

```
finf(x)

lllaplace(x, mu, sigma)

## S4 method for signature 'Laplace,numeric'
ll(distr, x)

elaplace(x, type = "mle", ...)

## S4 method for signature 'Laplace,numeric'
mle(distr, x, na.rm = FALSE)

## S4 method for signature 'Laplace,numeric'
me(distr, x, na.rm = FALSE)

vlaplace(mu, sigma, type = "mle")

## S4 method for signature 'Laplace'
avar_mle(distr)

## S4 method for signature 'Laplace'
avar_me(distr)
```

Arguments

`mu, sigma`	numeric. The distribution parameters.
`x`	For the density function, `x` is a numeric vector of quantiles. For the moments functions, `x` is an object of class `Laplace`. For the log-likelihood and the estimation functions, `x` is the sample of observations.
`log, log.p`	logical. Should the logarithm of the probability be returned?
`q`	numeric. Vector of quantiles.
`lower.tail`	logical. If TRUE (default), probabilities are $P(X \leq x)$, otherwise $P(X > x)$.
`p`	numeric. Vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`distr`	an object of class `Laplace`.
`type`	character, case ignored. The estimator type (mle or me).
`...`	extra arguments.
`na.rm`	logical. Should the `NA` values be removed?

Details

The probability density function (PDF) of the Laplace distribution is:

$$f(x; \mu, b) = \frac{1}{2b} \exp\left(-\frac{|x - \mu|}{b}\right).$$

Value

Each type of function returns a different type of object:

- Distribution Functions: When supplied with one argument (distr), the d(), p(), q(), r(), ll() functions return the density, cumulative probability, quantile, random sample generator, and log-likelihood functions, respectively. When supplied with both arguments (distr and x), they evaluate the aforementioned functions directly.
- Moments: Returns a numeric, either vector or matrix depending on the moment and the distribution. The moments() function returns a list with all the available methods.
- Estimation: Returns a list, the estimators of the unknown parameters. Note that in distribution families like the binomial, multinomial, and negative binomial, the size is not returned, since it is considered known.
- Variance: Returns a named matrix. The asymptotic covariance matrix of the estimator.

Examples

```
# -----------------------------------------------------
# Laplace Distribution Example
# -----------------------------------------------------

# Create the distribution
m <- 3 ; s <- 5
D <- Laplace(m, s)

# ------------------
# dpqr Functions
# ------------------

d(D, c(0.3, 2, 10)) # density function
p(D, c(0.3, 2, 10)) # distribution function
qn(D, c(0.4, 0.8)) # inverse distribution function
x <- r(D, 100) # random generator function

# alternative way to use the function
df <- d(D) ; df(x) # df is a function itself

# ------------------
# Moments
# ------------------

mean(D) # Expectation
median(D) # Median
mode(D) # Mode
var(D) # Variance
sd(D) # Standard Deviation
skew(D) # Skewness
kurt(D) # Excess Kurtosis
entro(D) # Entropy
finf(D) # Fisher Information Matrix

# List of all available moments
```

```
mom <- moments(D)
mom$mean # expectation

# ------------------
# Point Estimation
# ------------------

elaplace(x, type = "mle")
elaplace(x, type = "me")

mle(D, x)
me(D, x)
e(D, x, type = "mle")

mle("laplace", x) # the distr argument can be a character

# ------------------
# Estimator Variance
# ------------------

vlaplace(m, s, type = "mle")
vlaplace(m, s, type = "me")

avar_mle(D)
avar_me(D)

v(D, type = "mle")
```

LargeMetrics *Large Sample Metrics*

Description

This function calculates the asymptotic variance - covariance matrix characterizing the large sample (asymptotic) behavior of an estimator. The function evaluates the metrics as a function of a single parameter, keeping the other ones constant. See Details.

Usage

```
LargeMetrics(D, est, df)

large_metrics(D, prm, est = c("same", "me", "mle"), ...)
```

Arguments

D	A subclass of `Distribution`. The distribution family of interest.
est	character. The estimator of interest. Can be a vector.
df	data.frame. a data.frame with columns named "Row", "Col", "Parameter", "Estimator", and "Value".

prm	A list containing three elements (name, pos, val). See Details.
...	extra arguments.

Details

The distribution D is used to specify an initial distribution. The list prm contains details concerning a single parameter that is allowed to change values. The quantity of interest is evaluated as a function of this parameter.

The prm list includes two elements named "name" and "val". The first one specifies the parameter that changes, and the second one is a numeric vector holding the values it takes.

In case the parameter of interest is a vector, a third element named "pos" can be specified to indicate the exact parameter that changes. In the example shown below, the evaluation will be performed for the Dirichlet distributions with shape parameters (0.5, 1), (0.6, 1), ..., (2, 1). Notice that the initial shape parameter value (1) is not utilized in the function.

Value

An object of class LargeMetrics with slots D, est, and df.

See Also

SmallMetrics, PlotMetrics

Examples

```
# -----------------------------------------------------
# Beta Distribution Example
# -----------------------------------------------------

D <- Beta(shape1 = 1, shape2 = 2)

prm <- list(name = "shape1",
            val = seq(0.5, 2, by = 0.1))

x <- large_metrics(D, prm,
                   est = c("mle", "me", "same"))

plot(x)

# -----------------------------------------------------
# Dirichlet Distribution Example
# -----------------------------------------------------

D <- Dir(alpha = 1:2)

prm <- list(name = "alpha",
            pos = 1,
            val = seq(0.5, 2, by = 0.1))

x <- large_metrics(D, prm,
                   est = c("mle", "me", "same"))
```

```
plot(x)
```

Lnorm *Log-Normal Distribution*

Description

The Lognormal distribution is an absolute continuous probability distribution of a random variable whose logarithm is normally distributed. It is defined by parameters μ and $\sigma > 0$, which are the mean and standard deviation of the underlying normal distribution.

Usage

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

## S4 method for signature 'Lnorm,numeric'
d(distr, x, log = FALSE)

## S4 method for signature 'Lnorm,numeric'
p(distr, q, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Lnorm,numeric'
qn(distr, p, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Lnorm,numeric'
r(distr, n)

## S4 method for signature 'Lnorm'
mean(x)

## S4 method for signature 'Lnorm'
median(x)

## S4 method for signature 'Lnorm'
mode(x)

## S4 method for signature 'Lnorm'
var(x)

## S4 method for signature 'Lnorm'
sd(x)

## S4 method for signature 'Lnorm'
skew(x)
```

```
## S4 method for signature 'Lnorm'
kurt(x)

## S4 method for signature 'Lnorm'
entro(x)

## S4 method for signature 'Lnorm'
finf(x)

lllnorm(x, meanlog, sdlog)

## S4 method for signature 'Lnorm,numeric'
ll(distr, x)

elnorm(x, type = "mle", ...)

## S4 method for signature 'Lnorm,numeric'
mle(distr, x, na.rm = FALSE)

## S4 method for signature 'Lnorm,numeric'
me(distr, x, na.rm = FALSE)

vlnorm(meanlog, sdlog, type = "mle")

## S4 method for signature 'Lnorm'
avar_mle(distr)

## S4 method for signature 'Lnorm'
avar_me(distr)
```

Arguments

`meanlog, sdlog`	numeric. The distribution parameters.
`distr`	an object of class `Lnorm`.
`x`	For the density function, `x` is a numeric vector of quantiles. For the moments functions, `x` is an object of class `Lnorm`. For the log-likelihood and the estimation functions, `x` is the sample of observations.
`log, log.p`	logical. Should the logarithm of the probability be returned?
`q`	numeric. Vector of quantiles.
`lower.tail`	logical. If TRUE (default), probabilities are $P(X \leq x)$, otherwise $P(X > x)$.
`p`	numeric. Vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`type`	character, case ignored. The estimator type (mle or me).
`...`	extra arguments.
`na.rm`	logical. Should the `NA` values be removed?

Details

The probability density function (PDF) of the Lognormal distribution is:

$$f(x; \mu, \sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\log x - \mu)^2}{2\sigma^2}}, \quad x > 0.$$

Value

Each type of function returns a different type of object:

- Distribution Functions: When supplied with one argument (`distr`), the `d()`, `p()`, `q()`, `r()`, `ll()` functions return the density, cumulative probability, quantile, random sample generator, and log-likelihood functions, respectively. When supplied with both arguments (`distr` and `x`), they evaluate the aforementioned functions directly.
- Moments: Returns a numeric, either vector or matrix depending on the moment and the distribution. The `moments()` function returns a list with all the available methods.
- Estimation: Returns a list, the estimators of the unknown parameters. Note that in distribution families like the binomial, multinomial, and negative binomial, the size is not returned, since it is considered known.
- Variance: Returns a named matrix. The asymptotic covariance matrix of the estimator.

See Also

Functions from the `stats` package: `dlnorm()`, `plnorm()`, `qlnorm()`, `rlnorm()`

Examples

```
# -----------------------------------------------------
# Lnorm Distribution Example
# -----------------------------------------------------

# Create the distribution
m <- 3 ; s <- 5
D <- Lnorm(m, s)

# ------------------
# dpqr Functions
# ------------------

d(D, c(0.3, 2, 10)) # density function
p(D, c(0.3, 2, 10)) # distribution function
qn(D, c(0.4, 0.8)) # inverse distribution function
x <- r(D, 100) # random generator function

# alternative way to use the function
df <- d(D) ; df(x) # df is a function itself

# ------------------
# Moments
# ------------------
```

```
mean(D) # Expectation
median(D) # Median
mode(D) # Mode
var(D) # Variance
sd(D) # Standard Deviation
skew(D) # Skewness
kurt(D) # Excess Kurtosis
entro(D) # Entropy
finf(D) # Fisher Information Matrix

# List of all available moments
mom <- moments(D)
mom$mean # expectation

# ------------------
# Point Estimation
# ------------------

elnorm(x, type = "mle")
elnorm(x, type = "me")

mle(D, x)
me(D, x)
e(D, x, type = "mle")

mle("lnorm", x) # the distr argument can be a character

# ------------------
# Estimator Variance
# ------------------

vlnorm(m, s, type = "mle")
vlnorm(m, s, type = "me")

avar_mle(D)
avar_me(D)

v(D, type = "mle")
```

`loglikelihood` *Log-Likelihood Function*

Description

This function calculates the log-likelihood of an independent and identically distributed (iid) sample from a distribution. See Details.

Usage

```
ll(distr, x, ...)
```

Arguments

distr	A Distribution object
x	numeric. A sample under estimation.
...	extra arguments.

Details

The log-likelihood functions are provided in two forms: the ll<name> distribution-specific version that follows the stats package conventions, and the S4 generic ll. Examples for the ll<name> version are included in the distribution-specific help pages, e.g. ?Beta (all distributions can be found in the See Also section of the Distributions help page).

As with the d(), p(), q(), r() methods, ll() can be supplied only with distr to return the log-likelihood function (i.e. it can be used as a functional), or with both distr and x to be evaluated directly.

In some distribution families like beta and gamma, the MLE cannot be explicitly derived and numerical optimization algorithms have to be employed. Even in “good" scenarios, with plenty of observations and a smooth optimization function, extra care should be taken to ensure a fast and right convergence if possible. Two important steps are taken in package in this direction:

1. The log-likelihood function is analytically calculated for each distribution family, so that constant terms with respect to the parameters can be removed, leaving only the sufficient statistics as a requirement for the function evaluation.
2. Multidimensional problems are reduced to unidimensional ones by utilizing the score equations.

The resulting function that is inserted in the optimization algorithm is called lloptim(), and is not to be confused with the actual log-likelihood function ll(). The corresponding derivative is called dlloptim(). These functions are used internally and are not exported.

Therefore, whenever numerical computation of the MLE is required, optim() is called to optimize lloptim(), using the ME or SAME as the starting point (user's choice), and the L-BFGS-U optimization algorithm, with lower and upper limits defined by default as the parameter space boundary. Illustrative examples can be found in the package vignette.

Value

If only the distr argument is supplied, ll() returns a function. If both distr and x are supplied, ll() returns a numeric, the value of the log-likelihood function.

See Also

distributions, moments, estimation

Examples

```
# -----------------------------------------------------
# Beta Distribution Example
# -----------------------------------------------------
```

```
# Create the distribution
a <- 3
b <- 5
D <- Beta(a, b)

# ------------------
# dpqr Functions
# ------------------

d(D, c(0.3, 0.8, 0.5)) # density function
p(D, c(0.3, 0.8, 0.5)) # distribution function
qn(D, c(0.4, 0.8)) # inverse distribution function
x <- r(D, 100) # random generator function

# alternative way to use the function
df <- d(D) ; df(x) # df is a function itself

# ------------------
# Moments
# ------------------

mean(D) # Expectation
var(D) # Variance
sd(D) # Standard Deviation
skew(D) # Skewness
kurt(D) # Excess Kurtosis
entro(D) # Entropy
finf(D) # Fisher Information Matrix

# List of all available moments
mom <- moments(D)
mom$mean # expectation

# ------------------
# Point Estimation
# ------------------

ll(D, x)
llbeta(x, a, b)

ebeta(x, type = "mle")
ebeta(x, type = "me")
ebeta(x, type = "same")

mle(D, x)
me(D, x)
same(D, x)
e(D, x, type = "mle")

mle("beta", x) # the distr argument can be a character

# ------------------
# Estimator Variance
```

```
# ------------------

vbeta(a, b, type = "mle")
vbeta(a, b, type = "me")
vbeta(a, b, type = "same")

avar_mle(D)
avar_me(D)
avar_same(D)

v(D, type = "mle")
```

---

moments *Moments - Parametric Quantities of Interest*

Description

A set of functions that calculate the theoretical moments (expectation, variance, skewness, excess kurtosis) and other important parametric functions (median, mode, entropy, Fisher information) of a distribution.

Usage

```
moments(x)

mean(x, ...)

median(x, na.rm = FALSE, ...)

mode(x)

var(x, y = NULL, na.rm = FALSE, use)

sd(x, na.rm = FALSE)

skew(x, ...)

kurt(x, ...)

entro(x, ...)

finf(x, ...)
```

Arguments

x	a `Distribution` object.
...	extra arguments.
y, use, na.rm	arguments in `mean` and `var` standard methods from the `stats` package not used here.

Details

Given a distribution, these functions calculate the theoretical moments and other parametric quantities of interest. Some distribution-function combinations are not available; for example, the `sd()` function is available only for univariate distributions.

The `moments()` function automatically finds the available methods for a given distribution and results all of the results in a list.

Technical Note: The argument of the moment functions does not follow the naming convention of the package, i.e. the `Distribution` object is names `x` rather than `distr`. This is due to the fact that most of the generics are already defined in the `stats` package (`mean`, `median`, `mode`, `var`, `sd`), therefore the first argument was already named `x` and could not change.

Value

Numeric, either vector or matrix depending on the moment and the distribution. The `moments()` function returns a list with all the available methods.

Functions

- `median()`: Median
- `mode()`: Mode
- `var()`: Variance
- `sd()`: Standard Deviation
- `skew()`: Skewness
- `kurt()`: Kurtosis
- `entro()`: Entropy
- `finf()`: Fisher Information (numeric or matrix)

See Also

distributions, loglikelihood, estimation

Examples

```
# -----------------------------------------------------
# Beta Distribution Example
# -----------------------------------------------------

# Create the distribution
a <- 3
b <- 5
D <- Beta(a, b)

# ------------------
# dpqr Functions
# ------------------

d(D, c(0.3, 0.8, 0.5)) # density function
```

```
p(D, c(0.3, 0.8, 0.5)) # distribution function
qn(D, c(0.4, 0.8)) # inverse distribution function
x <- r(D, 100) # random generator function

# alternative way to use the function
df <- d(D) ; df(x) # df is a function itself

# ------------------
# Moments
# ------------------

mean(D) # Expectation
var(D) # Variance
sd(D) # Standard Deviation
skew(D) # Skewness
kurt(D) # Excess Kurtosis
entro(D) # Entropy
finf(D) # Fisher Information Matrix

# List of all available moments
mom <- moments(D)
mom$mean # expectation

# ------------------
# Point Estimation
# ------------------

ll(D, x)
llbeta(x, a, b)

ebeta(x, type = "mle")
ebeta(x, type = "me")
ebeta(x, type = "same")

mle(D, x)
me(D, x)
same(D, x)
e(D, x, type = "mle")

mle("beta", x) # the distr argument can be a character

# ------------------
# Estimator Variance
# ------------------

vbeta(a, b, type = "mle")
vbeta(a, b, type = "me")
vbeta(a, b, type = "same")

avar_mle(D)
avar_me(D)
avar_same(D)
```

```
v(D, type = "mle")
```

Multigam *Multivariate Gamma Distribution*

Description

The multivariate gamma distribution is a multivariate absolute continuous probability distribution, defined as the cumulative sum of independent gamma random variables with possibly different shape parameters $\alpha_i > 0, i \in \{1, \dots, k\}$ and the same scale $\beta > 0$.

Usage

```
Multigam(shape = 1, scale = 1)

dmultigam(x, shape, scale, log = FALSE)

rmultigam(n, shape, scale)

## S4 method for signature 'Multigam,numeric'
d(distr, x, log = FALSE)

## S4 method for signature 'Multigam,matrix'
d(distr, x, log = FALSE)

## S4 method for signature 'Multigam,numeric'
r(distr, n)

## S4 method for signature 'Multigam'
mean(x)

## S4 method for signature 'Multigam'
var(x)

## S4 method for signature 'Multigam'
finf(x)

llmultigam(x, shape, scale)

## S4 method for signature 'Multigam,matrix'
ll(distr, x)

emultigam(x, type = "mle", ...)

## S4 method for signature 'Multigam,matrix'
mle(
  distr,
```

```
  x,
  par0 = "same",
  method = "L-BFGS-B",
  lower = 1e-05,
  upper = Inf,
  na.rm = FALSE
)

## S4 method for signature 'Multigam,matrix'
me(distr, x, na.rm = FALSE)

## S4 method for signature 'Multigam,matrix'
same(distr, x, na.rm = FALSE)

vmultigam(shape, scale, type = "mle")

## S4 method for signature 'Multigam'
avar_mle(distr)

## S4 method for signature 'Multigam'
avar_me(distr)

## S4 method for signature 'Multigam'
avar_same(distr)
```

Arguments

shape, scale	numeric. The non-negative distribution parameters.
x	For the density function, x is a numeric vector of quantiles. For the moments functions, x is an object of class Multigam. For the log-likelihood and the estimation functions, x is the sample of observations.
log	logical. Should the logarithm of the probability be returned?
n	number of observations. If length(n) > 1, the length is taken to be the number required.
distr	an object of class Multigam.
type	character, case ignored. The estimator type (mle, me, or same).
...	extra arguments.
par0, method, lower, upper	arguments passed to optim for the mle optimization. See Details.
na.rm	logical. Should the NA values be removed?

Details

The probability density function (PDF) of the multivariate gamma distribution is given by:

$$f(x;\alpha,\beta) = \frac{\beta^{-\alpha_0}}{\prod_{i=1}^k \Gamma(\alpha_i)}, e^{-x_k/\beta} x_1^{\alpha_1-1} \prod_{i=1}^k (x_i - x_{i-1})^{(\alpha_i - 1)} \quad x > 0.$$

The MLE of the multigamma distribution parameters is not available in closed form and has to be approximated numerically. This is done with `optim()`. Specifically, instead of solving a $(k+1)$ optimization problem w.r.t α, β, the optimization can be performed on the shape parameter sum $\alpha_0 := \sum_{i=1}^k \alpha \in (0, +\infty)^k$. The default method used is the L-BFGS-B method with lower bound `1e-5` and upper bound `Inf`. The `par0` argument can either be a numeric (satisfying `lower <= par0 <= upper`) or a character specifying the closed-form estimator to be used as initialization for the algorithm (`"me"` or `"same"` - the default value).

Value

Each type of function returns a different type of object:

- Distribution Functions: When supplied with one argument (`distr`), the `d()`, `p()`, `q()`, `r()`, `ll()` functions return the density, cumulative probability, quantile, random sample generator, and log-likelihood functions, respectively. When supplied with both arguments (`distr` and `x`), they evaluate the aforementioned functions directly.
- Moments: Returns a numeric, either vector or matrix depending on the moment and the distribution. The `moments()` function returns a list with all the available methods.
- Estimation: Returns a list, the estimators of the unknown parameters. Note that in distribution families like the binomial, multinomial, and negative binomial, the size is not returned, since it is considered known.
- Variance: Returns a named matrix. The asymptotic covariance matrix of the estimator.

References

- Mathal, A. M., & Moschopoulos, P. G. (1992). A form of multivariate gamma distribution. Annals of the Institute of Statistical Mathematics, 44, 97-106.
- Oikonomidis, I. & Trevezas, S. (2025), Moment-Type Estimators for the Dirichlet and the Multivariate Gamma Distributions, arXiv, https://arxiv.org/abs/2311.15025

Examples

```
# -----------------------------------------------------
# Multivariate Gamma Distribution Example
# -----------------------------------------------------

# Create the distribution
a <- c(0.5, 3, 5) ; b <- 5
D <- Multigam(a, b)

# ------------------
# dpqr Functions
# ------------------

d(D, c(0.3, 2, 10)) # density function

# alternative way to use the function
df <- d(D) ; df(c(0.3, 2, 10)) # df is a function itself

x <- r(D, 100) # random generator function
```

```
# ------------------
# Moments
# ------------------

mean(D) # Expectation
var(D) # Variance
finf(D) # Fisher Information Matrix

# List of all available moments
mom <- moments(D)
mom$mean # expectation

# ------------------
# Point Estimation
# ------------------

ll(D, x)
llmultigam(x, a, b)

emultigam(x, type = "mle")
emultigam(x, type = "me")
emultigam(x, type = "same")

mle(D, x)
me(D, x)
same(D, x)
e(D, x, type = "mle")

mle("multigam", x) # the distr argument can be a character

# ------------------
# Estimator Variance
# ------------------

vmultigam(a, b, type = "mle")
vmultigam(a, b, type = "me")
vmultigam(a, b, type = "same")

avar_mle(D)
avar_me(D)
avar_same(D)

v(D, type = "mle")
```

`Multinom` *Multinomial Distribution*

Description

The multinomial distribution is a discrete probability distribution which models the probability of having x successes in n independent categorical trials with success probability vector p.

Usage

```
Multinom(size = 1, prob = c(0.5, 0.5))

## S4 method for signature 'Multinom,numeric'
d(distr, x, log = FALSE)

## S4 method for signature 'Multinom,numeric'
r(distr, n)

## S4 method for signature 'Multinom'
mean(x)

## S4 method for signature 'Multinom'
mode(x)

## S4 method for signature 'Multinom'
var(x)

## S4 method for signature 'Multinom'
entro(x)

## S4 method for signature 'Multinom'
finf(x)

llmultinom(x, size, prob)

## S4 method for signature 'Multinom,matrix'
ll(distr, x)

emultinom(x, type = "mle", ...)

## S4 method for signature 'Multinom,matrix'
mle(distr, x, na.rm = FALSE)

## S4 method for signature 'Multinom,matrix'
me(distr, x, na.rm = FALSE)

vmultinom(size, prob, type = "mle")

## S4 method for signature 'Multinom'
avar_mle(distr)

## S4 method for signature 'Multinom'
avar_me(distr)
```

Arguments

`size` number of trials (zero or more).

`prob`	numeric. Probability of success on each trial.
`distr`	an object of class `Multinom`.
`x`	For the density function, `x` is a numeric vector of quantiles. For the moments functions, `x` is an object of class `Multinom`. For the log-likelihood and the estimation functions, `x` is the sample of observations.
`log`	logical. Should the logarithm of the probability be returned?
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`type`	character, case ignored. The estimator type (mle or me).
`...`	extra arguments.
`na.rm`	logical. Should the `NA` values be removed?

Details

The probability mass function (PMF) of the Multinomial distribution is:

$$P(X_1 = x_1, ..., X_k = x_k) = \frac{n!}{x_1!x_2!...x_k!} \prod_{i=1}^{k} p_i^{x_i},$$

subject to $\sum_{i=1}^{k} x_i = n$.

Value

Each type of function returns a different type of object:

- Distribution Functions: When supplied with one argument (`distr`), the `d()`, `p()`, `q()`, `r()`, `ll()` functions return the density, cumulative probability, quantile, random sample generator, and log-likelihood functions, respectively. When supplied with both arguments (`distr` and `x`), they evaluate the aforementioned functions directly.
- Moments: Returns a numeric, either vector or matrix depending on the moment and the distribution. The `moments()` function returns a list with all the available methods.
- Estimation: Returns a list, the estimators of the unknown parameters. Note that in distribution families like the binomial, multinomial, and negative binomial, the size is not returned, since it is considered known.
- Variance: Returns a named matrix. The asymptotic covariance matrix of the estimator.

See Also

Functions from the `stats` package: `dmultinom()`, `rmultinom()`

Examples

```
# -----------------------------------------------------
# Multinomial Distribution Example
# -----------------------------------------------------

# Create the distribution
```

```
N <- 10 ; p <- c(0.1, 0.2, 0.7)
D <- Multinom(N, p)

# ------------------
# dpqr Functions
# ------------------

d(D, c(2, 3, 5)) # density function

# alternative way to use the function
df <- d(D) ; df(c(2, 3, 5)) # df is a function itself

x <- r(D, 100) # random generator function

# ------------------
# Moments
# ------------------

mean(D) # Expectation
mode(D) # Mode
var(D) # Variance
entro(D) # Entropy
finf(D) # Fisher Information Matrix

# List of all available moments
mom <- moments(D)
mom$mean # expectation

# ------------------
# Point Estimation
# ------------------

ll(D, x)
llmultinom(x, N, p)

emultinom(x, type = "mle")
emultinom(x, type = "me")

mle(D, x)
me(D, x)
e(D, x, type = "mle")

mle("multinom", x) # the distr argument can be a character

# ------------------
# Estimator Variance
# ------------------

vmultinom(N, p, type = "mle")
vmultinom(N, p, type = "me")

avar_mle(D)
avar_me(D)
```

```
v(D, type = "mle")
```

Nbinom *Negative Binomial Distribution*

Description

The Negative Binomial distribution is a discrete probability distribution that models the number of failures before a specified number of successes occurs in a sequence of independent Bernoulli trials. It is defined by parameters $r > 0$ (number of successes) and $0 < p \leq 1$ (probability of success).

Usage

```
Nbinom(size = 1, prob = 0.5)

## S4 method for signature 'Nbinom,numeric'
d(distr, x, log = FALSE)

## S4 method for signature 'Nbinom,numeric'
p(distr, q, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Nbinom,numeric'
qn(distr, p, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Nbinom,numeric'
r(distr, n)

## S4 method for signature 'Nbinom'
mean(x)

## S4 method for signature 'Nbinom'
median(x)

## S4 method for signature 'Nbinom'
mode(x)

## S4 method for signature 'Nbinom'
var(x)

## S4 method for signature 'Nbinom'
sd(x)

## S4 method for signature 'Nbinom'
skew(x)

## S4 method for signature 'Nbinom'
```

```
kurt(x)

## S4 method for signature 'Nbinom'
entro(x)

## S4 method for signature 'Nbinom'
finf(x)

llnbinom(x, size, prob)

## S4 method for signature 'Nbinom,numeric'
ll(distr, x)

enbinom(x, size, type = "mle", ...)

## S4 method for signature 'Nbinom,numeric'
mle(distr, x, na.rm = FALSE)

## S4 method for signature 'Nbinom,numeric'
me(distr, x, na.rm = FALSE)

vnbinom(size, prob, type = "mle")

## S4 method for signature 'Nbinom'
avar_mle(distr)

## S4 method for signature 'Nbinom'
avar_me(distr)
```

Arguments

`size`	number of trials (zero or more).
`prob`	numeric. Probability of success on each trial.
`distr`	an object of class `Nbinom`.
`x`	For the density function, `x` is a numeric vector of quantiles. For the moments functions, `x` is an object of class `Nbinom`. For the log-likelihood and the estimation functions, `x` is the sample of observations.
`log`, `log.p`	logical. Should the logarithm of the probability be returned?
`q`	numeric. Vector of quantiles.
`lower.tail`	logical. If TRUE (default), probabilities are $P(X \leq x)$, otherwise $P(X > x)$.
`p`	numeric. Vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`type`	character, case ignored. The estimator type (mle or me).
`...`	extra arguments.
`na.rm`	logical. Should the `NA` values be removed?

Details

The probability mass function (PMF) of the negative binomial distribution is:

$$P(X = k) = \binom{k + r - 1}{k} (1 - p)^k p^r, \quad k \in \mathbb{N}_0.$$

Value

Each type of function returns a different type of object:

- Distribution Functions: When supplied with one argument (`distr`), the `d()`, `p()`, `q()`, `r()`, `ll()` functions return the density, cumulative probability, quantile, random sample generator, and log-likelihood functions, respectively. When supplied with both arguments (`distr` and `x`), they evaluate the aforementioned functions directly.
- Moments: Returns a numeric, either vector or matrix depending on the moment and the distribution. The `moments()` function returns a list with all the available methods.
- Estimation: Returns a list, the estimators of the unknown parameters. Note that in distribution families like the binomial, multinomial, and negative binomial, the size is not returned, since it is considered known.
- Variance: Returns a named matrix. The asymptotic covariance matrix of the estimator.

See Also

Functions from the `stats` package: `dnbinom()`, `pnbinom()`, `qnbinom()`, `rnbinom()`

Examples

```
# -----------------------------------------------------
# Negative Binomial Distribution Example
# -----------------------------------------------------

# Create the distribution
N <- 10 ; p <- 0.4
D <- Nbinom(N, p)

# ------------------
# dpqr Functions
# ------------------

d(D, 0:4) # density function
p(D, 0:4) # distribution function
qn(D, c(0.4, 0.8)) # inverse distribution function
x <- r(D, 100) # random generator function

# alternative way to use the function
df <- d(D) ; df(x) # df is a function itself

# ------------------
# Moments
# ------------------
```

```
mean(D) # Expectation
median(D) # Median
mode(D) # Mode
var(D) # Variance
sd(D) # Standard Deviation
skew(D) # Skewness
kurt(D) # Excess Kurtosis
entro(D) # Entropy
finf(D) # Fisher Information Matrix

# List of all available moments
mom <- moments(D)
mom$mean # expectation

# ------------------
# Point Estimation
# ------------------

ll(D, x)
llnbinom(x, N, p)

enbinom(x, N, type = "mle")
enbinom(x, N, type = "me")

mle(D, x)
me(D, x)
e(D, x, type = "mle")

# ------------------
# Estimator Variance
# ------------------

vnbinom(N, p, type = "mle")
vnbinom(N, p, type = "me")

avar_mle(D)
avar_me(D)

v(D, type = "mle")
```

Norm *Normal Distribution*

Description

The Normal or Gaussian distribution, is an absolute continuous probability distribution characterized by two parameters: the mean μ and the standard deviation $\sigma > 0$.

Usage

```
Norm(mean = 0, sd = 1)

## S4 method for signature 'Norm,numeric'
d(distr, x, log = FALSE)

## S4 method for signature 'Norm,numeric'
p(distr, q, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Norm,numeric'
qn(distr, p, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Norm,numeric'
r(distr, n)

## S4 method for signature 'Norm'
mean(x)

## S4 method for signature 'Norm'
median(x)

## S4 method for signature 'Norm'
mode(x)

## S4 method for signature 'Norm'
var(x)

## S4 method for signature 'Norm'
sd(x)

## S4 method for signature 'Norm'
skew(x)

## S4 method for signature 'Norm'
kurt(x)

## S4 method for signature 'Norm'
entro(x)

## S4 method for signature 'Norm'
finf(x)

llnorm(x, mean, sd)

## S4 method for signature 'Norm,numeric'
ll(distr, x)

enorm(x, type = "mle", ...)
```

```
## S4 method for signature 'Norm,numeric'
mle(distr, x, na.rm = FALSE)

## S4 method for signature 'Norm,numeric'
me(distr, x, na.rm = FALSE)

vnorm(mean, sd, type = "mle")

## S4 method for signature 'Norm'
avar_mle(distr)

## S4 method for signature 'Norm'
avar_me(distr)
```

Arguments

`mean, sd`	numeric. The distribution parameters.
`distr`	an object of class `Norm`.
`x`	For the density function, `x` is a numeric vector of quantiles. For the moments functions, `x` is an object of class `Norm`. For the log-likelihood and the estimation functions, `x` is the sample of observations.
`log, log.p`	logical. Should the logarithm of the probability be returned?
`q`	numeric. Vector of quantiles.
`lower.tail`	logical. If TRUE (default), probabilities are $P(X \leq x)$, otherwise $P(X > x)$.
`p`	numeric. Vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`type`	character, case ignored. The estimator type (mle or me).
`...`	extra arguments.
`na.rm`	logical. Should the `NA` values be removed?

Details

The probability density function (PDF) of the Normal distribution is:

$$f(x; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}.$$

Value

Each type of function returns a different type of object:

- Distribution Functions: When supplied with one argument (`distr`), the `d()`, `p()`, `q()`, `r()`, `ll()` functions return the density, cumulative probability, quantile, random sample generator, and log-likelihood functions, respectively. When supplied with both arguments (`distr` and `x`), they evaluate the aforementioned functions directly.

- Moments: Returns a numeric, either vector or matrix depending on the moment and the distribution. The `moments()` function returns a list with all the available methods.
- Estimation: Returns a list, the estimators of the unknown parameters. Note that in distribution families like the binomial, multinomial, and negative binomial, the size is not returned, since it is considered known.
- Variance: Returns a named matrix. The asymptotic covariance matrix of the estimator.

See Also

Functions from the `stats` package: `dnorm()`, `pnorm()`, `qnorm()`, `rnorm()`

Examples

```
# -----------------------------------------------------
# Normal Distribution Example
# -----------------------------------------------------

# Create the distribution
m <- 3 ; s <- 5
D <- Norm(m, s)

# ------------------
# dpqr Functions
# ------------------

d(D, c(0.3, 2, 10)) # density function
p(D, c(0.3, 2, 10)) # distribution function
qn(D, c(0.4, 0.8)) # inverse distribution function
x <- r(D, 100) # random generator function

# alternative way to use the function
df <- d(D) ; df(x) # df is a function itself

# ------------------
# Moments
# ------------------

mean(D) # Expectation
median(D) # Median
mode(D) # Mode
var(D) # Variance
sd(D) # Standard Deviation
skew(D) # Skewness
kurt(D) # Excess Kurtosis
entro(D) # Entropy
finf(D) # Fisher Information Matrix

# List of all available moments
mom <- moments(D)
mom$mean # expectation

# ------------------
```

```
# Point Estimation
# ------------------

enorm(x, type = "mle")
enorm(x, type = "me")

mle(D, x)
me(D, x)
e(D, x, type = "mle")

mle("norm", x) # the distr argument can be a character

# ------------------
# Estimator Variance
# ------------------

vnorm(m, s, type = "mle")
vnorm(m, s, type = "me")

avar_mle(D)
avar_me(D)

v(D, type = "mle")
```

---

plot *Plot Metrics*

Description

This function provides an easy way to illustrate objects of class SmallMetrics and LargeMetrics, using the ggplot2 package. See details.

Usage

```
plot(x, y, ...)

## S4 method for signature 'SmallMetrics,missing'
plot(
  x,
  y = NULL,
  colors = NULL,
  title = NULL,
  save = FALSE,
  path = NULL,
  name = "myplot.pdf",
  width = 15,
  height = 8
)
```

```
## S4 method for signature 'LargeMetrics,missing'
plot(
  x,
  y = NULL,
  colors = NULL,
  title = NULL,
  save = FALSE,
  path = NULL,
  name = "myplot.pdf",
  width = 15,
  height = 8
)
```

Arguments

`x`	An object of class `SmallMetrics` or `LargeMetrics`.
`y`	NULL.
`...`	extra arguments.
`colors`	character. The colors to be used in the plot.
`title`	character. The plot title.
`save`	logical. Should the plot be saved?
`path`	A path to the directory in which the plot will be saved.
`name`	character. The name of the output pdf file.
`width`	numeric. The plot width in inches.
`height`	numeric. The plot height in inches.

Details

Objects of class `SmallMetrics` and `LargeMetrics` are returned by the `small_metrics()` and `large_metrics()` functions, respectively.

For the `SmallMetrics`, a grid of line charts is created for each metric and sample size. For the `LargeMetrics`, a grid of line charts is created for each element of the asymptotic variance - covariance matrix.

Each estimator is plotted with a different color and line type. The plot can be saved in pdf format.

Value

The plot is returned invisibly in the form of a `ggplot` object.

See Also

SmallMetrics, LargeMetrics

Examples

```
# -----------------------------------------------------
# Beta Distribution Example
# -----------------------------------------------------

D <- Beta(shape1 = 1, shape2 = 2)

prm <- list(name = "shape1",
            val = seq(0.5, 2, by = 0.1))

x <- small_metrics(D, prm,
                   est = c("mle", "me", "same"),
                   obs = c(20, 50),
                   sam = 1e2,
                   seed = 1)

plot(x)

# -----------------------------------------------------
# Dirichlet Distribution Example
# -----------------------------------------------------

D <- Dir(alpha = 1:2)

prm <- list(name = "alpha",
            pos = 1,
            val = seq(0.5, 2, by = 0.1))

x <- small_metrics(D, prm,
                   est = c("mle", "me", "same"),
                   obs = c(20, 50),
                   sam = 1e2,
                   seed = 1)

plot(x)
```

Pois *Poisson Distribution*

Description

The Poisson distribution is a discrete probability distribution that models the number of events occurring in a fixed interval of time or space, given that the events occur with a constant rate $\lambda > 0$ and independently of the time since the last event.

Usage

```
Pois(lambda = 1)
```

```
## S4 method for signature 'Pois,numeric'
d(distr, x, log = FALSE)

## S4 method for signature 'Pois,numeric'
p(distr, q, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Pois,numeric'
qn(distr, p, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Pois,numeric'
r(distr, n)

## S4 method for signature 'Pois'
mean(x)

## S4 method for signature 'Pois'
median(x)

## S4 method for signature 'Pois'
mode(x)

## S4 method for signature 'Pois'
var(x)

## S4 method for signature 'Pois'
sd(x)

## S4 method for signature 'Pois'
skew(x)

## S4 method for signature 'Pois'
kurt(x)

## S4 method for signature 'Pois'
entro(x)

## S4 method for signature 'Pois'
finf(x)

llpois(x, lambda)

## S4 method for signature 'Pois,numeric'
ll(distr, x)

epois(x, type = "mle", ...)

## S4 method for signature 'Pois,numeric'
```

```
mle(distr, x, na.rm = FALSE)

## S4 method for signature 'Pois,numeric'
me(distr, x, na.rm = FALSE)

vpois(lambda, type = "mle")

## S4 method for signature 'Pois'
avar_mle(distr)

## S4 method for signature 'Pois'
avar_me(distr)
```

Arguments

`lambda`	numeric. The distribution parameter.
`distr`	an object of class `Pois`.
`x`	For the density function, `x` is a numeric vector of quantiles. For the moments functions, `x` is an object of class `Pois`. For the log-likelihood and the estimation functions, `x` is the sample of observations.
`log, log.p`	logical. Should the logarithm of the probability be returned?
`q`	numeric. Vector of quantiles.
`lower.tail`	logical. If TRUE (default), probabilities are $P(X \leq x)$, otherwise $P(X > x)$.
`p`	numeric. Vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`type`	character, case ignored. The estimator type (mle or me).
`...`	extra arguments.
`na.rm`	logical. Should the `NA` values be removed?

Details

The probability mass function (PMF) of the Poisson distribution is:

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}, \quad k \in \mathbb{N}_0.$$

Value

Each type of function returns a different type of object:

- Distribution Functions: When supplied with one argument (`distr`), the `d()`, `p()`, `q()`, `r()`, `ll()` functions return the density, cumulative probability, quantile, random sample generator, and log-likelihood functions, respectively. When supplied with both arguments (`distr` and `x`), they evaluate the aforementioned functions directly.
- Moments: Returns a numeric, either vector or matrix depending on the moment and the distribution. The `moments()` function returns a list with all the available methods.

- Estimation: Returns a list, the estimators of the unknown parameters. Note that in distribution families like the binomial, multinomial, and negative binomial, the size is not returned, since it is considered known.
- Variance: Returns a named matrix. The asymptotic covariance matrix of the estimator.

See Also

Functions from the stats package: dpois(), ppois(), qpois(), rpois()

Examples

```
# -----------------------------------------------------
# Pois Distribution Example
# -----------------------------------------------------

# Create the distribution
lambda <- 5
D <- Pois(lambda)

# ------------------
# dpqr Functions
# ------------------

d(D, 0:10) # density function
p(D, 0:10) # distribution function
qn(D, c(0.4, 0.8)) # inverse distribution function
x <- r(D, 100) # random generator function

# alternative way to use the function
df <- d(D) ; df(x) # df is a function itself

# ------------------
# Moments
# ------------------

mean(D) # Expectation
median(D) # Median
mode(D) # Mode
var(D) # Variance
sd(D) # Standard Deviation
skew(D) # Skewness
kurt(D) # Excess Kurtosis
entro(D) # Entropy
finf(D) # Fisher Information Matrix

# List of all available moments
mom <- moments(D)
mom$mean # expectation

# ------------------
# Point Estimation
# ------------------
```

```
ll(D, x)
llpois(x, lambda)

epois(x, type = "mle")
epois(x, type = "me")

mle(D, x)
me(D, x)
e(D, x, type = "mle")

mle("pois", x) # the distr argument can be a character

# ------------------
# Estimator Variance
# ------------------

vpois(lambda, type = "mle")
vpois(lambda, type = "me")

avar_mle(D)
avar_me(D)

v(D, type = "mle")
```

SmallMetrics *Small Sample Metrics*

Description

This function performs Monte Carlo simulations to estimate the main metrics (bias, variance, and RMSE) characterizing the small (finite) sample behavior of an estimator. The function evaluates the metrics as a function of a single parameter, keeping the other ones constant. See Details.

Usage

```
SmallMetrics(D, est, df)

small_metrics(
  D,
  prm,
  est = c("same", "me", "mle"),
  obs = c(20, 50, 100),
  sam = 10000,
  seed = 1,
  bar = TRUE,
  ...
)
```

Arguments

`D`	A subclass of `Distribution`. The distribution family of interest.
`est`	character. The estimator of interest. Can be a vector.
`df`	data.frame. a data.frame with columns named "Row", "Col", "Parameter", "Estimator", and "Value".
`prm`	A list containing three elements (name, pos, val). See Details.
`obs`	numeric. The size of each sample. Can be a vector.
`sam`	numeric. The number of Monte Carlo samples used to estimate the metrics.
`seed`	numeric. Passed to `set.seed()` for reproducibility.
`bar`	logical. Should a progress bar be printed?
`...`	extra arguments.

Details

The distribution `D` is used to specify an initial distribution. The list `prm` contains details concerning a single parameter that is allowed to change values. The quantity of interest is evaluated as a function of this parameter.

The `prm` list includes two elements named "name" and "val". The first one specifies the parameter that changes, and the second one is a numeric vector holding the values it takes.

In case the parameter of interest is a vector, a third element named "pos" can be specified to indicate the exact parameter that changes. In the example shown below, the evaluation will be performed for the Dirichlet distributions with shape parameters `(0.5, 1)`, `(0.6, 1)`, ..., `(2, 1)`. Notice that the initial shape parameter value (`1`) is not utilized in the function.

Value

An object of class `SmallMetrics` with slots `D`, `est`, and `df`.

See Also

LargeMetrics, PlotMetrics

Examples

```
# -----------------------------------------------------
# Beta Distribution Example
# -----------------------------------------------------

D <- Beta(shape1 = 1, shape2 = 2)

prm <- list(name = "shape1",
            val = seq(0.5, 2, by = 0.1))

x <- small_metrics(D, prm,
                   est = c("mle", "me", "same"),
                   obs = c(20, 50),
                   sam = 1e2,
```

```
                  seed = 1)

plot(x)

# -----------------------------------------------------
# Dirichlet Distribution Example
# -----------------------------------------------------

D <- Dir(alpha = 1:2)

prm <- list(name = "alpha",
            pos = 1,
            val = seq(0.5, 2, by = 0.1))

x <- small_metrics(D, prm,
                   est = c("mle", "me", "same"),
                   obs = c(20, 50),
                   sam = 1e2,
                   seed = 1)

plot(x)
```

Stud *Student Distribution*

Description

The Student's t-distribution is a continuous probability distribution used primarily in hypothesis testing and in constructing confidence intervals for small sample sizes. It is defined by one parameter: the degrees of freedom $\nu > 0$.

Usage

```
Stud(df = 1)

## S4 method for signature 'Stud,numeric'
d(distr, x, log = FALSE)

## S4 method for signature 'Stud,numeric'
p(distr, q, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Stud,numeric'
qn(distr, p, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Stud,numeric'
r(distr, n)

## S4 method for signature 'Stud'
```

```
mean(x)

## S4 method for signature 'Stud'
median(x)

## S4 method for signature 'Stud'
mode(x)

## S4 method for signature 'Stud'
var(x)

## S4 method for signature 'Stud'
sd(x)

## S4 method for signature 'Stud'
skew(x)

## S4 method for signature 'Stud'
kurt(x)

## S4 method for signature 'Stud'
entro(x)

llt(x, df)

## S4 method for signature 'Stud,numeric'
ll(distr, x)
```

Arguments

`df`	numeric. The distribution degrees of freedom parameter.
`distr`	an object of class `Stud`.
`x`	For the density function, `x` is a numeric vector of quantiles. For the moments functions, `x` is an object of class `Stud`. For the log-likelihood and the estimation functions, `x` is the sample of observations.
`log`, `log.p`	logical. Should the logarithm of the probability be returned?
`q`	numeric. Vector of quantiles.
`lower.tail`	logical. If TRUE (default), probabilities are $P(X \leq x)$, otherwise $P(X > x)$.
`p`	numeric. Vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.

Details

The probability density function (PDF) of the Student's t-distribution is:

$$f(x;\nu) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\,\Gamma\left(\frac{\nu}{2}\right)}\left(1+\frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}}.$$

Value

Each type of function returns a different type of object:

- Distribution Functions: When supplied with one argument (`distr`), the `d()`, `p()`, `q()`, `r()`, `ll()` functions return the density, cumulative probability, quantile, random sample generator, and log-likelihood functions, respectively. When supplied with both arguments (`distr` and `x`), they evaluate the aforementioned functions directly.
- Moments: Returns a numeric, either vector or matrix depending on the moment and the distribution. The `moments()` function returns a list with all the available methods.
- Estimation: Returns a list, the estimators of the unknown parameters. Note that in distribution families like the binomial, multinomial, and negative binomial, the size is not returned, since it is considered known.
- Variance: Returns a named matrix. The asymptotic covariance matrix of the estimator.

See Also

Functions from the `stats` package: `dt()`, `pt()`, `qt()`, `rt()`

Examples

```
# -----------------------------------------------------
# Student Distribution Example
# -----------------------------------------------------

# Create the distribution
df <- 12
D <- Stud(df)

# ------------------
# dpqr Functions
# ------------------

d(D, c(-3, 0, 3)) # density function
p(D, c(-3, 0, 3)) # distribution function
qn(D, c(0.4, 0.8)) # inverse distribution function
x <- r(D, 100) # random generator function

# alternative way to use the function
d1 <- d(D) ; d1(x) # d1 is a function itself

# ------------------
# Moments
# ------------------

mean(D) # Expectation
median(D) # Median
mode(D) # Mode
var(D) # Variance
sd(D) # Standard Deviation
skew(D) # Skewness
```

```
kurt(D) # Excess Kurtosis
entro(D) # Entropy

# List of all available moments
mom <- moments(D)
mom$mean # expectation

# ------------------
# Point Estimation
# ------------------

ll(D, x)
llt(x, df)
```

Unif *Uniform Distribution*

Description

The Uniform distribution is an absolute continuous probability distribution where all intervals of the same length within the distribution's support are equally probable. It is defined by two parameters: the lower bound a and the upper bound b, with $a < b$.

Usage

```
Unif(min = 0, max = 1)

## S4 method for signature 'Unif,numeric'
d(distr, x, log = FALSE)

## S4 method for signature 'Unif,numeric'
p(distr, q, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Unif,numeric'
qn(distr, p, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Unif,numeric'
r(distr, n)

## S4 method for signature 'Unif'
mean(x)

## S4 method for signature 'Unif'
median(x)

## S4 method for signature 'Unif'
mode(x)
```

```
## S4 method for signature 'Unif'
var(x)

## S4 method for signature 'Unif'
sd(x)

## S4 method for signature 'Unif'
skew(x)

## S4 method for signature 'Unif'
kurt(x)

## S4 method for signature 'Unif'
entro(x)

llunif(x, min, max)

## S4 method for signature 'Unif,numeric'
ll(distr, x)

eunif(x, type = "mle", ...)

## S4 method for signature 'Unif,numeric'
mle(distr, x, na.rm = FALSE)

## S4 method for signature 'Unif,numeric'
me(distr, x, na.rm = FALSE)
```

Arguments

`min`, `max`	numeric. The distribution parameters.
`distr`	an object of class `Unif`.
`x`	For the density function, `x` is a numeric vector of quantiles. For the moments functions, `x` is an object of class `Unif`. For the log-likelihood and the estimation functions, `x` is the sample of observations.
`log`, `log.p`	logical. Should the logarithm of the probability be returned?
`q`	numeric. Vector of quantiles.
`lower.tail`	logical. If TRUE (default), probabilities are $P(X \leq x)$, otherwise $P(X > x)$.
`p`	numeric. Vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`type`	character, case ignored. The estimator type (mle or me).
`...`	extra arguments.
`na.rm`	logical. Should the `NA` values be removed?

Details

The probability density function (PDF) of the Uniform distribution is:

$$f(x; a, b) = \frac{1}{b - a}, \quad a \le x \le b.$$

Value

Each type of function returns a different type of object:

- Distribution Functions: When supplied with one argument (`distr`), the `d()`, `p()`, `q()`, `r()`, `ll()` functions return the density, cumulative probability, quantile, random sample generator, and log-likelihood functions, respectively. When supplied with both arguments (`distr` and `x`), they evaluate the aforementioned functions directly.
- Moments: Returns a numeric, either vector or matrix depending on the moment and the distribution. The `moments()` function returns a list with all the available methods.
- Estimation: Returns a list, the estimators of the unknown parameters. Note that in distribution families like the binomial, multinomial, and negative binomial, the size is not returned, since it is considered known.
- Variance: Returns a named matrix. The asymptotic covariance matrix of the estimator.

See Also

Functions from the `stats` package: `dunif()`, `punif()`, `qunif()`, `runif()`

Examples

```
# -----------------------------------------------------
# Uniform Distribution Example
# -----------------------------------------------------

# Create the distribution
a <- 3 ; b <- 5
D <- Unif(a, b)

# ------------------
# dpqr Functions
# ------------------

d(D, c(0.3, 0.8, 0.5)) # density function
p(D, c(0.3, 0.8, 0.5)) # distribution function
qn(D, c(0.4, 0.8)) # inverse distribution function
x <- r(D, 100) # random generator function

# alternative way to use the function
df <- d(D) ; df(x) # df is a function itself

# ------------------
# Moments
# ------------------
```

```
mean(D) # Expectation
var(D) # Variance
sd(D) # Standard Deviation
skew(D) # Skewness
kurt(D) # Excess Kurtosis
entro(D) # Entropy

# List of all available moments
mom <- moments(D)
mom$mean # expectation

# ------------------
# Point Estimation
# ------------------

ll(D, x)
llunif(x, a, b)

eunif(x, type = "mle")
eunif(x, type = "me")

mle(D, x)
me(D, x)
e(D, x, type = "mle")

mle("unif", x) # the distr argument can be a character
```

`variance` *Estimator Variance*

Description

These functions calculate the variance (or variance - covariance matrix in the multidimensional case) of an estimator, given a specified family of distributions and the true parameter values.

Usage

```
v(distr, type, ...)

avar_mle(distr, ...)

avar_me(distr, ...)

avar_same(distr, ...)
```

Arguments

distr	A Distribution object.
type	character, case ignored. The estimator type.
...	extra arguments.

Value

numeric, or matrix for multidimensional cases.

Functions

- `avar_mle()`: Asymptotic Variance of the Maximum Likelihood Estimator
- `avar_me()`: Asymptotic Variance of the Moment Estimator
- `avar_same()`: Asymptotic Variance of the Score-Adjusted Moment Estimator

References

General Textbooks

- Van der Vaart, A. W. (2000), Asymptotic statistics, Vol. 3, Cambridge university press.

Beta and gamma distribution families

- Ye, Z.-S. & Chen, N. (2017), Closed-form estimators for the gamma distribution derived from likelihood equations, The American Statistician 71(2), 177–181.
- Tamae, H., Irie, K. & Kubokawa, T. (2020), A score-adjusted approach to closed-form estimators for the gamma and beta distributions, Japanese Journal of Statistics and Data Science 3, 543–561.
- Mathal, A. & Moschopoulos, P. (1992), A form of multivariate gamma distribution, Annals of the Institute of Statistical Mathematics 44, 97–106.
- Oikonomidis, I. & Trevezas, S. (2023), Moment-Type Estimators for the Dirichlet and the Multivariate Gamma Distributions, arXiv, https://arxiv.org/abs/2311.15025

See Also

avar_mle, avar_me, avar_same

Examples

```
# -----------------------------------------------------
# Beta Distribution Example
# -----------------------------------------------------

# Create the distribution
a <- 3
b <- 5
D <- Beta(a, b)

# ------------------
# dpqr Functions
# ------------------

d(D, c(0.3, 0.8, 0.5)) # density function
p(D, c(0.3, 0.8, 0.5)) # distribution function
qn(D, c(0.4, 0.8)) # inverse distribution function
x <- r(D, 100) # random generator function
```

```
# alternative way to use the function
df <- d(D) ; df(x) # df is a function itself

# ------------------
# Moments
# ------------------

mean(D) # Expectation
var(D) # Variance
sd(D) # Standard Deviation
skew(D) # Skewness
kurt(D) # Excess Kurtosis
entro(D) # Entropy
finf(D) # Fisher Information Matrix

# List of all available moments
mom <- moments(D)
mom$mean # expectation

# ------------------
# Point Estimation
# ------------------

ll(D, x)
llbeta(x, a, b)

ebeta(x, type = "mle")
ebeta(x, type = "me")
ebeta(x, type = "same")

mle(D, x)
me(D, x)
same(D, x)
e(D, x, type = "mle")

mle("beta", x) # the distr argument can be a character

# ------------------
# Estimator Variance
# ------------------

vbeta(a, b, type = "mle")
vbeta(a, b, type = "me")
vbeta(a, b, type = "same")

avar_mle(D)
avar_me(D)
avar_same(D)

v(D, type = "mle")
```

Weib *Weibull Distribution*

Description

The Weibull distribution is an absolute continuous probability distribution, parameterized by a shape parameter $k > 0$ and a scale parameter $\lambda > 0$.

Usage

```
Weib(shape = 1, scale = 1)

## S4 method for signature 'Weib,numeric'
d(distr, x, log = FALSE)

## S4 method for signature 'Weib,numeric'
p(distr, q, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Weib,numeric'
qn(distr, p, lower.tail = TRUE, log.p = FALSE)

## S4 method for signature 'Weib,numeric'
r(distr, n)

## S4 method for signature 'Weib'
mean(x)

## S4 method for signature 'Weib'
median(x)

## S4 method for signature 'Weib'
mode(x)

## S4 method for signature 'Weib'
var(x)

## S4 method for signature 'Weib'
sd(x)

## S4 method for signature 'Weib'
skew(x)

## S4 method for signature 'Weib'
kurt(x)

## S4 method for signature 'Weib'
entro(x)
```

```
llweibull(x, shape, scale)

## S4 method for signature 'Weib,numeric'
ll(distr, x)

eweibull(x, type = "mle", ...)

## S4 method for signature 'Weib,numeric'
mle(
  distr,
  x,
  par0 = "lme",
  method = "L-BFGS-B",
  lower = 1e-05,
  upper = Inf,
  na.rm = FALSE
)

## S4 method for signature 'Weib,numeric'
me(distr, x, par0 = "lme", lower = 0.5, upper = Inf, na.rm = FALSE)
```

Arguments

`shape, scale`	numeric. The non-negative distribution parameters.
`distr`	an object of class `Weib`.
`x`	For the density function, `x` is a numeric vector of quantiles. For the moments functions, `x` is an object of class `Weib`. For the log-likelihood and the estimation functions, `x` is the sample of observations.
`log, log.p`	logical. Should the logarithm of the probability be returned?
`q`	numeric. Vector of quantiles.
`lower.tail`	logical. If TRUE (default), probabilities are $P(X \leq x)$, otherwise $P(X > x)$.
`p`	numeric. Vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`type`	character, case ignored. The estimator type (mle, me or lme).
`...`	extra arguments.
`par0, method, lower, upper`	arguments passed to optim for the mle and me optimization. See Details.
`na.rm`	logical. Should the `NA` values be removed?

Details

The probability density function (PDF) of the Weibull distribution is:

$$f(x; k, \lambda) = \frac{k}{\lambda} \left(\frac{x}{\lambda}\right)^{k-1} \exp\left[-\left(\frac{x}{\lambda}\right)^k\right], \quad x \geq 0.$$

For the parameter estimation, both the MLE and the ME cannot be explicitly derived. However, the L-moment estimator (`type = "lme"`) is available, and is used as initialization for the numerical approximation of the MLE and the ME.

The MLE and ME of the Weibull distribution parameters is not available in closed form and has to be approximated numerically. The optimization can be performed on the shape parameter $k \in (0, +\infty)$.

For the MLE, this is done with `optim()`. The default method used is the L-BFGS-B method with lower bound `1e-5` and upper bound `Inf`. The `par0` argument can either be a numeric (satisfying `lower <= par0 <= upper`) or a character specifying the closed-form estimator to be used as initialization for the algorithm (`"lme"` - the default value).

For the ME, this is done with `uniroot()`. Again, the `par0` argument can either be a numeric (satisfying `lower <= par0 <= upper`) or a character specifying the closed-form estimator to be used as initialization for the algorithm (`"mle"` or `"lme"` - the default value). The lower and upper bounds are set by default to `0.5` and `Inf`, respectively. Note that the ME equations involve the $\Gamma(1 + 1\ k)$, which can become unreliable for small values of `k`, hence the `0.5` lower bound. Specifying a lower bound below `0.5` will result in a warning and be ignored.

Value

Each type of function returns a different type of object:

- Distribution Functions: When supplied with one argument (`distr`), the `d()`, `p()`, `q()`, `r()`, `ll()` functions return the density, cumulative probability, quantile, random sample generator, and log-likelihood functions, respectively. When supplied with both arguments (`distr` and `x`), they evaluate the aforementioned functions directly.
- Moments: Returns a numeric, either vector or matrix depending on the moment and the distribution. The `moments()` function returns a list with all the available methods.
- Estimation: Returns a list, the estimators of the unknown parameters. Note that in distribution families like the binomial, multinomial, and negative binomial, the size is not returned, since it is considered known.
- Variance: Returns a named matrix. The asymptotic covariance matrix of the estimator.

References

Kim, H. M., Jang, Y. H., Arnold, B. C., & Zhao, J. (2024). New efficient estimators for the Weibull distribution. Communications in Statistics-Theory and Methods, 53(13), 4576-4601.

See Also

Functions from the `stats` package: `dweibull()`, `pweibull()`, `qweibull()`, `rweibull()`

Examples

```
# -----------------------------------------------------
# Weibull Distribution Example
# -----------------------------------------------------

# Create the distribution
```

```
a <- 3 ; b <- 5
D <- Weib(a, b)

# ------------------
# dpqr Functions
# ------------------

d(D, c(0.3, 2, 10)) # density function
p(D, c(0.3, 2, 10)) # distribution function
qn(D, c(0.4, 0.8)) # inverse distribution function
x <- r(D, 100) # random generator function

# alternative way to use the function
df <- d(D) ; df(x) # df is a function itself

# ------------------
# Moments
# ------------------

mean(D) # Expectation
median(D) # Median
mode(D) # Mode
var(D) # Variance
sd(D) # Standard Deviation
skew(D) # Skewness
kurt(D) # Excess Kurtosis
entro(D) # Entropy

# List of all available moments
mom <- moments(D)
mom$mean # expectation

# ------------------
# Point Estimation
# ------------------

ll(D, x)
llweibull(x, a, b)

eweibull(x, type = "mle")
eweibull(x, type = "me")
eweibull(x, type = "lme")

mle(D, x)
me(D, x)
e(D, x, type = "mle")

mle("weib", x) # the distr argument can be a character
```

Index

*,Norm,numeric-method *(calculus)*, 15
*,numeric,Norm-method *(calculus)*, 15
+,Norm,Norm-method *(calculus)*, 15
+,Norm,numeric-method *(calculus)*, 15
+,numeric,Norm-method *(calculus)*, 15
-,Norm,Norm-method *(calculus)*, 15
-,Norm,numeric-method *(calculus)*, 15
-,numeric,Norm-method *(calculus)*, 15
/,Norm,numeric-method *(calculus)*, 15

avar_me, *99*
avar_me *(variance)*, 98
avar_me,Bern-method *(Bern)*, 2
avar_me,Beta-method *(Beta)*, 6
avar_me,Binom-method *(Binom)*, 11
avar_me,Cat-method *(Cat)*, 16
avar_me,Chisq-method *(Chisq)*, 23
avar_me,Dir-method *(Dir)*, 27
avar_me,Exp-method *(Exp)*, 37
avar_me,Gam-method *(Gam)*, 46
avar_me,Geom-method *(Geom)*, 51
avar_me,Laplace-method *(Laplace)*, 55
avar_me,Lnorm-method *(Lnorm)*, 61
avar_me,Multigam-method *(Multigam)*, 70
avar_me,Multinom-method *(Multinom)*, 73
avar_me,Nbinom-method *(Nbinom)*, 77
avar_me,Norm-method *(Norm)*, 80
avar_me,Pois-method *(Pois)*, 86
avar_mle, *99*
avar_mle *(variance)*, 98
avar_mle,Bern-method *(Bern)*, 2
avar_mle,Beta-method *(Beta)*, 6
avar_mle,Binom-method *(Binom)*, 11
avar_mle,Cat-method *(Cat)*, 16
avar_mle,Cauchy-method *(Cauchy)*, 20
avar_mle,Chisq-method *(Chisq)*, 23
avar_mle,Dir-method *(Dir)*, 27
avar_mle,Exp-method *(Exp)*, 37
avar_mle,Gam-method *(Gam)*, 46
avar_mle,Geom-method *(Geom)*, 51
avar_mle,Laplace-method *(Laplace)*, 55
avar_mle,Lnorm-method *(Lnorm)*, 61
avar_mle,Multigam-method *(Multigam)*, 70
avar_mle,Multinom-method *(Multinom)*, 73
avar_mle,Nbinom-method *(Nbinom)*, 77
avar_mle,Norm-method *(Norm)*, 80
avar_mle,Pois-method *(Pois)*, 86
avar_same, *99*
avar_same *(variance)*, 98
avar_same,Beta-method *(Beta)*, 6
avar_same,Dir-method *(Dir)*, 27
avar_same,Gam-method *(Gam)*, 46
avar_same,Multigam-method *(Multigam)*, 70

Bern, 2, *32*, *45*
Beta, 6, *32*, *45*
Binom, 11, *32*, *45*

calculus, 14
Cat, 16, *32*, *45*
Cauchy, 19, *32*, *45*
Chisq, 23, *32*, *45*

d *(distributions)*, 31
d,Bern,numeric-method *(Bern)*, 2
d,Beta,numeric-method *(Beta)*, 6
d,Binom,numeric-method *(Binom)*, 11
d,Cat,numeric-method *(Cat)*, 16
d,Cauchy,numeric-method *(Cauchy)*, 20
d,Chisq,numeric-method *(Chisq)*, 23
d,Dir,matrix-method *(Dir)*, 27
d,Dir,numeric-method *(Dir)*, 27
d,Distribution,missing-method *(functionals)*, 44
d,Exp,numeric-method *(Exp)*, 37
d,Fisher,numeric-method *(Fisher)*, 41
d,Gam,numeric-method *(Gam)*, 46
d,Geom,numeric-method *(Geom)*, 51
d,Laplace,numeric-method *(Laplace)*, 55
d,Lnorm,numeric-method *(Lnorm)*, 61

d,Multigam,matrix-method *(Multigam)*, 70
d,Multigam,numeric-method *(Multigam)*, 70
d,Multinom,numeric-method *(Multinom)*, 73
d,Nbinom,numeric-method *(Nbinom)*, 77
d,Norm,numeric-method *(Norm)*, 80
d,Pois,numeric-method *(Pois)*, 86
d,Stud,numeric-method *(Stud)*, 92
d,Unif,numeric-method *(Unif)*, 95
d,Weib,numeric-method *(Weib)*, 101
dbern *(Bern)*, 2
dbeta(), *9*
dbinom(), *5*, *13*
dcat *(Cat)*, 16
dcauchy(), *22*
dchisq(), *26*
ddir *(Dir)*, 27
dexp(), *39*
df(), *43*
dgamma(), *49*
dgeom(), *53*
Dir, 27, *32*, *45*
distributions, 31, *65*, *68*
dlaplace *(Laplace)*, 55
dlnorm(), *63*
dmultigam *(Multigam)*, 70
dmultinom(), *18*, *75*
dnbinom(), *79*
dnorm(), *83*
dpois(), *89*
dt(), *94*
dunif(), *97*
dweibull(), *103*

e *(estimation)*, 34
ebern *(Bern)*, 2
ebeta *(Beta)*, 6
ebinom *(Binom)*, 11
ecat *(Cat)*, 16
ecauchy *(Cauchy)*, 20
echisq *(Chisq)*, 23
edir *(Dir)*, 27
eexp *(Exp)*, 37
egamma *(Gam)*, 46
egeom *(Geom)*, 51
elaplace *(Laplace)*, 55
elnorm *(Lnorm)*, 61
emultigam *(Multigam)*, 70
emultinom *(Multinom)*, 73
enbinom *(Nbinom)*, 77
enorm *(Norm)*, 80
entro *(moments)*, 67
entro,Bern-method *(Bern)*, 2
entro,Beta-method *(Beta)*, 6
entro,Binom-method *(Binom)*, 11
entro,Cat-method *(Cat)*, 16
entro,Cauchy-method *(Cauchy)*, 20
entro,Chisq-method *(Chisq)*, 23
entro,Dir-method *(Dir)*, 27
entro,Exp-method *(Exp)*, 37
entro,Fisher-method *(Fisher)*, 41
entro,Gam-method *(Gam)*, 46
entro,Geom-method *(Geom)*, 51
entro,Laplace-method *(Laplace)*, 55
entro,Lnorm-method *(Lnorm)*, 61
entro,Multinom-method *(Multinom)*, 73
entro,Nbinom-method *(Nbinom)*, 77
entro,Norm-method *(Norm)*, 80
entro,Pois-method *(Pois)*, 86
entro,Stud-method *(Stud)*, 92
entro,Unif-method *(Unif)*, 95
entro,Weib-method *(Weib)*, 101
epois *(Pois)*, 86
estimation, *32*, 34, *45*, *65*, *68*
eunif *(Unif)*, 95
eweibull *(Weib)*, 101
Exp, *32*, 37, *45*
exp,Norm-method *(calculus)*, 15

finf *(moments)*, 67
finf,Bern-method *(Bern)*, 2
finf,Beta-method *(Beta)*, 6
finf,Binom-method *(Binom)*, 11
finf,Cat-method *(Cat)*, 16
finf,Cauchy-method *(Cauchy)*, 20
finf,Chisq-method *(Chisq)*, 23
finf,Dir-method *(Dir)*, 27
finf,Exp-method *(Exp)*, 37
finf,Gam-method *(Gam)*, 46
finf,Geom-method *(Geom)*, 51
finf,Laplace-method *(Laplace)*, 55
finf,Lnorm-method *(Lnorm)*, 61
finf,Multigam-method *(Multigam)*, 70
finf,Multinom-method *(Multinom)*, 73
finf,Nbinom-method *(Nbinom)*, 77
finf,Norm-method *(Norm)*, 80
finf,Pois-method *(Pois)*, 86
Fisher, *32*, 41, *45*
functionals, 44

Gam, *32*, *45*, 46
Geom, *32*, *45*, 51

idigamma, 54

kurt *(moments)*, 67
kurt,Bern-method *(Bern)*, 2
kurt,Beta-method *(Beta)*, 6
kurt,Binom-method *(Binom)*, 11
kurt,Cauchy-method *(Cauchy)*, 20
kurt,Chisq-method *(Chisq)*, 23
kurt,Exp-method *(Exp)*, 37
kurt,Fisher-method *(Fisher)*, 41
kurt,Gam-method *(Gam)*, 46
kurt,Geom-method *(Geom)*, 51
kurt,Laplace-method *(Laplace)*, 55
kurt,Lnorm-method *(Lnorm)*, 61
kurt,Nbinom-method *(Nbinom)*, 77
kurt,Norm-method *(Norm)*, 80
kurt,Pois-method *(Pois)*, 86
kurt,Stud-method *(Stud)*, 92
kurt,Unif-method *(Unif)*, 95
kurt,Weib-method *(Weib)*, 101

Laplace, *32*, *45*, 55
large_metrics *(LargeMetrics)*, 59
LargeMetrics, 59, *85*, *91*
ll *(loglikelihood)*, 64
ll,Bern,numeric-method *(Bern)*, 2
ll,Beta,numeric-method *(Beta)*, 6
ll,Binom,numeric-method *(Binom)*, 11
ll,Cat,numeric-method *(Cat)*, 16
ll,Cauchy,numeric-method *(Cauchy)*, 20
ll,Chisq,numeric-method *(Chisq)*, 23
ll,Dir,matrix-method *(Dir)*, 27
ll,Distribution,missing-method *(functionals)*, 44
ll,Exp,numeric-method *(Exp)*, 37
ll,Fisher,numeric-method *(Fisher)*, 41
ll,Gam,numeric-method *(Gam)*, 46
ll,Geom,numeric-method *(Geom)*, 51
ll,Laplace,numeric-method *(Laplace)*, 55
ll,Lnorm,numeric-method *(Lnorm)*, 61
ll,Multigam,matrix-method *(Multigam)*, 70
ll,Multinom,matrix-method *(Multinom)*, 73
ll,Nbinom,numeric-method *(Nbinom)*, 77
ll,Norm,numeric-method *(Norm)*, 80
ll,Pois,numeric-method *(Pois)*, 86
ll,Stud,numeric-method *(Stud)*, 92
ll,Unif,numeric-method *(Unif)*, 95
ll,Weib,numeric-method *(Weib)*, 101
llbern *(Bern)*, 2
llbeta *(Beta)*, 6
llbinom *(Binom)*, 11
llcat *(Cat)*, 16
llcauchy *(Cauchy)*, 20
llchisq *(Chisq)*, 23
lldir *(Dir)*, 27
llexp *(Exp)*, 37
llf *(Fisher)*, 41
llgamma *(Gam)*, 46
llgeom *(Geom)*, 51
lllaplace *(Laplace)*, 55
lllnorm *(Lnorm)*, 61
llmultigam *(Multigam)*, 70
llmultinom *(Multinom)*, 73
llnbinom *(Nbinom)*, 77
llnorm *(Norm)*, 80
llpois *(Pois)*, 86
llt *(Stud)*, 92
llunif *(Unif)*, 95
llweibull *(Weib)*, 101
Lnorm, *32*, *45*, 61
loglikelihood, *32*, *45*, 64, *68*

me, *35*
me *(estimation)*, 34
me,Bern,numeric-method *(Bern)*, 2
me,Beta,numeric-method *(Beta)*, 6
me,Binom,numeric-method *(Binom)*, 11
me,Cat,numeric-method *(Cat)*, 16
me,Cauchy,numeric-method *(Cauchy)*, 20
me,character,ANY-method *(estimation)*, 34
me,Chisq,numeric-method *(Chisq)*, 23
me,Dir,matrix-method *(Dir)*, 27
me,Distribution,missing-method *(functionals)*, 44
me,Exp,numeric-method *(Exp)*, 37
me,Gam,numeric-method *(Gam)*, 46
me,Geom,numeric-method *(Geom)*, 51
me,Laplace,numeric-method *(Laplace)*, 55
me,Lnorm,numeric-method *(Lnorm)*, 61
me,Multigam,matrix-method *(Multigam)*, 70
me,Multinom,matrix-method *(Multinom)*, 73
me,Nbinom,numeric-method *(Nbinom)*, 77
me,Norm,numeric-method *(Norm)*, 80
me,Pois,numeric-method *(Pois)*, 86
me,Unif,numeric-method *(Unif)*, 95

me,Weib,numeric-method *(*Weib*)*, 101
mean *(*moments*)*, 67
mean,Bern-method *(*Bern*)*, 2
mean,Beta-method *(*Beta*)*, 6
mean,Binom-method *(*Binom*)*, 11
mean,Cat-method *(*Cat*)*, 16
mean,Cauchy-method *(*Cauchy*)*, 20
mean,Chisq-method *(*Chisq*)*, 23
mean,Dir-method *(*Dir*)*, 27
mean,Exp-method *(*Exp*)*, 37
mean,Fisher-method *(*Fisher*)*, 41
mean,Gam-method *(*Gam*)*, 46
mean,Geom-method *(*Geom*)*, 51
mean,Laplace-method *(*Laplace*)*, 55
mean,Lnorm-method *(*Lnorm*)*, 61
mean,Multigam-method *(*Multigam*)*, 70
mean,Multinom-method *(*Multinom*)*, 73
mean,Nbinom-method *(*Nbinom*)*, 77
mean,Norm-method *(*Norm*)*, 80
mean,Pois-method *(*Pois*)*, 86
mean,Stud-method *(*Stud*)*, 92
mean,Unif-method *(*Unif*)*, 95
mean,Weib-method *(*Weib*)*, 101
median *(*moments*)*, 67
median,Bern-method *(*Bern*)*, 2
median,Beta-method *(*Beta*)*, 6
median,Cauchy-method *(*Cauchy*)*, 20
median,Chisq-method *(*Chisq*)*, 23
median,Exp-method *(*Exp*)*, 37
median,Fisher-method *(*Fisher*)*, 41
median,Gam-method *(*Gam*)*, 46
median,Geom-method *(*Geom*)*, 51
median,Laplace-method *(*Laplace*)*, 55
median,Lnorm-method *(*Lnorm*)*, 61
median,Nbinom-method *(*Nbinom*)*, 77
median,Norm-method *(*Norm*)*, 80
median,Pois-method *(*Pois*)*, 86
median,Stud-method *(*Stud*)*, 92
median,Unif-method *(*Unif*)*, 95
median,Weib-method *(*Weib*)*, 101
mle, *35*
mle *(*estimation*)*, 34
mle,Bern,numeric-method *(*Bern*)*, 2
mle,Beta,numeric-method *(*Beta*)*, 6
mle,Binom,numeric-method *(*Binom*)*, 11
mle,Cat,numeric-method *(*Cat*)*, 16
mle,Cauchy,numeric-method *(*Cauchy*)*, 20
mle,character,ANY-method *(*estimation*)*, 34
mle,Chisq,numeric-method *(*Chisq*)*, 23
mle,Dir,matrix-method *(*Dir*)*, 27
mle,Distribution,missing-method *(*functionals*)*, 44
mle,Exp,numeric-method *(*Exp*)*, 37
mle,Gam,numeric-method *(*Gam*)*, 46
mle,Geom,numeric-method *(*Geom*)*, 51
mle,Laplace,numeric-method *(*Laplace*)*, 55
mle,Lnorm,numeric-method *(*Lnorm*)*, 61
mle,Multigam,matrix-method *(*Multigam*)*, 70
mle,Multinom,matrix-method *(*Multinom*)*, 73
mle,Nbinom,numeric-method *(*Nbinom*)*, 77
mle,Norm,numeric-method *(*Norm*)*, 80
mle,Pois,numeric-method *(*Pois*)*, 86
mle,Unif,numeric-method *(*Unif*)*, 95
mle,Weib,numeric-method *(*Weib*)*, 101
mode *(*moments*)*, 67
mode,Bern-method *(*Bern*)*, 2
mode,Beta-method *(*Beta*)*, 6
mode,Cat-method *(*Cat*)*, 16
mode,Cauchy-method *(*Cauchy*)*, 20
mode,Chisq-method *(*Chisq*)*, 23
mode,Dir-method *(*Dir*)*, 27
mode,Exp-method *(*Exp*)*, 37
mode,Fisher-method *(*Fisher*)*, 41
mode,Gam-method *(*Gam*)*, 46
mode,Geom-method *(*Geom*)*, 51
mode,Laplace-method *(*Laplace*)*, 55
mode,Lnorm-method *(*Lnorm*)*, 61
mode,Multinom-method *(*Multinom*)*, 73
mode,Nbinom-method *(*Nbinom*)*, 77
mode,Norm-method *(*Norm*)*, 80
mode,Pois-method *(*Pois*)*, 86
mode,Stud-method *(*Stud*)*, 92
mode,Unif-method *(*Unif*)*, 95
mode,Weib-method *(*Weib*)*, 101
moments, *32*, *45*, *65*, 67
Multigam, *32*, *45*, 70
Multinom, *32*, *45*, 73

Nbinom, *32*, *45*, 77
Norm, *32*, *45*, 80

optim(), *55*

p *(*distributions*)*, 31

p,Bern,numeric-method (Bern), 2
p,Beta,numeric-method (Beta), 6
p,Binom,numeric-method (Binom), 11
p,Cauchy,numeric-method (Cauchy), 20
p,Chisq,numeric-method (Chisq), 23
p,Distribution,missing-method (functionals), 44
p,Exp,numeric-method (Exp), 37
p,Fisher,numeric-method (Fisher), 41
p,Gam,numeric-method (Gam), 46
p,Geom,numeric-method (Geom), 51
p,Laplace,numeric-method (Laplace), 55
p,Lnorm,numeric-method (Lnorm), 61
p,Nbinom,numeric-method (Nbinom), 77
p,Norm,numeric-method (Norm), 80
p,Pois,numeric-method (Pois), 86
p,Stud,numeric-method (Stud), 92
p,Unif,numeric-method (Unif), 95
p,Weib,numeric-method (Weib), 101
pbern (Bern), 2
pbeta(), *9*
pbinom(), *5*, *13*
pcauchy(), *22*
pchisq(), *26*
pexp(), *39*
pf(), *43*
pgamma(), *49*
pgeom(), *53*
plaplace (Laplace), 55
plnorm(), *63*
plot, 84
plot,LargeMetrics,missing-method (plot), 84
plot,SmallMetrics,missing-method (plot), 84
PlotMetrics, *60*, *91*
PlotMetrics (plot), 84
pnbinom(), *79*
pnorm(), *83*
Pois, *32*, *45*, 86
ppois(), *89*
pt(), *94*
punif(), *97*
pweibull(), *103*

q (distributions), 31
qbern (Bern), 2
qbeta(), *9*
qbinom(), *5*, *13*
qcauchy(), *22*
qchisq(), *26*
qexp(), *39*
qf(), *43*
qgamma(), *49*
qgeom(), *53*
qlaplace (Laplace), 55
qlnorm(), *63*
qn (distributions), 31
qn,Bern,numeric-method (Bern), 2
qn,Beta,numeric-method (Beta), 6
qn,Binom,numeric-method (Binom), 11
qn,Cauchy,numeric-method (Cauchy), 20
qn,Chisq,numeric-method (Chisq), 23
qn,Distribution,missing-method (functionals), 44
qn,Exp,numeric-method (Exp), 37
qn,Fisher,numeric-method (Fisher), 41
qn,Gam,numeric-method (Gam), 46
qn,Geom,numeric-method (Geom), 51
qn,Laplace,numeric-method (Laplace), 55
qn,Lnorm,numeric-method (Lnorm), 61
qn,Nbinom,numeric-method (Nbinom), 77
qn,Norm,numeric-method (Norm), 80
qn,Pois,numeric-method (Pois), 86
qn,Stud,numeric-method (Stud), 92
qn,Unif,numeric-method (Unif), 95
qn,Weib,numeric-method (Weib), 101
qnbinom(), *79*
qnorm(), *83*
qpois(), *89*
qt(), *94*
qunif(), *97*
qweibull(), *103*

r (distributions), 31
r,Bern,numeric-method (Bern), 2
r,Beta,numeric-method (Beta), 6
r,Binom,numeric-method (Binom), 11
r,Cat,numeric-method (Cat), 16
r,Cauchy,numeric-method (Cauchy), 20
r,Chisq,numeric-method (Chisq), 23
r,Dir,numeric-method (Dir), 27
r,Distribution,missing-method (functionals), 44
r,Exp,numeric-method (Exp), 37
r,Fisher,numeric-method (Fisher), 41
r,Gam,numeric-method (Gam), 46
r,Geom,numeric-method (Geom), 51

r,Laplace,numeric-method (Laplace), 55
r,Lnorm,numeric-method (Lnorm), 61
r,Multigam,numeric-method (Multigam), 70
r,Multinom,numeric-method (Multinom), 73
r,Nbinom,numeric-method (Nbinom), 77
r,Norm,numeric-method (Norm), 80
r,Pois,numeric-method (Pois), 86
r,Stud,numeric-method (Stud), 92
r,Unif,numeric-method (Unif), 95
r,Weib,numeric-method (Weib), 101
rbern (Bern), 2
rbeta(), *9*
rbinom(), *5*, *13*
rcat (Cat), 16
rcauchy(), *22*
rchisq(), *26*
rdir (Dir), 27
rexp(), *39*
rf(), *43*
rgamma(), *49*
rgeom(), *53*
rlaplace (Laplace), 55
rlnorm(), *63*
rmultigam (Multigam), 70
rmultinom(), *18*, *75*
rnbinom(), *79*
rnorm(), *83*
rpois(), *89*
rt(), *94*
runif(), *97*
rweibull(), *103*

same, *35*
same (estimation), 34
same,Beta,numeric-method (Beta), 6
same,character,ANY-method (estimation), 34
same,Dir,matrix-method (Dir), 27
same,Distribution,missing-method (functionals), 44
same,Gam,numeric-method (Gam), 46
same,Multigam,matrix-method (Multigam), 70
sd (moments), 67
sd,Bern-method (Bern), 2
sd,Beta-method (Beta), 6
sd,Binom-method (Binom), 11
sd,Cauchy-method (Cauchy), 20
sd,Chisq-method (Chisq), 23
sd,Exp-method (Exp), 37
sd,Fisher-method (Fisher), 41
sd,Gam-method (Gam), 46
sd,Geom-method (Geom), 51
sd,Laplace-method (Laplace), 55
sd,Lnorm-method (Lnorm), 61
sd,Nbinom-method (Nbinom), 77
sd,Norm-method (Norm), 80
sd,Pois-method (Pois), 86
sd,Stud-method (Stud), 92
sd,Unif-method (Unif), 95
sd,Weib-method (Weib), 101
skew (moments), 67
skew,Bern-method (Bern), 2
skew,Beta-method (Beta), 6
skew,Binom-method (Binom), 11
skew,Cauchy-method (Cauchy), 20
skew,Chisq-method (Chisq), 23
skew,Exp-method (Exp), 37
skew,Fisher-method (Fisher), 41
skew,Gam-method (Gam), 46
skew,Geom-method (Geom), 51
skew,Laplace-method (Laplace), 55
skew,Lnorm-method (Lnorm), 61
skew,Nbinom-method (Nbinom), 77
skew,Norm-method (Norm), 80
skew,Pois-method (Pois), 86
skew,Stud-method (Stud), 92
skew,Unif-method (Unif), 95
skew,Weib-method (Weib), 101
small_metrics (SmallMetrics), 90
SmallMetrics, *60*, *85*, 90
Stud, *32*, *45*, 92
sum,Norm,logical-method (calculus), 15

Unif, *32*, *45*, 95

v (variance), 98
var (moments), 67
var,Bern-method (Bern), 2
var,Beta-method (Beta), 6
var,Binom-method (Binom), 11
var,Cat-method (Cat), 16
var,Cauchy-method (Cauchy), 20
var,Chisq-method (Chisq), 23
var,Dir-method (Dir), 27
var,Exp-method (Exp), 37
var,Fisher-method (Fisher), 41
var,Gam-method (Gam), 46

var,Geom-method *(*Geom*)*, 51
var,Laplace-method *(*Laplace*)*, 55
var,Lnorm-method *(*Lnorm*)*, 61
var,Multigam-method *(*Multigam*)*, 70
var,Multinom-method *(*Multinom*)*, 73
var,Nbinom-method *(*Nbinom*)*, 77
var,Norm-method *(*Norm*)*, 80
var,Pois-method *(*Pois*)*, 86
var,Stud-method *(*Stud*)*, 92
var,Unif-method *(*Unif*)*, 95
var,Weib-method *(*Weib*)*, 101
variance, 98
vbern *(*Bern*)*, 2
vbeta *(*Beta*)*, 6
vbinom *(*Binom*)*, 11
vcat *(*Cat*)*, 16
vcauchy *(*Cauchy*)*, 20
vchisq *(*Chisq*)*, 23
vdir *(*Dir*)*, 27
vexp *(*Exp*)*, 37
vgamma *(*Gam*)*, 46
vgeom *(*Geom*)*, 51
vlaplace *(*Laplace*)*, 55
vlnorm *(*Lnorm*)*, 61
vmultigam *(*Multigam*)*, 70
vmultinom *(*Multinom*)*, 73
vnbinom *(*Nbinom*)*, 77
vnorm *(*Norm*)*, 80
vpois *(*Pois*)*, 86

Weib, *32*, *45*, 101